

# TNT, Watch me Explode: A Light in the Dark for Revealing MPLS Tunnels

Yves Vanaubel\*, Jean-Romain Luttringer†, Pascal Mérindol‡, Jean-Jacques Pansiot‡, Benoit Donnet\*

\* Montefiore Institute, Université de Liège – Belgium

† Icube, Université de Strasbourg – France

**Abstract**—Internet topology discovery has been a recurrent research topic for nearly 20 years now. Usually, it works by sending hop-limited probes (i.e., traceroute) towards a set of destinations to collect topological data in order to infer the Internet topology at a given scale (e.g., at the router or the AS level). However, traceroute comes with multiple limitations, in particular with layer-2 clouds such as MPLS that might hide their content to traceroute exploration. Thus, the resulting Internet topology data and models are incomplete and inaccurate.

In this paper, we introduce TNT (Trace the Naughty Tunnels), an extension to Paris traceroute for revealing most (if not all) MPLS tunnels along a path. TNT works in two basic stages. First, along with traceroute probes, it looks for evidences of the potential presence of hidden tunnels. Those evidences are surprising patterns in the traceroute output, e.g., abrupt and significant TTL shifts. Second, if alarms are triggered due to the presence of such evidences, TNT launches additional and dedicated probing for possibly revealing the content of the hidden tunnel. We validate TNT through emulation with GNS3 and tune its parameters through a dedicated measurement campaign. We also largely deploy TNT on the Archipelago platform and provide a quantification of tunnels, updating so the state of the art vision of MPLS tunnels. Finally, TNT and its validation platform are fully and publicly available, as well as the collected data and scripts used for processing data.

## I. INTRODUCTION

For now twenty years, the Internet topology discovery has attracted a lot of attention from the research community [1], [2]. First, numerous tools have been proposed to better capture the Internet at the IP interface level (mainly based on traceroute) and at the router level (by aggregating IP interfaces of a router through *alias resolution*). Second, the data collected has been used to model the Internet [3], but also to have a better knowledge of the network ecosystem and how it is organized by operators.

However, despite the work done so far, a lot of issues still need to be fixed, especially in data collection processes based on traceroute. For instance, collecting data about Layer-2 devices connecting routers is still an open question, although it has been addressed previously with a, nowadays, deprecated tool (i.e., IGMP-based probing) [4]. Another example is the relationship between traditional network hardware and the so-called middleboxes [5], [6]. Finally, MPLS tunnels [7] also have an impact on topology discovery as they allow to hide internal hops [8], [9].

This paper focuses on the interaction between traceroute and MPLS. In a nutshell, MPLS has been designed to reduce the time required to make forwarding

decisions thanks to the insertion of *labels* (called *Label Stack Entries*, or LSE) before the IP header.<sup>1</sup> In an MPLS network, packets are forwarded using an exact match lookup of a 20-bit value found in the LSE. At each MPLS hop, the label of the incoming packet is replaced by a corresponding outgoing label found in an MPLS switching table. The MPLS forwarding engine is lighter than the IP forwarding engine because finding an exact match for a label is simpler than finding the longest matching prefix for an IP address. Some MPLS tunnels may be revealed to traceroute because MPLS routers are able to generate ICMP time-exceeded messages when the MPLS TTL expires and the ICMP message embeds the LSE, revealing so the presence of the tunnel [10], [8]. However, MPLS supports optional features that make tunnels more or less invisible to traceroute. Such features modify the way routers process the IP and MPLS TTL of a packet. By carefully analyzing some MPLS related patterns like TTLs (e.g., the quoted forward TTL, the returned TTL of both error and standard replies), one can identify and possibly discover the L3-hops hidden within an MPLS cloud. A first attempt has been already proposed for revealing so-called Invisible tunnels [9]. Here we are going several steps further by providing new revelation techniques (in particular for dealing with the ultimate hop popping feature), and its validation with multiple MPLS and BGP configurations (by emulating MPLS network through GNS3).

