Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 17, 2013

E. Abdo
M. Boucadair
J. Queiroz
France Telecom
July 16, 2012

HOST_ID TCP Options: Implementation & Preliminary Test Results
draft-abdo-hostid-tcpopt-implementation-03

Abstract

This memo documents the implementation of the HOST_ID TCP Options. It also discusses the preliminary results of the tests that have been conducted to assess the technical feasibility of the approach as well as its scalability. Several HOST_ID TCP options have been implemented and tested.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 17, 2013.

Table of Contents

1. Introduction

To ensure IPv4 service continuity, service providers will need to deploy IPv4 address sharing techniques. Several issues are likely to be encountered (refer to [RFC6269] for a detailed survey of the issues) and they may affect the delivery of services that depends on the enforcement of policies based upon the source IPv4 address.

Some of these issues may be mitigated owing to the activation of advanced features. Among the solutions analyzed in [I-D.boucadair-intarea-nat-reveal-analysis], the use of a new TCP option to convey a HOST_ID seems to be a promising solution.

This memo documents some implementation and experimentation efforts that have been conducted to assess the viability of using HOST_ID TCP options at large scale. In particular, this document provides experimentation results related to the support of the HOST_ID TCP Options, the behavior of legacy TCP servers when receiving the HOST_ID TCP options. This draft also discusses the impact of using a HOST_ID TCP options on the time it takes to establish a connection; it also tries to evaluate the impact of the new TCP options on the performance of the CGN. Finally it presents the enforcement policies that could be applied by remote servers based upon the HOST_ID options contents.

2. Objectives

The implementation of several HOST_ID TCP options is primarily meant to:

- Assess the validity of the HOST_ID TCP option approach
- Evaluate the impact on the TCP stack to support the HOST_ID TCP options
- Improve filtering and logging capabilities based upon the contents of the HOST_ID TCP option. This means the enforcement of various policies based upon the content of the HOST_ID TCP option at the server side: Log, Deny, Accept, etc.
- Assess the behavior of legacy TCP servers when receiving a HOST_ID TCP option
- Assess the success ratio of TCP communications when a HOST_ID TCP option is received
- Assess the impact of injecting a HOST_ID TCP option on the time it takes to establish a connection
- Assess the performance impact on the CGN device that has been configured to inject the HOST_ID option

3. NAT Reveal TCP Options: Overview

The original idea of defining a TCP option is documented in [I-D.wing-nat-reveal-option] and denoted as HOST_ID_WING.

An additional TCP option is also considered and denoted as HOST_ID_BOUCADAIR. The main motivation is to cover also the load-balancer use case and provide richer functionality as Forwarded-For HTTP header than HOST_ID_WING can provide.

The following sub-sections provide an overview of these HOST_ID TCP options.

3.1. HOST_ID_WING TCP Option

HOST_ID_WING is defined in [I-D.wing-nat-reveal-option]. Figure 1 shows the format of this option.

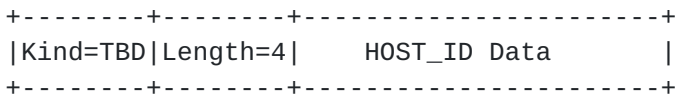

Figure 1: Format of HOST_ID_WING TCP Option

This option must be sent only upon the initial connection request, i.e., in SYN packets as shown in Figure 2

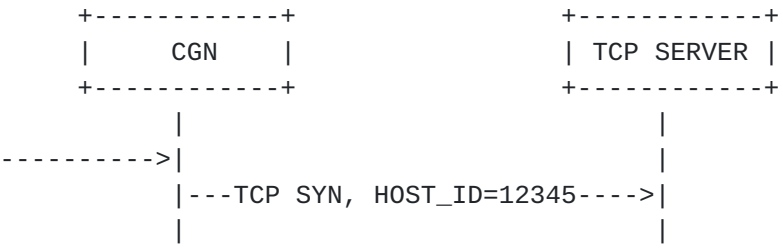

Figure 2: HOST_ID_WING TCP Option: Flow example

3.2. HOST_ID_BOUCADAIR TCP Option

As mentioned above, the HOST_ID_BOUCADAIR TCP Option is inspired from HOST_ID_WING and XFF.

The HOST_ID_BOUCADAIR option is a 10-byte long TCP option, where KIND, Length and lifetime-Origin fields fill one byte each, and HOST_ID data is 7-byte long as shown in Figure 3

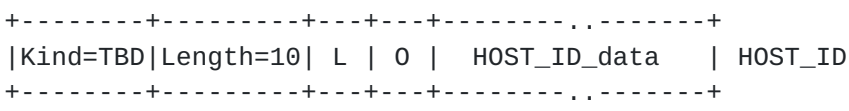

Figure 3: Format of HOST_ID_BOUCADAIR TCP option

- o L: Indicates the validity lifetime of the enclosed data (in the spirit of [RFC6250]). The following values are supported:
  0: Permanent;
  >0:Dynamic; this value indicates the validity time.
- o Origin: Indicates the origin of the data conveyed in the data field. The following values are supported:
  0: Internal Port
  1: Internal IPv4 address
  2: Internal Port: Internal IPv4 address
  3: IPv6 Prefix
  >3: No particular semantic
- o HOST_ID_data depends on the content of the Origin field; padding is required.

Two modes are described below: the SYN mode (Section 3.2.1) and the ACK mode. (Section 3.2.2).

If the ACK mode is used (Section 3.2.2), Figure 4 shows the HOST_ID_ENABLED option (2-bytes long) to be included in the SYN.

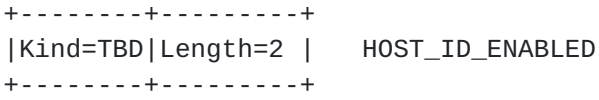

Figure 4: Format of HOST_ID_ENABLED

3.2.1. SYN Mode

This mode is similar to the mode described in Section 3.1. In this mode, HOST_ID_BOUCADAIR is sent in SYN packets.

