# Prototyping TCP Options to Reveal Host Identity in IP Address Sharing Environments

Elie Abdo
France Telecom,
38, rue du General Leclerc
Issy Les Moulineaux, France
e-mails: elie.abdo@orange.com

Mohamed Boucadair
France Telecom
3, rue Clos Courtel
Rennes, France
mohamed.boucadair@orange.com

Jaqueline Queiroz
France Telecom
38, rue du General Leclerc
Issy Les Moulineaux, France
jaqueline.queiroz@orange.com

***Abstract*—Internet Service Providers must maintain the delivery of IPv4 services during the forthcoming IPv6 transition period. For this purpose, Service Providers are likely to deploy address sharing mechanisms. However, address sharing techniques raise specific issues such as the difficulty to distinguish unambiguously different hosts sharing the same public IPv4 address. To mitigate some of the encountered issues, HOST_ID TCP Option has been proposed as a means to reveal the identity of a host when address sharing is deployed by Internet service providers. If no HOST_ID is revealed to remote servers, all subscribers sharing the same IP address will be impacted by a misbehaving user. This paper documents implementation and testing results of HOST_ID TCP Option. Linux kernel and Carrier Grade NAT have been ported to support the ability to inject HOST_ID Options while iptables module has been modified to interpret the information conveyed in HOST_ID and also to enforce dedicated policies.**

***Keywords- address sharing; HOST_ID; TCP Option.***

## I. INTRODUCTION

The explosion of the Internet in the past few years has accelerated the exhaustion of IPv4 global addresses. While only IPv6 deployment can solve IPv4 address shortage, service providers are required to maintain their IPv4 service offerings using the remaining global IPv4 addresses. To do so, large scale address sharing techniques should be implemented to serve a large number of subscribers with a limited IPv4 address space. However, when different hosts are sharing the same IPv4 address, several issues are likely to be encountered [5]. These issues impact subscribers, service providers and content providers: e.g. many services will fail to work, legitimate users will share the reputation of misbehaving users or 'spammers', etc. A use case example would be, when a user is misbehaving, the shared IPv4 address will be reported on a blacklist by the content provider; the access could be then denied for all subscribers sharing that IP address. More issues encountered with IPv4 sharing techniques are detailed in [5].

To mitigate some of these issues, [2] identifies a list of solutions aiming to reveal extra information that must be unique for each host sharing the same IPv4 address: this information is called HOST_ID.

If HOST_ID is revealed to remote servers, hosts are not identified by the sole use of IPv4 address but the identification will be based on the combination of the external IPv4 address and the HOST_ID information. To make such distinction possible, the HOST_ID must be unique to each user who shares the same global IPv4 address (no need to be globally unique). This information can be an IPv6 prefix address, the private source IPv4 address, etc.

The HOST_ID can be injected by the address sharing function (e.g., CGN (Carrier Grade NAT)) which is transparent to the host. Another alternative to reduce potential CGN performance degradation is to let the Customer Premises Equipment (CPE) or the host injecting the HOST_ID information; the CGN only verifies the content of the Option.

The HOST_ID can be leaked in multiple levels of an IP packet. The IP Identification (IP-ID) field of IP header may be used to hold HOST_ID but this will require a dedicated channel to inform servers whether this header is conveying HOST_ID or not. HOST_ID can be put at IP level as a new IP Option (e.g., [13]); however this alternative is unlikely because IP options are not processed by intermediate routers [4]. [3] defines HOST_ID solution as being a new TCP Option suitable for all TCP-based applications. Other proposals such as Proxy Protocol [12] and HIP (Host Identity Protocol [9]) require modifications at both servers and CGN; otherwise, connection could not be established. Another HOST_ID proposal consists of sending the HOST_ID information at the application level (e.g., HTTP header (XFF or Forwarded-For [10]); this proposal solves the issue for HTTP traffic only.

Defining HOST_ID as a TCP Option is superior to XFF. This paper focuses on this alternative.