This paper aims at plugging the gaps in identifying and revealing the content of MPLS tunnels. This is done by introducing TNT (Trace the Naughty Tunnels), an open-source extension for Paris traceroute [11] including techniques for inferring and revealing MPLS tunnels content. Compared to our previous work [8], [9], this paper provides multiple contributions:

- 1) we strongly **revise the MPLS tunnel classification** as proposed by Donnet et al. [8]. In particular, when possible, we subdivide the “Invisible Tunnel” class in two more accurate categories, “Invisible PHP” and “Invisible UHP”. We show that actually those tunnels can be systematically revealed when they are built with basic P2P LDP [12] or RSVP-TE [13] circuits (and can be at least detected if constructed with more complex technologies such as P2MP VPRN [14]). We also explain why most “Opaque” tunnels content cannot be revealed in practice.

<sup>1</sup>Although MPLS can also be used with IPv6, we consider only IPv4 in this paper.

Finally, we refine and even correct previous quantification of each tunnel class with large-scale measurements performed in the wild;

- 2) we complement the state of the art with `traceroute`-based **measurement techniques** able to reveal most (or at least detect all) MPLS tunnels, even those that were built for hiding their content. While our previous work [9] required to target suspect and pre-analyzed zones in the Internet (i.e., considering high degree nodes and their neighbors visible in the ITDK dataset [15]), we provide here measurement techniques fully integrated in `traceroute`. We associate with each category of the classification *indicators* or *triggers* that are used to determine, on the fly, the potential presence of a tunnel and possibly its nature. In particular, in this paper, we are able to identify the presence of the newly introduced “UHP Invisible” tunnel class thanks to the duplication of an IP address in the `traceroute` output. When a trigger is pulled during a `traceroute` exploration, an MPLS *revelation* [9] is launched with the objective of revealing the tunnel content. We validate the indicators, triggers, and revelations using GNS-3, an emulator running the actual IOS of real routers in a virtualized environment<sup>2</sup>, on a large set of realistic configurations. We also show, through measurements, that our techniques are efficient in terms of cost (i.e., the additional amount of probes injected is reasonable, specially compared to the quality of new data discovered) and errors (false positives and false negatives);
- 3) we **implement** those techniques within Scamper [16], the state of the art network measurements toolbox as a Paris `traceroute` extension, called `TNT`, and deploy it on the Archipelago infrastructure [17]. `TNT` aims at replacing the old version of Scamper and is, thus, subject to run every day towards millions of destinations. As such, we believe `TNT` will be useful to study MPLS deployment and usage over time, increasing so our knowledge and culture on this technology;
- 4) we **analyze** the data collected, the efficiency of `TNT` in doing so (for tuning it to its best set of calibration parameters) and report a new quantification on MPLS deployment in the wild, correcting and updating so previous results that erroneously underestimated or overestimated the prevalence of some tunnel classes [8];
- 5) we work in a **reproducibility** perspective. As such, all our code (`TNT`, GNS-3, data processing and analysis) as well as our collected dataset are made available.<sup>3</sup>

The remainder of this paper is organized as follows: Sec. II provides the required technical background for this paper; Sec. III revises the MPLS taxonomy initially introduced by Donnet et al. [8] in the light of newly understood MPLS behaviors; Sec. IV formally introduces `TNT`, our extension to `traceroute` for revealing the content of all MPLS tunnels; Sec. V discusses `TNT` parameters and its calibration, while

<sup>2</sup>See <https://gns3.com/> Note that it is also possible to emulate other router brands, e.g., Juniper, with GNS-3.

<sup>3</sup>See <http://www.montefiore.ulg.ac.be/~bdonnet/mpls>

Router Signature	Router Brand and OS
< 255, 255 >	Cisco (IOS, IOS XR)
< 255, 64 >	Juniper (Junos)
< 128, 128 >	Juniper (JunosE)
< 64, 64 >	Brocade, Alcatel, Linux

TABLE I: Summary of main router signature, the first initial TTL of the pair corresponds to ICMP time-exceeded, while the second is for ICMP echo-reply.

Label	TC	S	LSE-TTL
0 19 20 22 23 24 31			

Fig. 1: The MPLS label stack entry (LSE) format.

Sec. VI presents results of the `TNT` deployment over the Archipelago architecture; Sec. VII positions our work with respect to the state of the art; finally, Sec. VIII concludes this paper by summarizing its main achievements.