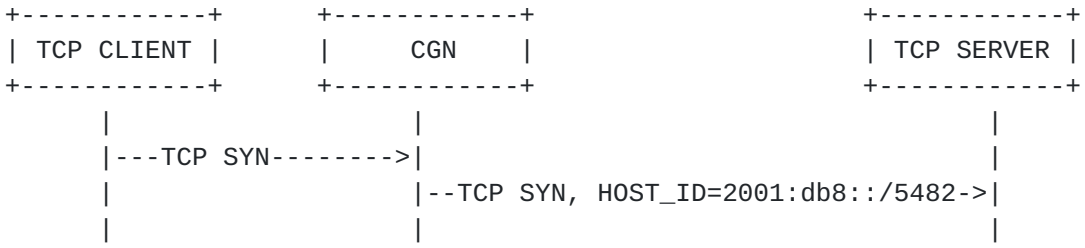

Figure 5: HOST_ID_BOUCADAIR: SYN Mode

3.2.2. ACK Mode

The ACK Mode is as follows (see Figure 6):

- Send HOST_ID_ENABLED (Figure 4) in SYN
- If the remote TCP server supports that option, it must return it in SYNACK
- Then the TCP Client sends an ACK in which the CGN injects HOST_ID_BOUCADAIR (Figure 3)

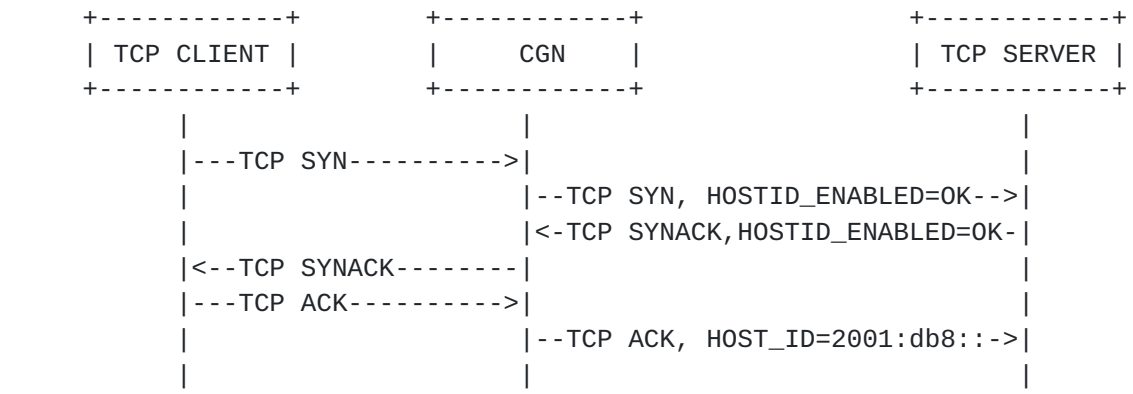

Figure 6: HOST_ID_BOUCADAIR: ACK Mode

4. Overview of the Linux Kernel Modifications

The objective of this phase is to support HOST_ID_WING, HOST_ID_BOUCADAIR and HOST_ID_ENABLED in the SYN mode.

In order to support the injection of the HOST_ID TCP options presented in Section 3, some modifications were applied to the Linux Kernel (more precisely to the TCP stack part of the Kernel). The header file tcp.h, file where are defined the TCP variables and functions, is updated to define the new HOST_ID options' KINDs (option numbers) and Lengths.

Major modifications have been made in the "tcp_output.c" file. This file is responsible for building and transmitting all TCP packets. For each HOST_ID TCP option, the required modifications to increase the header size and to inject KIND, Length and the corresponding HOST_ID data are implemented for the TCP SYN packets.

As we have three different HOST_ID options and as HOST_ID_BOUCADAIR can convey different information the configuration of the HOST_ID options have to be simple with minimal complexity. Since the manipulation of HOST_ID options impacts the Kernel TCP drivers, a suitable solution is to define new sysctl variables (system control variables) that allow the modification of Kernel parameters at

runtime, without having to reboot the machine so that it takes into account a new configuration.

Once modifications have taken place, the Kernel must be recompiled so that the new TCP options are taken into account.

Kernel modifications and recompilation have been done and tested successfully on Fedora and Debian Linux distributions, on different kernel versions.

The following configuration options are supported:

- Enable/Disable injecting the TCP Option
- Support HOST_ID WING, HOST_ID BOUCADAIR and HOST_ID_ENABLED
- When the HOST_ID TCP option is supported, the information to be injected is configurable:
  - Source IPv6 address or the first 56 bits of the address
  - Source IPv4 address
  - Source port number
  - Source IPv4 address and Source port
  - IPv6 address or the first 56 bits of the B4 when DS-Lite is activated

5. Testbed Setup & Configuration

The setup of three testbed configurations have been considered:

1. HOST_ID TCP option is injected by the host itself. No CGN is present in the forwarding path (Figure 7)
2. HOST_ID TCP option is injected by hosts deployed behind a HTTP proxy. No CGN is present in the forwarding path (Figure 8)
3. HOST_ID TCP option is injected by the DS-Lite AFTR element (Figure 9).