This paper defines an extended HOST_ID TCP Option and provides experimentation results of this TCP Option. Linux Kernel, CGN and iptables modules have been ported to support the HOST_ID TCP Option. Appropriate validation effort has been conducted to achieve the following objectives:

 ISBN: 978-1-61208-204-2 

- 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 behaviour 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. DS-Lite CGN is used (see Section III)

The remainder of this paper is organized as follows. An overview of the HOST_ID TCP Option is described in Section II. Then, Section III sketches at a glance an overview of DS-Lite technique. Section IV highlights the required Linux Kernel modifications to support the HOST_ID TCP Option. Section V describes the testing conducted to evaluate the behavior of legacy TCP servers and connection delays when servers receive HOST_ID Options. Section VI presents the modifications of the CGN to inject HOST_ID TCP Option. Finally, Section VII illustrates the policies to be enforced at servers' side to make use of the HOST_ID and therefore, to mitigate identification issues introduced by address sharing mechanisms.

## II. FOCUS ON HOST_ID TCP OPTION

The initial idea of defining a TCP Option to convey a HOST_ID was defined in [14]. Nevertheless, the format of that Option does not allow covering various use cases (such as the load-balancer use case). A new TCP 10-byte Option is proposed to meet this requirement (Figure 1). This Option offers similar features than "Forwarded-For" HTTP header [10].

- KIND number
- Lifetime (4 bits) indicates the validity lifetime of the enclosed data, the following values are supported:
  - 0: Permanent
  - >0: Dynamic; this value indicates the validity time.
- Origin (4 bits) 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 and Internal IPv4 address
  - "3": IPv6 Prefix
  - ">3": No particular semantic

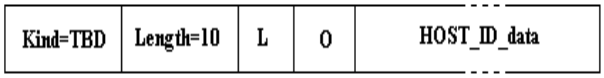


Figure 1: Format of HOST_ID TCP Option

- HOST_ID_data (7 bytes) depends on the Origin field; padding is then required as data of different length can be added.

Two modes of sending HOST_ID are supported: (1) The SYN mode in which the HOST_ID TCP Option is sent in SYN packets and (2) the ACK mode which requires to define a new 2-byte long TCP Option called HOST_ID_ENABLED and which is characterized as follows: The address sharing function injects the HOST_ID_ENABLED TCP Option in a SYN packet. If the remote server supports the HOST_ID Option, it must return the HOST_ID_ENABLED in the SYNACK packet. Then, the TCP client sends an ACK including the HOST_ID TCP Option.

## III. DS-LITE AT A GLANCE

DS-Lite [3] address sharing technique is enabled in the validation platform to conducted testing on CGN (Figure 2). The DS-Lite model is composed of two components: (1) DS-Lite CPE (Customer Premises Equipment) with a B4 (Basic Bridging BroadBand) element and (2) one or several AFTR (Address Family Transition Router) elements, deployed in the network. The DS-Lite combines two techniques: IPv4-in-IPv6 tunnel encapsulation/de-capsulation that is performed at the B4 and the AFTR elements and the NAPT function [11] implemented at the AFTR (i.e., CGN).

## IV. LINUX KERNEL MODIFICATIONS

The objective of Linux Kernel modifications is to support the HOST_ID Option in the SYN mode and then conduct appropriate testing to assess the behavior of top 100,000 legacy HTTP servers, a list of FTP servers, Telnet and SSH services when the HOST_ID TCP Option is conveyed to the servers. The Kernel modified machine will be used afterwards when the HOST_ID Options injection is performed by the host; the address sharing function (see Section VI) only verifies the Options' content validity. This implementation has the advantage to avoid overloading the CGN.

TCP stack of the Linux Kernel has been modified to support HOST_ID TCP Option. Subsequently, recompiling the machine allows the machine to inject the HOST_ID Options and then drive the testing. Through these modifications, we can inject the HOST_ID TCP Options in all SYN packets.