## II. BACKGROUND

This section discusses the technical background required for the paper. Sec. II-A explains how hardware brand can be inferred from collected TTLs. Sec. II-B provides the basics of MPLS labels and introduces the MPLS control plane while Sec. II-C focuses on the MPLS data plane and MPLS TTL processing.

### A. Network Fingerprinting

Vanaubel et al. [18] have presented a router fingerprinting technique that classifies networking devices based on their hardware and operating system (OS). This method infers initial TTL values used by a router when forging different kinds of packets. It then builds the router *signature*, i.e., the  $n$ -tuple of  $n$  initial TTLs. A basic pair-signature (with  $n = 2$ ) simply uses the initial TTL of two different messages: an ICMP time-exceeded message elicited by a `traceroute` probe, and an ICMP echo-reply message obtained from an echo-request probe. Table I summarizes the main router signatures, with associated router brands and router OSes. This feature is really interesting since the two most deployed router brands, Cisco and Juniper, have distinct MPLS behaviors and signatures.

### B. MPLS Basics and Control Plane

MPLS routers, i.e., *Label Switching Routers* (LSRs), exchange labeled packets over *Label Switched Paths* (LSPs). In practice, those packets are tagged with one or more *label stack entries* (LSE) inserted between the frame header (data-link layer) and the IP packet (network layer). Each LSE is made of four fields as illustrated by Fig. 1: an MPLS label used for forwarding the packet to the next router, a Traffic Class field for quality of service, priority, and Explicit Congestion Notification [19], a bottom of stack flag bit (to indicate whether the current LSE is the last in the stack [20])<sup>4</sup>,

<sup>4</sup>To simplify the presentation we will consider only one LSE in the remainder of this paper

and a time-to-live (LSE-TTL) field having the same purpose as the IP-TTL field [21] (i.e., avoiding routing loops).

Labels may be allocated through the *Label Distribution Protocol* (LDP) [12]. Each LSR announces to its neighbors the association between a prefix in its routing table and a label it has chosen for a given *Forwarding Equivalent Class* (a FEC is a destination prefix by default), populating so a *Label Forwarding Information Table* (LFIB) in each LSR. With LDP, a router advertises the same label to all its neighbors for a given FEC. LDP is mainly used for scalability reasons (e.g., to limit BGP-IGP interactions to edge routers) and to avoid anomalies for the transit traffic such as iBGP deflection issues. Indeed, LDP deployed tunnels use the routes computed by the IGP (without any interest at the first, and naive, glance) as the LFIB is built on top of the IGP FIB. Labels can also be distributed through RSVP-TE [13], when MPLS is used for Traffic Engineering (TE) purposes. In practice, most operators deploying RSVP-TE tunnels use LDP [9] as a default labeling protocol.

With LDP, MPLS has two ways of binding labels to destination prefixes:

- 1) through ordered LSP control (default configuration of Juniper routers [22]);
- 2) through independent LSP control (default configuration of Cisco routers [23, Chap. 4]).

In the former mode, a LSR only binds a label to a prefix if this prefix is local (typically, the exit point of the LSR), or if it has received a label binding proposal from the IGP next hop towards this prefix. This mode is thus iterative as each intermediate upstream LSR waits for a proposal of its downstream LSR (to build the LSP from the exit to the entry point). Juniper routers use this mode as default and only propose labels for loopback IP addresses. In the second mode, the Cisco default one, a LSR creates a label binding for each prefix it has in its RIB (connected or – redistributed in – IGP routes only) and distributes it to all its neighbors. This mode does not require any proposal from downstream LSRs. Consequently, a label proposal is sent to all neighbors without ensuring that the LSP is enabled up to the exit point of the tunnel. LSP setup takes less time but may lead to uncommon situations in which an LSP can end abruptly before reaching the exit point (see Sec. III for details.)