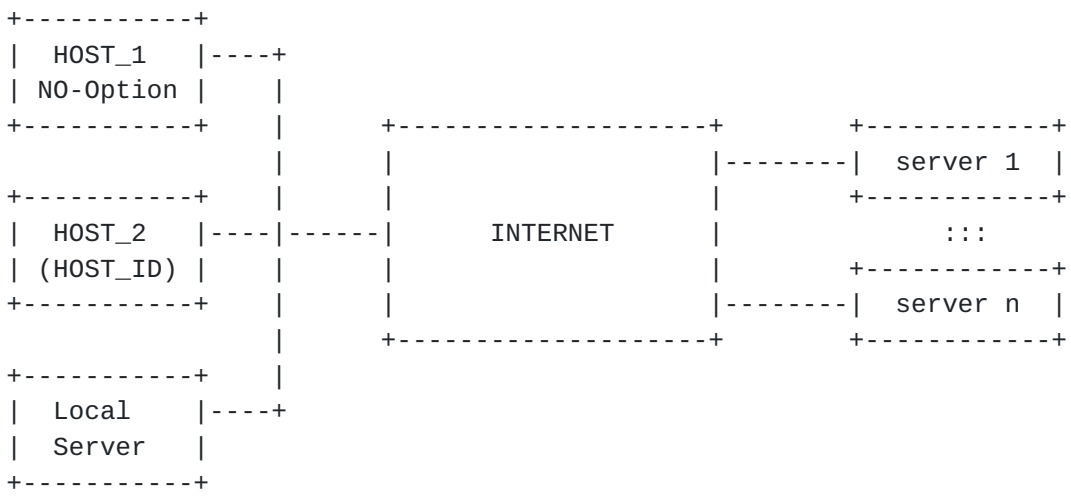

Figure 7: Testbed setup: No Proxy and no CGN

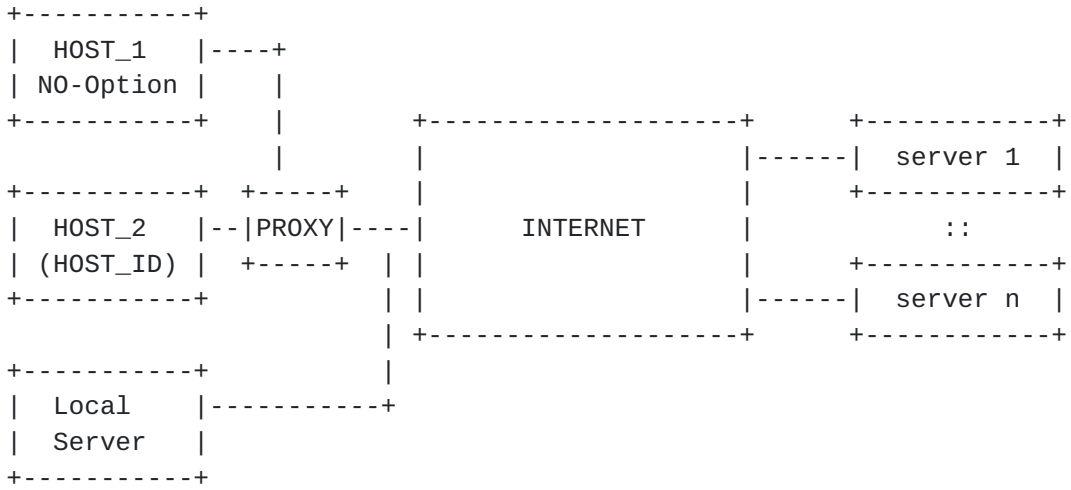

Figure 8: Testbed setup: HTTP Proxy

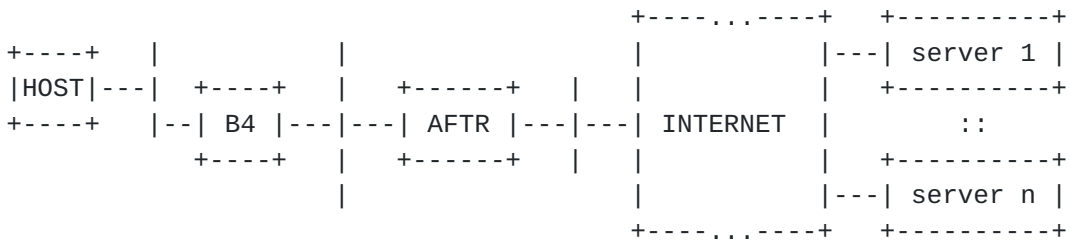

Figure 9: DS-Lite CGN Environment

Figure 7 and Figure 8 are used to assess the behavior of the top 100,000 sites when a HOST_ID option is enabled and to evaluate the impact of the option on both the session establishment delay and the success ratio.

On the other hand, the configuration shown in Figure 9 will be used to evaluate the impact on the CGN performances when HOST_ID TCP option is injected by the CGN.

5.1. Automated TCP Traffic Generator

A Python-encoded robot has been used as the traffic generator. The robot automates the retrieval of HTTP pages identified by URLs, and returns different connection information. The retrieval of pages is based upon Pycurl, a Python interface of libcurl. Libcurl is an URL transfer library that supports different protocols (e.g., HTTP, FTP).

The robot consists of two programs:

1. The first one takes an URL as a input parameter, performs the DNS lookup and then tries to connect to the corresponding machine. It returns either different time values and connection status or an error message with the source of the error in case of connection failure (e.g., DNS error). The TCP connection establishment time is calculated as the difference between the CONNECT_TIME and NAMELOOKUP_TIME where:
   * NAMELOOKUP_TIME is the time it took from the start until the name resolution is completed.
   * CONNECT_TIME is the time it took from the start until the connection to the remote host (or proxy) is completed.
2. The second program aims to increase efficiency and speed of the testing by using a multi-thread technique. It takes the number of threads and an input file listing URLs as parameters. This program prints URLs to an output file with the corresponding connection time. If something wrong happened so that the connection failed, the program returns an error message with the corresponding error type.

5.2. Testing Methodology and Procedure

The testing is done using two machines, one that supports the HOST_ID TCP options and the other that does not. The second machine is used as a reference for the measurements. Testing is performed in parallel on the two machines that are directly connected to the Internet. For each HOST_ID TCP option, the test is repeated many times. The cycle is repeated in different days. Then results are grouped into tables where averages are calculated. The comparison between the different HOST_ID options results is made by using the no-option testing results as a reference.

Testing was also performed behind a proxy (Figure 8) to evaluate the impact of embedding the HOST_ID TCP options on the connection establishment time when a proxy is in the path. When a proxy is

present, the connection delay is impacted (the delay is calculated for the connection between the host and the proxy).

Tests have been conducted from hosts:
1. Connected to an enterprise network
2. In a lab behind a firewall
3. Connected to two (2) commercial ISP networks

5.3. Check HOST_ID TCP Options are Correctly Injected

To check whether the HOST_ID TCP options are correctly injected, the local server in Figure 7 is configured to be reachable from Internet. Packets conveying the HOST_ID TCP options are sent from a host supporting the options. These packets are used without alteration by the local server.

This configuration confirms the packets sent to remote servers conveys HOST_ID TCP options.

5.4. Top Site List

The Alexa top sites list has been used to conduct the HTTP tests.

Anonymous FTP sites list from ftp-sites.org has been used to conduct the FTP tests.

6. Experimentation Results

Various combinations of the HOST_ID TCP options have been tested:
1. HOST_ID_WING
2. HOST_ID_WING has also been adapted to include 32 bits and 64 bits values. No particular impact on session establishment has been observed.
3. HOST_ID_BOUCADAIR (source port)
4. HOST_ID_BOUCADAIR (IPv4 address)
5. HOST_ID_BOUCADAIR (source port:IPv4 address)
6. HOST_ID_BOUCADAIR (IPv6 Prefix)
7. HOST_ID_ENABLED

Both the success ratio and the average time to establish the TCP session are reported below.