To configure the different HOST_ID Data forms, we defined new Kernel sysctl (system control) variables as HOST_ID injection impacts Kernel TCP driver which allows changing the configuration without rebooting the machine under test. Kernel modifications and recompilation have been made using Fedora and Debian Linux distributions, on

 ISBN: 978-1-61208-204-2 

different Kernel versions. The following configurations options have been implemented:

- Enable/Disable injecting the TCP Options
- Support HOST_ID and HOST_ID_ENABLED
- Data form is configurable and can inject: Source IPv6 address or the first 56 bits of the IPv6 address, Source IPv4 address, Source port number, Source IPv4 address and Source port number.

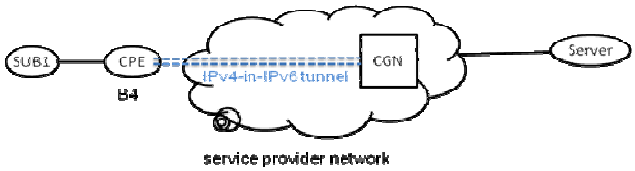


Figure 2: The DS-Lite Architecture

## V. EXPERIMENTATION AND RESULTS

The testbed setup is shown in Figure 3. Two hosts are directly connected to the Internet: *Host1* is a machine which does not support HOST_ID while *Host2* is a modified machine (i.e., patched with the updated Kernel described in Section IV). We run the testing on both machines in parallel for all the HOST_ID TCP Options. The results obtained for *Host1* are used as reference for measurements. In this testing, we first connected the hosts to an enterprise network and then to two ISPs networks to make sure that HOST_ID Options are not stripped. For this purpose, we configured a local server with a public IPv4 address to make it reachable from the Internet.

This configuration is then used to assess the behavior of the top 100,000 websites when a HOST_ID Option is enabled. Also FTP, Telnet and SSH services have been tested.

We coded a Python robot as the traffic generator. The robot automates the retrieval of objects identified by URLs, and returns different connection information (different timing measures). The retrieval of pages is based upon Pycurl, a Python interface of libcurl. The robot consists of two programs. The first one takes an URL as an input parameter, performs the DNS lookup and then tries to connect to the corresponding machine and retrieves the objects identified by the URL. 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).

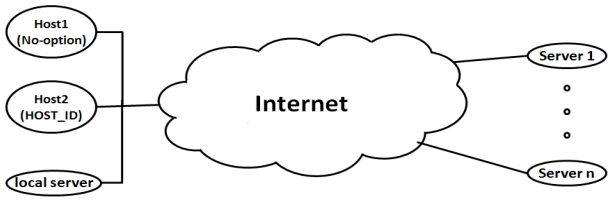


Figure 3: Machines directly connected to Internet

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 name resolution is completed and CONNECT_TIME is the time it took from the start until the connection to the remote host (or proxy) is completed. The second program prints URLs to an output file with the corresponding connection time. If connection could not be established, the program returns an error message with the corresponding error type.

We performed the testing in parallel on the two machines (Figure 3) for all the HOST_ID TCP Options. We repeated the cycle several times for each Option in different days. Then, we calculated TCP sessions establishment delays as average of testing repetitions. Also we computed sessions' success ratio and compared the results using the no-Option testing results (used as reference). The local server, shown in Figure 3, is used to verify HOST_ID TCP Options are correctly injected.

We considered various combinations of Data revealed in the HOST_ID TCP Options (see Section II): source port, IPv4 address, source port: IPv4 address, 56 bits of IPv6 Prefix and HOST_ID_ENABLED.

SSH and Telnet sessions have been successfully initiated for all HOST_ID TCP Options with the local server.

Below are reported both the success ratio and the average time to establish the TCP session a connection for HTTP and FTP services.

### A. HTTP

The same results were obtained for hosts connected to an enterprise network and to networks of two ISPs. These results are synthesized in Tables 1.