6.1. HTTP Experimentation Results

Tests have been conducted from hosts:

1. Connected to an enterprise network
2. Connected to two commercial ISP networks
3. In a lab behind a firewall

6.1.1. Configuration 1: Connected to an enterprise network

The results show that the success ratio for establishing TCP connection with legacy servers is almost the same for all the HOST_ID options as shown in Figure 10, Figure 11 and Figure 12.

6.1.1.1. Results

	NO-OPTION	O-WING	Failure Ratio
Top10	100,00000%	100,00000%	0,00000%
Top100	100,00000%	100,00000%	0,00000%
Top200	100,00000%	100,00000%	0,00000%
Top300	99,66667%	99,66667%	0,00000%
Top400	99,50000%	99,50000%	0,00000%
Top500	99,40000%	99,40000%	0,00000%
Top600	99,50000%	99,50000%	0,00000%
Top700	99,57143%	99,57143%	0,00000%
Top800	99,50000%	99,50000%	0,00000%
Top900	99,44444%	99,44444%	0,00000%
Top1000	99,50000%	99,50000%	0,00000%
Top2000	99,35000%	99,30000%	0,05000%
Top3000	99,10000%	99,06667%	0,03333%
Top4000	99,10000%	99,05000%	0,05000%
Top5000	99,14000%	99,10000%	0,04000%
Top6000	99,21667%	99,18333%	0,03333%
Top7000	99,25714%	99,21429%	0,04286%
Top8000	99,15000%	99,10000%	0,05000%
Top9000	99,16667%	99,12222%	0,04444%
Top10000	99,16000%	99,12000%	0,04000%
Top20000	98,50500%	98,44000%	0,06500%
Top30000	98,21667%	98,11667%	0,10000%
Top40000	98,10750%	98,00750%	0,10000%
Top50000	98,00000%	97,89800%	0,10200%
Top60000	97,95167%	97,85000%	0,10167%
Top70000	97,88857%	97,78857%	0,10000%
Top80000	97,84500%	97,74875%	0,09625%
Top90000	97,79444%	97,69889%	0,09556%
Top100000	97,75100%	97,64800%	0,10300%

Figure 10: Cumulated Success ratio (HOST_ID_WING)

	NO-OPTION	O-WING	Failure Ratio
1-100	100,00%	100,00%	0,00%
101-200	100,00%	100,00%	0,00%
201-300	99,00%	99,00%	0,00%
301-400	99,00%	99,00%	0,00%
401-500	99,00%	99,00%	0,00%
501-600	100,00%	100,00%	0,00%
601-700	100,00%	100,00%	0,00%
701-800	99,00%	99,00%	0,00%
801-900	99,00%	99,00%	0,00%
901-1000	100,00%	100,00%	0,00%
1-1000	99,50%	99,50%	0,00%
1001-2000	99,20%	99,10%	0,10%
2001-3000	98,60%	98,60%	0,00%
3001-4000	99,10%	99,00%	0,10%
4001-5000	99,30%	99,30%	0,00%
5001-6000	99,60%	99,60%	0,00%
6001-7000	99,50%	99,40%	0,10%
7001-8000	98,40%	98,30%	0,10%
8001-9000	99,30%	99,30%	0,00%
9001-10000	99,10%	99,10%	0,00%
10001-20000	97,85%	97,76%	0,90%
20001-30000	97,64%	97,47%	1,70%
30001-40000	97,78%	97,68%	1,00%
40001-50000	97,57%	97,46%	1,10%
50001-60000	97,71%	97,61%	1,00%
60001-70000	97,61%	97,52%	0,90%
70001-80000	97,44%	97,37%	0,70%
80001-90000	97,39%	97,30%	0,90%
90001-100000	97,36%	97,19%	1,70%

Figure 11: TopX000 Success Ratio (HOST_ID_WING)

	NO-OPTION	O-BOUCADAIR	Failure Ratio
1-100	100,00%	100,00%	0,00%
101-200	100,00%	100,00%	0,00%
201-300	99,00%	99,00%	0,00%
301-400	99,00%	99,00%	0,00%
401-500	99,00%	99,00%	0,00%
501-600	100,00%	100,00%	0,00%
601-700	100,00%	100,00%	0,00%
701-800	99,00%	99,00%	0,00%
801-900	99,00%	99,00%	0,00%
901-1000	100,00%	100,00%	0,00%
0-1000	99,50%	99,50%	0,00%
1001-2000	99,20%	99,10%	0,10%
2001-3000	98,60%	98,60%	0,00%
3001-4000	99,30%	99,30%	0,00%
5001-6000	99,60%	99,60%	0,00%
6001-7000	99,50%	99,40%	0,10%
7001-8000	98,40%	98,30%	0,10%
8001-9000	99,30%	99,20%	0,10%
9001-10000	99,10%	99,10%	0,00%
10001-20000	97,85%	97,76%	0,90%
20001-30000	97,64%	97,46%	1,80%
30001-40000	97,78%	97,66%	1,20%
40001-50000	97,57%	97,46%	1,10%
50001-60000	97,71%	97,61%	1,00%
60001-70000	97,61%	97,51%	1,00%
70001-80000	97,44%	97,36%	0,80%
80001-90000	97,39%	97,30%	0,90%
90001-100000	97,36%	97,19%	1,70%

Figure 12: TopX000 Success Ratio (HOST_ID_BOUCADAIR)

6.1.1.2. Analysis

- For the top 100,000 sites, connection failures occur for 2249 HTTP sites. These failures were reported as being caused by DNS issues (servers not mounted), connection timeouts (servers down...), connection resets by peers, connection problems and empty replies from servers. The 2249 failures occur, whether HOST_ID options are injected or not.
- When any HOST_ID TCP option is conveyed, 103 servers did not respond; however when no option is injected, all these servers responded normally.

- Same results were obtained for HOST_ID_WING and HOST_ID_ENABLED.
- Same results were obtained for all the HOST_ID_BOUCADAIR options (source port, IPv6 prefix, etc.).

When HOST_ID_BOUCADAIR is enabled, six (6) additional servers did not respond:

- Three (3) servers (www.teufel.de - www.1001fonts.com - www.sigur-ros.co.uk) did not respond to the SYN packets sent by the host.
- Three (3) servers (www.lawyers.com, www.lexis.com, www.nexis.com) responded with "strange" SYN/ACK packets with same TCP options length including a part of the HOST_ID options that was sent. This part of HOST_ID option caused an erroneous SYN/ACK packet received by the host: in fact the second byte of the HOST_ID part is considered as its length and this length does not really fit with the real length of the part. So the machine does not respond back to the server with an ACK packet. This is why we have no response for these servers.

When HOST_ID_WING or HOST_ID_ENABLED is enabled, also strange SYN/ACKs were received by the host but no errors in these packets (a long series of NOP options). This justifies the connection success for these 2 options.

The results show that including a HOST_ID TCP option does not systematically imply an extra delay for the establishment of the TCP session. Based on the average of session establishment with the top 100 000 sites, the following results have been obtained:

- delay(HOST_ID_WING) < delay(NO_OPTION): 42,55 %
- delay(HOST_ID_BOUCADAIR) < delay(NO_OPTION): 48,16 %
- delay(HOST_ID_ENABLED) < delay(NO_OPTION): 51,28 %

6.1.2. Configuration 2: In a lab behind a firewall

When a HTTP proxy is in the path, the injection of HOST_ID TCP option does not impact the success ratio. This is due to that the HTTP proxy strips the HOST_ID TCP options; these options are not leaked to remote Internet servers. The testing has been done by observing packets received to a server installed with a public IP address (no HOST_ID options were seen in the received SYN packets).

6.1.3. Configuration 3: Connected to two commercial ISP networks

The results obtained when testing was performed by connecting to two ISP networks confirmed the results obtained in the testing described in Section 6.1.1

6.1.4. Additional Results

In one of our testing for top 1000 sites, when padding was badly implemented for HOST_ID_BOUCADAIR (padding was implemented as a prefix so option's Length does not correspond to the real length because the padding was not counted), we got for configuration(1) in the lab and for one of the ISP the following results:

	No-Option	O-BOUCADAIR	Failure Ratio
Top10	100,00000%	100,00000%	0,00000%
Top100	100,00000%	100,00000%	0,00000%
Top200	100,00000%	100,00000%	0,00000%
Top300	100,00000%	99,66667%	0,33333%
Top400	99,75000%	99,00000%	0,75000%
Top500	99,80000%	99,00000%	0,80000%
Top600	99,83333%	98,66667%	1,16667%
Top700	99,85714%	98,14286%	1,71429%
Top800	99,75000%	98,00000%	1,75000%
Top900	99.66667%	97,33333%	2,33333%
Top1000	99,70000%	97,10000%	2,60000%

Cumulated Success ratio (HOST_ID_Boucadair with wrong padding)

The results for HOST_ID_WING for all three configurations are the same as Section 6 (this option was correctly coded). Results obtained for HOST_ID_BOUCADAIR are not the same.

For the configuration (2) behind a firewall, we did not face any rejection because of parsing the TCP options (the HOST_ID options were retrieved from the packet).

6.1.5. Analysis

Configuration (1) in Lab and for one of the two CPEs lead to the results because 2.6% of these 1000 servers perform parsing validation for the received options so when the bad HOST_ID_BOUCADAIR option is sent, 2.6% of the servers treat the received SYN packets as erroneous packets and discard them.

For the connection behind the second ISP, we didn't get a response for any of the servers. After investigation, the reason was that the Box validates the received packets before sending them to the Internet. The erroneous SYN packets holding badly encoded options (HOST_ID_BOUCADAIR in this case) were dropped and no connection was

established. On the other hand, the other box did not validate options length for received packets before sending them to the Internet.

6.2. FTP

Various combinations of the HOST_ID TCP options have been tested:

1. HOST_ID_WING
2. HOST_ID_BOUCADAIR (source port)
3. HOST_ID_BOUCADAIR (source port:IPv4 address)

A list of 5591 FTP servers has been used to conduct these testings. Among this list, only 2045 were reachable:

- Failure to reach 942 FTP servers due to connection timeout
- Failure to reach 1286 FTP servers due to DNS errors
- Failure to reach 717 FTP servers because access was denied
- Could not connect to 500 FTP servers
- Response reading failed for 81 servers
- Bad response from server for 20 servers

When HOST_ID TCP options are injected, 9 errors are observed (connection timeout).

Figure 13 and Figure 14 provide more data about the error distribution.

	NOB	HOST_ID	Failure Ratio
1-100	100%	100%	0,000%
101-200	100%	99%	1,000%
201-300	100%	99%	1,000%
301-400	100%	100%	0,000%
401-500	100%	100%	0,000%
501-600	100%	100%	0,000%
601-700	100%	100%	0,000%
701-800	100%	100%	0,000%
801-900	100%	99%	1,000%
901-1000	100%	99%	1,000%
1001-2000	100%	99,5%	0,500%
2000-2045	100%	100%	0,000%

Figure 13: Cumulated Success Ratio (FTP)

	NOB	HOST_ID	Failure Ratio
first 10	100,000%	100,000%	0,000%
first 100	100,000%	100,000%	0,000%
first 200	100,000%	99,500%	0,500%
first 300	100,000%	99,333%	0,667%
first 400	100,000%	99,500%	0,500%
first 500	100,000%	99,600%	0,400%
first 600	100,000%	99,667%	0,333%
first 700	100,000%	99,714%	0,286%
first 800	100,000%	99,750%	0,250%
first 900	100,000%	99,667%	0,333%
first 1000	100,000%	99,600%	0,400%
first 2000	100,000%	99,550%	0,450%
first 2045	100,000%	99,560%	0,440%

Figure 14: FirstXXX FTP Servers

The results show that including a HOST_ID TCP option does not systematically imply an extra delay for the establishment of the TCP session with remote FTP servers. Based upon the average of the session establishment with the 2045 FTP sites, the following results have been obtained:

- delay(HOST_ID_WING) < delay(NO_OPTION): 49,36585 %
- delay(HOST_ID_BOUCADAIR (source port:IPv4 address)) < delay(NO_OPTION): 48,41076%
- delay(HOST_ID_BOUCADAIR (source port)) < delay(NO_OPTION): 48,43902 %

6.3. SSH

The secure shell service has been tested between a host and a SSH server connected to the same network.