TABLE I. HTTP RESULTS – CUMULATED SUCCESS RATIO

| | No-Option | HOST_ID | Failures | Failure Ratio |
|---|---|---|---|---|
| 1-1000 | 995 | 995 | 0 | 0.000% |
| 1001-2000 | 992 | 991 | 1 | 0.101% |
| 2001-3000 | 986 | 986 | 0 | 0.000% |
| 3001-4000 | 991 | 990 | 1 | 0.101% |
| 4001-5000 | 993 | 993 | 0 | 0.000% |
| 5001-6000 | 996 | 996 | 0 | 0.000% |
| 6001-7000 | 995 | 994 | 1 | 0.101% |
| 7001-8000 | 984 | 983 | 1 | 0.102% |
| 8001-9000 | 993 | 992 | 1 | 0.101% |
| 9001-10000 | 991 | 991 | 0 | 0.000% |
| 10001-20000 | 9785 | 9776 | 9 | 0.092% |
| 20001-30000 | 9764 | 9746 | 18 | 0.184% |
| 30001-40000 | 9778 | 9766 | 12 | 0.123% |
| 40001-50000 | 9757 | 9746 | 11 | 0.113% |
| 50001-60000 | 9771 | 9761 | 10 | 0.102% |
| 60001-70000 | 9761 | 9751 | 10 | 0.102% |
| 70001-80000 | 9744 | 9736 | 8 | 0.082% |
| 80001-90000 | 9739 | 9730 | 9 | 0.092% |
| 90001-100000 | 9736 | 9719 | 17 | 0.175% |
| 1-100000 | 97751 | 97642 | 109 | 0.112% |

For the top 100,000 websites [15], connection failures occurred for 2249 HTTP sites. These failures were reported

 ISBN: 978-1-61208-204-2 

as being caused by DNS issues, connection timeouts (e.g., 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

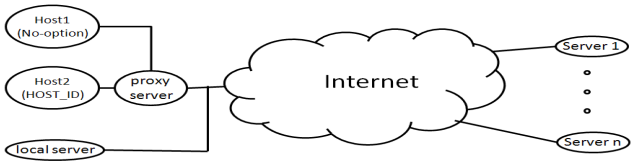


Figure 4: Proxy Server

The same results were obtained for HOST_ID and HOST_ID_ENABLED. The connection failures' ratio for HOST_ID_ENABLED is 0,105% while it is 0.112% for the HOST_ID Option in comparison with total established connections (when no HOST_ID Option is present). These results were obtained for all the HOST_ID TCP Options (source port, IPv6 prefix, etc.). When any HOST_ID TCP Option is conveyed, 103 servers did not respond; however when no Option is injected, all these servers responded normally. For six additional servers which did not respond: Three servers did not respond to the SYN packets sent by the host and three servers responded with malformed and erroneous SYN/ACK packets so connection is dropped by host when receiving the SYN/ACK packet. When HOST_ID_ENABLED is enabled, malformed SYN/ACKs were received by the host too, but these packets were error-free (a long series of NOP Options). This justifies the connection success for these two 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.

When an HTTP proxy is in the path (Figure 4), it strips the HOST_ID TCP Options. The testing has been conducted by verifying packets' content received by the local server: no HOST_ID Options were present in the received SYN packets at the server despite being sent by the host.

### *B. FTP*

Two combinations of the HOST_ID TCP Option have been tested: (1) HOST_ID (source port) and (3) HOST_ID (source port: IPv4 address).

A list of 5591 FTP servers [16] has been used to conduct these tests. Among this list, only 2045 were reachable: failure to reach 942 servers due to connection timeout, failure to reach 1286 servers due to DNS errors, failure to reach 717 servers because access was denied, connection error with 500 servers, failure to reading response from 81 servers and bad response from 20 servers. When HOST_ID TCP Options are injected, 9 FTP servers did not respond to the SYN packets sent by the host. The connection failure distribution is presented in Table 2.