SSH connections have been successfully established with the server for all the HOST_ID TCP options. Same results were obtained using configuration (1) and configuration (2).

6.4. Telnet

Telnet sessions have been successfully initiated for all HOST_ID TCP options with a server (the CGN used in Figure 9).

7. AFTR Module Modifications

This section highlights the support the HOST_ID functionalities in the AFTR element of the DS-Lite model (Figure 9) and presents the testing results in order to conclude about the HOST_ID TCP options impacts on the performance of the CGN.

We used ISC AFTR implementation.

7.1. Specification

All privately-addressed IPv4 packets sent from DS-Lite serviced hosts go through an AFTR device where an isc_aftr daemon program is responsible for establishing the tunnel, configuring network interfaces and processing received packets.

The aftr.c source code controls all functionalities to be included or modified on packets received by the CGN, e.g., patching TCP MSS values, fix MTU, etc.

In order to activate/deactivate such functionalities, the corresponding parameters can be configured in a specific configuration file called "aftr.conf". In this file, other parameters are configured, e.g., the IPv6 addresses assigned to the tunnel endpoint and the global IPv4 address pool maintained by the CGN.

To support the injection of HOST_ID TCP options, "aftr.c" must be updated to inject, retrieve or verify the HOST_ID options depending on the HOST_ID parameters defined in "aftr.conf" file. Four HOST_ID parameters are defined in the configuration file:

1. hostid: to enable the injection, retrieval, matching... of HOST_ID options
2. hostid_wing: to enable injection/verification of HOST_ID_WING - to disable injection or to remove HOST_ID_WING
3. hostid_boucadair: to enable injection/verification of HOST_ID_BOUCADAIR - to disable injection or to remove HOST_ID_BOUCADAIR
4. hostid_enabled: to enable or disable HOST_ID_ENABLED injection

hostid, hostid_wing and hostid_enabled can be simply enabled or disabled. hostid_boucadair can be disabled or enabled with the corresponding Origin as HOST_ID data can be:

- Source Port Number
- Source IPv4 Address
- Source IPv4 Address + Source Port Number

- 56 bits of Tunnel Softwire IPv6 Source Address.

Based on different HOST_ID parameters, the "aftr.c" code has been modified to control HOST_ID options; the AFTR is able to:

- Inject the enabled HOST_ID TCP option if it is not already present in the packet
- Retrieve an existing HOST_ID TCP option if this option is not enabled
- Check an existing HOST_ID option's content if it is enabled; if the content's verification failed, the AFTR replaces the HOST_ID contents with the suitable information

The implementation takes into consideration the SYN mode for all the HOST_ID options (even for HOST_ID_enabled). The Support of HOST_ID_BOUCADAIR in the ACK mode needs implementation on the server's side and since both Enabled and Boucadair's options have been tested and no impact observed; the ACK mode should not imply any complication in implementation or impact on the performance.

7.2. Verification

The verification of HOST_ID implementation in the CGN has taken place using the testbed setup shown in Figure 9. The host used in this testing is a modified Linux machine that can inject HOST_ID options. The objective of the testing is to verify the different functionalities implemented in the AFTR. Verification has occurred using a local server where all the received packets were observed to make sure that the content of the HOST_ID fields is consistent with the enabled option.

The testing consists in observing the SYN packets (as SYN mode is supported) sent by the host and in comparing these packets to those received by the server. Different combinations of HOST_ID options sent by the host and HOST_ID configured options at the CGN level have been used.

The results show that once the host sends packets without any HOST_ID option injected, the SYN packets received by the server contain the correct option that has been enabled by the CGN (if any). Once HOST_ID_WING or HOST_ID_BOUCADAIR are injected by the host, if the hostid parameter in aftr.conf is enabled, the enabled (in "aftr.conf") HOST_ID option will be injected if not already present, or else its content will be verified and corrected (if wrong); the other disabled option will be discarded if it has already been sent by the host.

One additional case has been tested when both Wing's and Boucadair's HOST_ID options are sent by the host, the contents of the enabled

option are checked and corrected (if wrong), the other option is retrieved from the packet. The two options are dropped from the packet if they are both disabled.

The testing has been repeated for all the HOST_ID options sent by the host and enabled by the CGN. Verification also occurred for HOST_ID_ENABLED option.

7.3. CGN Performance Testing

To conclude about the impact of using HOST_ID, a commercial testing product has been used. This tool supports multiple application protocols such as HTTP and FTP for both IPv4 and IPv6 (including encapsulation). The DS-Lite model can be built directly from a port of this product: IPv4 packets are directly encapsulated in an IPv6 tunnel; the client's port emulates hosts and B4 elements at the same time. This port is directly connected to the AFTR tunnel endpoint. The AFTR's IPv4 interface is connected to the testing product server side where servers are assigned IPv4 addresses.

The testbed setup of this testing is shown in Figure 15:

Figure 15: Platform Testbed

7.3.1. Configuration

At the IP level, the testing client port was configured with IPv6 addresses representing the B4. The testing tool also supports the DS-Lite "level" where the number of clients connected to each B4 and their addresses are configured. The AFTR address is defined at this level.

In the current testing, the total number of B4 elements is 5000 behind; One client is connected to each B4 (in total, 5000 clients are configured). However, the number of active users varies from 10 to 100, 500, 1000 and 10,000 during each testing simulation.

From the server standpoint, five servers have been assigned IPv4 addresses. These servers support HTTP and FTP traffic. For each HOST_ID TCP option, the testing was repeated for a different number of active users (N=10, 100, 500, 1000 and 10,000) and for HTTP and FTP traffic.

The HOST_ID options are injected by the CGN.

7.3.2. HTTP Testing

The testing duration was about 50 seconds during which the number of active users varies as a function of time: during the first 10s, the number of active users reaches the maximum and remains the same for the next 20 s. Then it decreases to zero during the next 20s.

Hereafter are provided some testing statistics providing some details about connections' success ratio, latency and other information that can be useful to evaluate the impact of HOST_ID on the CGN.

	No-Opt	O-WING	O-BOUCADAIR3	O-ENABLED
TCP connection established	1378	1267	1363	1369
TCP SYN SENT	1378	1267	1363	1369
Success Ratio	100	100	100	100
TCP Retries	193	193	197	177
TCP timeouts	140	136	152	111
HTTP connect' latencies t=20s	0,11	0,21	0,20	0,1
t=40s	0,40	0,50	0,50	0,45
t=60s	0,60	0,60	0,50	0,6
HTTP throughput received	46,47	45,31	45,88	46,12
TCP Connections Established/s	20,29	19,88	20,06	20,18

Figure 16: Results HTTP (N=10)

	No-Opt	O-WING	O-BOUCADAIR3	O-ENABLED
TCP connection established	1662	1739	1813	1679
TCP SYN SENT	1718	1770	1819	1729
Success Ratio	96	98	99	97
TCP Retries	1577	1569	1783	1576
TCP timeouts	798	806	934	808
HTTP connect' latencies t=20s	1,70	2,00	1,90	1,80
t=30s	3,30	2,40	2,25	3,30
t=40s	4,20	3,70	3,75	4,00
t=50s	5,00	4,80	4,50	5,00
HTTP throughput received	47,56	46,65	48,59	48,06
TCP Connections Established/s	20,94	20,53	21,35	21,19

Figure 17: Results HTTP (N=100)

	No-Opt	O-WING	O-BOUCADAIR3	O-ENABLED
TCP connection established	1956	1923	1944	1873
TCP SYN SENT	2088	2095	2137	1986
Success Ratio	93	91	90	94
TCP Retries	2734	2576	2453	2773
TCP timeouts	1261	1110	995	1213
HTTP connect' latencies t=20s	2,00	1,80	1,50	2,30
t=40s	4,00	3,30	2,80	4,30
t=50s	6,50	6,90	6,00	8,00
HTTP throughput received	70,19	65,00	69,81	67,13
TCP Connections Established/s	30,69	28,41	30,50	29,38

Figure 18: Results HTTP (N=1000)

	No-Opt	O-WING	O-BOUCADAIR4	O-ENABLED
TCP connection established	1576	2000	1796	1998
TCP SYN SENT	2088	2304	2009	2262
Success Ratio	87	86	89	88
TCP Retries	3018	3101	3013	3148
TCP timeouts	1167	1298	1213	1417
HTTP connect' latencies t=20s	2,20	3,00	2,20	2,50
t=40s	3,70	3,00	3,30	3,00
t=60s	7,80	5,00	7,00	5,60
t=70s	9,60	6,00	8,70	7,00
HTTP throughput received	45,00	54,52	51,45	57,20
TCP Connections Established/s	19,98	24,05	22,45	25,04

Figure 19: Results HTTP (N=10000)

7.3.2.1. Analysis of results

The results clearly show that there is no impact of any HOST_ID option on session establishment success ratio, which is quite similar to the success ratio when packets do not hold options or when HOST_ID options are not used. Also, the number of established connections does not decrease when any HOST_ID option is injected, so the CGN performance is not impacted by the fact of adding the HOST_ID options.

Another important factor to study is the latency that can be caused by HOST_ID injection. As the results show, the HTTP connection latency does not increase when HOST_ID is present if we compare the latency measured at different times for the different options.

As a result, we clearly see that the average throughput measured at servers is identical, whether HOST_ID options are used or not (given that the number of session established is quite the same).

Another consequence is that the TCP connection establishment rate at servers is not decreasing when a HOST_ID option is taken into account.

7.3.2.2. Conclusion

The results that have been obtained show that the performance of the CGN is not impacted by HOST_ID option injection even when the number of active users is high (10,000 is not negligible for a CGN run on an ordinary Linux machine): neither the session success ratio, nor the connection latency are impacted by the presence of the HOST_ID in SYN

packets.

7.3.3. FTP

The same testing was also run for FTP traffic. No particular impact on the performance of the CGN has been observed.

8. IPTABLES: Modifications to Enforce Policies at the Server Side

8.1. Overview

iptables module has been updated to:
- Log the content of TCP header with HOST_ID
- Drop packets holding a HOST_ID option
- Match any HOST_ID value
- Drop packets holding a specific HOST_ID value
- Strip any existing HOST_ID option

To support the above functionalities, modification should take into consideration stripping and matching options as described below:

1. To strip the content of any existing HOST_ID option, the shared library "libxt_TCPOPTSTRIP.so" is modified: the HOST_ID_WING and HOST_ID_BOUCADAIR Kinds' numbers were defined in the corresponding source file (libxt_TCPOPTSTRIP.c) with the corresponding names to enforce the iptables stripping rule. After enforcing these changes, the shared library must be created to replace the existing one and to allow applying the rule of stripping of the HOST_ID options. Once modifications have taken place, the following command should be used to strip the HOST_ID options:

```
iptables -t mangle -A INPUT -j TCPOPTSTRIP -p tcp --strip-options
 hostid_wing, hostid_boucadair
```

2. In order to allow blocking, logging or applying any rule based upon the HOST_ID_WING or HOST_ID_BOUCADAIR values or range of values, a HOST_ID shared library must be created to:
   * Match HOST_ID options values entered in corresponding iptables rules,
   * Print the HOST_ID rules on screen,
   * Save values,
   * Check the values (or range values) entered by user if they respect the limit values of these options.

In addition to the shared library: a specific Kernel module must be built to apply HOST_ID matching rules on the packets passing through the network interfaces. This module compares the HOST_ID options' values held by packets with the HOST_ID values specified in the iptables rule table: when a packet matches the HOST_ID's range, the corresponding rule will be applied for this packet.
The HOST_ID_WING matching value is 2 bytes long corresponding to HOST_ID_WING data.
The HOST_ID_BOUCADAIR matching value is 8 bytes long corresponding to Lifetime + Origin field (1 byte) and HOST_ID_WING data (7 bytes).

8.2. Validation

After having updated the iptables package with the suitable HOST_ID libraries and module, different HOST_ID policies should be applied and tested on the server side. The testing has been done using a simple configuration as shown below (Figure 20).