The results show that the sending a HOST_ID TCP Option does not systematically imply an average extra delay for the establishment of the TCP sessions with remote FTP servers. Based upon the average of the session establishment time with the 2045 FTP sites, no extra delay is observed when the HOST_ID TCP Option is injected.

TABLE II. FTP RESULTS – CUMULATED SUCCESS RATIO

| | No-Option | HOST_ID | Failures | Failure Ratio |
|---|---|---|---|---|
| 1-100 | 100 | 100 | 0 | 0,00% |
| 101-200 | 100 | 99 | 1 | 1,00% |
| 201-300 | 100 | 99 | 1 | 1,00% |
| 301-400 | 100 | 100 | 0 | 0,00% |
| 401-500 | 100 | 100 | 0 | 0,00% |
| 501-600 | 100 | 100 | 0 | 0,00% |
| 601-700 | 100 | 100 | 0 | 0,00% |
| 701-800 | 100 | 100 | 0 | 0,00% |
| 801-900 | 100 | 99 | 1 | 1,00% |
| 901-1000 | 100 | 99 | 1 | 1,00% |
| 1001-2000 | 1000 | 995 | 5 | 0,50% |
| 2000-2045 | 45 | 45 | 0 | 0,00% |
| Total | 2045 | 2036 | 9 | 0,44% |

## VI. ISC AFTR MODULE MODIFICATIONS

This section presents the modifications to support the HOST_ID functionalities by the ISC-AFTR module [7].

All privately-addressed IPv4 packets sent from DS-Lite serviced hosts are sent to an AFTR device where an isc_aftr daemon program is responsible for processing received packets. The NAPT function is performed by the AFTR. To activate/de-activate ISC-AFTR functionalities, e.g., patching TCP MSS values, fix MTU, etc, the corresponding variables must be configured in the 'aftr.conf' configuration file. We modified the ISC-AFTR code in order to support the following functionalities: (1) Inject the HOST_ID TCP Options, (2) Retrieve an existing HOST_ID TCP Option in case this Option is not configured and (3) Check the validity of the integrity of the contents of HOST_ID TCP Option in case the corresponding Option is already present in the SYN packet and at the same time the Option is enabled at the AFTR. We modified the "aftr.c" source code to support the HOST_ID Options functionalities (described above) depending on the configuration variables in "aftr.conf". Modified ISC-AFTR can be configured to inject HOST_ID TCP Option conveying: Source Port Number, Source IPv4 Address, Source IPv4 Address + Source Port Number, 56 bits of the IPv6 Source Address used by the AFTR to identify a tunnel endpoint.

The setup shown in Figure 5 is used to validate the implemented modifications in the ISC-AFTR module. We used the local server in our testing to check the contents of HOST_ID Options held in SYN packets. We also investigated the SYN packets sent by the host. Thereby, we compared the content of the packets sent by the host and those received by the server to judge if the functions implemented at the AFTR are applied properly. All possible combinations of HOST_ID Options sent by the host and HOST_ID Options configured at the AFTR. The AFTR can inject several Options, strip existing Options, check the validity of received Options. The host is a machine supporting HOST_ID TCP Option.

 ISBN: 978-1-61208-204-2 

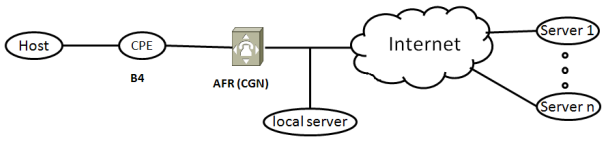


Figure 5: Testbed Setup – DS-Lite CGN environment

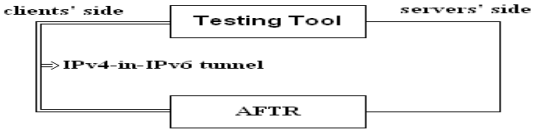


Figure 6: Platform Testbed – AFTR Performance

To conclude about the performance impact of enabling to inject HOST_ID on the CGN, we used a commercial testing product. 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 6.