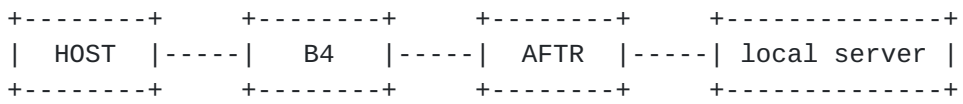

Figure 20: Platform configuration: HOST_ID enforcing policies

In the current testing, the AFTR supports HOST_ID options injection and iptables is modified at the local server. Logging recommendations consists of logging the IPv4 address and the HOST_ID option for each connection. Because HOST_ID is sent only in SYN packets (in the current implementation), only SYN packets will be logged to a specific file called iptables.log: the rsyslog.d must be updated with the corresponding command to log iptables messages into the specific file. Then rsyslog must be reloaded to apply changes.

8.3. Stripping HOST_ID Options

To strip a certain HOST_ID option, TCPOPTSTRIP rule must be called. Verification consists in logging and then checking the SYN packets and more precisely the corresponding TCP options, e.g., the following rules must be applied to strip HOST_ID_WING:

```
iptables -t mangle -A INPUT -j TCPOPTSTRIP -p tcp --strip-options
 hostid_wing
iptables -A INPUT -j LOG --log-tcp-options -p tcp --syn
```

The first rule applies for the mangle table. This table allows

stripping HOST_ID_WING whose role is to remove option Wing's fields and replaces them by NOP options (NOP=No Operation=0x01). The second rule enables the logging of SYN packets with the corresponding TCP options.

After applying these rules (to strip and log HOST_ID_WING) on the local server, we tried to access the server's HTTP pages from the host. The test is repeated several times and a different HOST_ID option is enabled by the AFTR each time.

Then the "iptables.log" file is checked: only one SYN packet is logged with 4 bytes stripped out in the TCP option part. All IPv4 packets going through the AFTR are also logged to be compared with the server's logged stripped packets.

The comparison of the SYN packets logged by the server with the SYN packets sent by the AFTR clearly shows that the stripped option is HOST_ID_WING (all the header fields have been verified to ensure packet matching): the 4 bytes corresponding to the HOST_ID_WING option are replaced with NOP options (each one of the 4 bytes is equal to '1' = NOP).

The same testing was repeated with HOST_ID_BOUCADAIR. The testing shows that the 10 bytes corresponding to this option were successfully stripped.

8.4. Logging a Specific HOST_ID Option Value

The remote server should be able to track connections coming from different clients; it should log packets headers including the HOST_ID TCP option information. This can be enforced using the following command:

```
iptables -t mangle -A INPUT -j TCPOPTSTRIP -p tcp --strip-options
 hostid_wing
```

Now, to log packets matching a certain HOST_ID value or range of values, the following rule must be applied:

```
iptables -A INPUT -p tcp --syn -m hostid --hostid_wing value[:value]
 -j LOG -log-tcp-options
```

This command matches the HOST_ID_WING values held by SYN packets with the specific value [or the specific range of values] determined by

the rule.

The testing configuration in Figure 20 was used. The HOST_ID_WING data are implemented as being the last 16 bits of the IPv4 private source address. When the HOST_ID_WING option is injected by the CGN, if the data field value corresponds to the iptables value (or range of values), the packet header is logged. Otherwise, if the HOST_ID_WING data is said out of range or the packet does not hold the HOST_ID_WING option, the packet is not logged.

The same testing was repeated to match HOST_ID_BOUCADAIR data information:

```
iptables -A INPUT -p tcp --syn -m hostid --hostid_boucadair value
 [:value] -j LOG -log-tcp-options
```

To verify the logging of a specific Boucadair's value, the Boucadair's options holding source IP address (Origin=2) or IPv6 prefix (Origin=4) were tested successfully; these data values are fixed since they depend on the host's address. The two other options that include source port numbers (variable) cannot be tested by value because the port number varies for each connection.

The iptables rules to log HOST_ID_BOUCADAIR range values have been verified successfully for all four HOST_ID_BOUCADAIR options.

8.5. Dropping a specific HOST_ID Option Value

The same testing methodology described in the previous section was repeated to drop packets matching HOST_ID value (or a range of values); e.g. to drop SYN packets matching a particular HOST_ID_WING value:

```
iptables -A INPUT -p tcp --syn -m hostid --hostid_wing value[:value]
 -j DROP
```

In this testing, the HOST_ID_WING option is enabled at the CGN level. After applying the previous rule where Wing's specified value corresponds to the HOST_ID_WING data value (last 16 bits of the host's IPv4 source address), the hosts tries to access HTTP pages of the local server. It sends SYN packets but the server does not respond. Because this packet matches the iptables matching value, the corresponding rule is applied to the SYN packets: a SYN packet is dropped so the host does not receive any packet in return.

When the host is still trying to retrieve pages by sending SYN packets, the command 'iptables -F' will flush all iptables rules. Once applied, this command will let the host retrieve the required pages and the connection is therefore established successfully.

The same testing was repeated for HOST_ID_BOUCADAIR options. SYN packets matching the corresponding rule value or range of values were dropped. Once iptables rules are flushed, connection is established normally.

9. IANA Considerations

This document makes no request of IANA.

10. Security Considerations

Security considerations discussed in [I-D.wing-nat-reveal-option] should be taken into account.

11. Acknowledgments

Many thanks to M. Meulle, P. Ng Tung and L. Valeyre for their help and review. Special thanks to C. Jacquenet for his careful review.

12. References

12.1. Normative References

[I-D.wing-nat-reveal-option]
Yourtchenko, A. and D. Wing, "Revealing hosts sharing an IP address using TCP option", draft-wing-nat-reveal-option-03 (work in progress), December 2011.

[RFC6250] Thaler, D., "Evolution of the IP Model", RFC 6250, May 2011.

12.2. Informative References

[I-D.boucadair-intarea-nat-reveal-analysis]
Boucadair, M., Touch, J., Levis, P., and R. Penno, "Analysis of Solution Candidates to Reveal a Host Identifier in Shared Address Deployments", draft-boucadair-intarea-nat-reveal-analysis-04 (work in

progress), September 2011.

[RFC6269] Ford, M., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, June 2011.

Authors' Addresses

Elie Abdo
France Telecom
Issy-les-Moulineaux

Email: elie.abdo@orange.com

Mohamed Boucadair
France Telecom

Email: mohamed.boucadair@orange.com

Jaqueline Queiroz
France Telecom
Issy-les-Moulineaux

Email: jaqueline.queiroz@orange.com