The testing client's port is 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 this test group, 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 5,000 during each testing simulation. We configured five servers with IPv4 addresses. These servers support HTTP and FTP traffic. For each HOST_ID TCP Option, we repeated the testing for a different number of active users (N=10, 100, 500, 1000 and 5,000) and for HTTP and FTP traffic. The HOST_ID Options are injected by the CGN.

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

Tables 6 and 7 show some testing statistics showing details about connections' success ratio, latency and other information that can be useful to evaluate the impact of HOST_ID on the CGN (ISC-AFTR). The results clearly show that there is no impact of HOST_ID Options 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 (ISC-AFTR) performance is not impacted by the fact of adding the HOST_ID Options. 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.

## VII. ENFORCE POLICIES AT THE SERVER SIDE

Internet-facing servers should be able to manipulate the HOST_ID information. For illustrating purpose, we modified *iptables* module to enforce policies based on the content of the HOST_ID. The modification of the *iptables* module aims to: strip any existing HOST_ID Option, match any HOST_ID value, log the content of TCP headers including the HOST_ID information, print the HOST_ID rules on screen, drop packets holding a HOST_ID Option and drop packets holding a specific HOST_ID value.

TABLE III. HTTP RESULTS (N=100)

| | | No Option | HOST_ID | O-Enabled |
|---|---|---|---|---|
| **TCP connection established** | | 1662 | 1813 | 1679 |
| **TCP SYN sent** | | 1718 | 1819 | 1726 |
| **Success Ratio** | | 96 | 99 | 97 |
| **TCP Retries** | | 1577 | 1783 | 1576 |
| **TCP timeouts** | | 798 | 934 | 808 |
| **Latency** | **t=20s** | 1,7 | 1,9 | 1,8 |
| | **t=30s** | 3,3 | 2,25 | 3,3 |
| | **t=50s** | 5 | 4,5 | 5 |
| **HTTP throughput** | | 47,56 | 48,59 | 48,06 |
| **TCP connections Established/s** | | 20,94 | 21,35 | 21,19 |

TABLE IV. HTTP RESULTS (N=5,000)

| | | No Option | HOST_ID | O-Enabled |
|---|---|---|---|---|
| **TCP connection established** | | 1576 | 1796 | 1998 |
| **TCP SYN sent** | | 1794 | 2009 | 2262 |
| **Success Ratio** | | 87 | 89 | 88 |
| **TCP Retries** | | 3018 | 3013 | 3149 |
| **TCP timeouts** | | 1167 | 1213 | 1417 |
| **Latency** | **t=20s** | 2,2 | 2,2 | 2,5 |
| | **t=40s** | 3,7 | 3,3 | 3 |
| | **t=60s** | 7,8 | 7 | 5,6 |
| **HTTP throughput** | | 45 | 51,45 | 57,2 |
| **TCP connections Established/s** | | 19,8 | 22,45 | 25,05 |

We built a specific Kernel module to apply HOST_ID matching rules on the packets passing through the network interfaces. This module compares the HOST_ID Options' values hold 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. After updating the iptables package with the required HOST_ID libraries, we enforced and tested different HOST_ID policies at the server side. Testbed

 ISBN: 978-1-61208-204-2 

configuration shown in Figure 5 is used for the testing. The AFTR supports injecting HOST_ID Options and iptables modules have been patched at the local server. Logging is performed only for received SYN packets. A specific file is generated for that purpose.

To strip a given HOST_ID Option, TCPOPTSTRIP rule must be applied. The verification consists in logging and then checking the headers of the SYN packets, precisely the TCP Options: *e.g.*, the following rules must be enforced to strip HOST_ID from a received SYN packet:

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

The first rule applies for the mangle table and allows stripping HOST_ID whose role is to remove Option 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 (i.e., to strip and log HOST_ID) on the local server, we tried to access the local server's pages from the host. We repeated the testing 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 compare 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. 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 is implemented owing to a simple command:

```
iptables -A INPUT -j LOG --log-tcp-options -p tcp --syn
```

To log packets matching a given HOST_ID value or range of values, the following rule must be enforced:

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

This command matches the HOST_ID values conveyed in SYN packets with the specific value [or the specific range of values] determined by the configured rule. The value to match for HOST_ID is the content of HOST_ID_Data.

When the HOST_ID 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 data is out of range or the packet does not hold the HOST_ID Option, the packet is not logged. To drop packets matching HOST_ID value (or a range of values), the following command must be executed:

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

The HOST_ID Option is enabled at the CGN level. After applying the previous rule, hosts try to access HTTP content of the local server. A host 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. While the host is still trying to retrieve pages by sending SYN packets, the command '*iptables –F*' will flush all *iptables* rules. Once applied, the host establishes successfully a TCP session with the server.

## VIII. CONCLUSION AND NEXT STEPS

Both implementations of HOST_ID Option at the Linux Kernel TCP stack and the CGN demonstrated that HOST_ID support is feasible and not complex. Testing, conducted using different testbed configurations, has led to: no impact is induced by injecting HOST_ID TCP Options on TCP session establishment delay, only few HTTP servers did not respond when HOST_ID Option was present. The success ratio is not significantly impacted, FTP session success ratio is slightly impacted by the presence of HOST_ID Options (0.44% of connection failures have been observed for 2045 servers), no impact of HOST_ID Options on the performance of the CGN (ISC-AFTR), SSH and Telnet sessions were established successfully, filtering and logging the incoming connections based upon the content of HOST_ID Option information were applied and tested successfully. Further work will focus on security implications of revealing a host identifier.

## REFERENCES


[1] M. Bagnulo, P. Matthews and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.

[2] M. Boucadair, J. Touch, P. Levis and R. Penno, "Analysis of Solution Candidates to Reveal a Host Identifier in Shared Address Deployments", draft-ietf-intarea-nat-reveal-analysis, February 2012.

[3] A. Durand, R. Droms, J. Woodyatt and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.

[4] R. Fonseca, G. Porter, R. Katz, S. Shenker and I. Stoica, "IP options are not an option", UCB/EECS- 2005-24, 2005.

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

[6] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley and H. Tokuda,, "Is it still possible to extend TCP?", November 2011, http://nrg.cs.ucl.ac.uk/mjh/tmp/mboxes.pdf.

[7] ISC AFTR, http://www.isc.org/software/aftr [retrieved: June, 2012].

[8] A. Medina, M. Allman and S. Floyd, "Measuring the Evolution of Transport Protocols in the Internet", ACM CCR, 35(2):37–52, 2005.

[9] R. Moskowitz, P. Nikander, P. Jokela and T. Henderson, "Host Identity Protocol", RFC 5201, April 2008.


 ISBN: 978-1-61208-204-2 

[10] A. Petersson and M. Nilsson, “Forwarded HTTP Extension”, draft-ietf-appsawg-forwarded-for, January 2012.
[11] P. Srisuresh and K. Egevang, “Traditional IP Network Address Translator”, RFC 3022, January 2001.
[12] W. Tarreau, “The Proxy protocol”, November 2010, http://haproxy.1wt.eu/download/1.5/doc/proxy-protocol.txt [retrieved: June, 2012].
[13] Y. Wu, H. Ji, Q. Chen and T. Zou, “IPv4 Header Option For User Identification In CGN Scenario”, draft-chen-intarea-v4-uid-header-option, March 2011.
[14] A. Yourtchenko and D. Wing, “Revealing hosts sharing an IP address using TCP option”, draft-wing-nat-reveal-option, December 2011.
[15] Alexa, http://www.alexa.com/topsites [retrieved: June, 2012]
[16] FTP sites, ftp-sites.org [retrieved: June, 2012]

 ISBN: 978-1-61208-204-2 