The last LSR towards a FEC is the *Egress Label Edge Router* (the Egress LER –  $PE_2$  in Fig. 2). Depending on its configuration, two labeling modes may be performed. The default mode [9] is *Penultimate Hop Popping* (PHP), where the Egress advertises an Implicit NULL label (label value of 3 [20] – see Table III for details on reserved MPLS label values). The previous LSR (*Penultimate Hop LSR, PH* –  $P_3$  in Fig. 2) is in charge of removing the LSE to reduce the load on the Egress. In the *Ultimate Hop Popping* (UHP), the Egress LER advertises an Explicit NULL label (label value of 0 [20] – see Table III for details on reserved MPLS label values). The PH will use this Explicit NULL label and the Egress LER will be responsible for its removal. Labels assigned by LSRs other than the Egress LER are distinct from Implicit or Explicit NULL labels. The *Ending Hop LSR* (EH) is the LSR in charge of removing the label, it can be the PH in case of

Acronym	Meaning	IP
LSR	<u>L</u> abel <u>S</u> witching <u>R</u> outer	
PH LSR	<u>P</u> enultimate <u>H</u> op LSR	Router
EH	<u>E</u> nding <u>H</u> op LSR	
LER	<u>L</u> abel <u>E</u> dge <u>R</u> outer	Border Router
LSP	<u>L</u> abel <u>S</u> witching <u>P</u> ath	Tunnel
LSE	<u>L</u> abel <u>S</u> ack <u>E</u> ntry	
LSE-TTL	<u>L</u> SE <u>T</u> ime- <u>T</u> o- <u>L</u> ive	Header IP-TTL
LDP	<u>L</u> abel <u>D</u> istribution <u>P</u> rotocol	
RSVP-TE	<u>R</u> e <u>S</u> er <u>V</u> etation <u>P</u> rotocol – <u>T</u> raffic <u>E</u> ngineering	Signaling (control plane)
LIB	<u>L</u> abel <u>I</u> nformation <u>B</u> ase	RIB
LFIB	<u>L</u> abel <u>F</u> orwarding <u>I</u> nformation <u>B</u> ase	FIB
PHP	<u>P</u> enultimate <u>H</u> op <u>P</u> opping	
UHP	<u>U</u> ltimate <u>H</u> op <u>P</u> opping	Decapsulation
FEC	<u>F</u> orwarding <u>E</u> quivalent <u>C</u> lass	QoS Class

TABLE II: MPLS terminology with its matching in the classic IP world.

PHP, the Egress LER in case of UHP or possibly another LSR in the case of independent LSP control.

Table II provides a summary of main acronyms used by MPLS and their corresponding concept in the classic IP world.

### C. MPLS Data Plane and TTL processing

Depending on its location along the LSP, a LSR applies one of the three following operations:

- **PUSH** (Sec. II-C1). The first MPLS router, i.e., the tunnel entry point pushes one or several LSEs in the IP packet that turns into an MPLS one. The *Ingress Label Edge Router* (Ingress LER) associates the packet FEC to its LSP.
- **SWAP** (Sec. II-C2). Within the LSP, each LSR makes a label lookup in the LFIB, swaps the incoming label with its corresponding outgoing label, and sends the MPLS packet further along the LSP.
- **POP** (Sec. II-C3). The EH, the last LSR of the LSP, deletes the LSE, and converts the MPLS packet back into an IP one. The EH can be the *Egress Label Edge Router* (the Egress LER) when UHP is enabled or the PH otherwise.

Fig. 2 (above part) illustrates the main vocabulary associated to MPLS tunnels.

*1) LSP Entry Behavior:* When an IP packet enters an MPLS cloud, the Ingress LER binds a label to the packet thanks to a lookup into its LFIB, depending on the packet FEC, e.g., its IP destination prefix. Prior to pushing the LSE into the packet, the Ingress LER has to initialize the LSE-TTL (see Fig. 1). Two behaviors can then be configured: either the Ingress LER resets the LSE-TTL to an arbitrary value (255, no-ttl-propagate) or it copies the current IP-TTL value into the LSE-TTL (ttl-propagate, the default behavior). Operators can configure this operation using the no-ttl-propagate option provided by the router manufacturer [21]. In the former case, the LSP is called a *pipe LSP*, while, in the latter case, a *uniform* one.

Once the LSE-TTL has been initialized, the LSE is pushed on the packet and then sent to an outgoing interface of the Ingress LER. In most cases, except for a given Juniper OS (i.e., Olive), the IP-TTL is decremented before being encapsulated into the MPLS header.

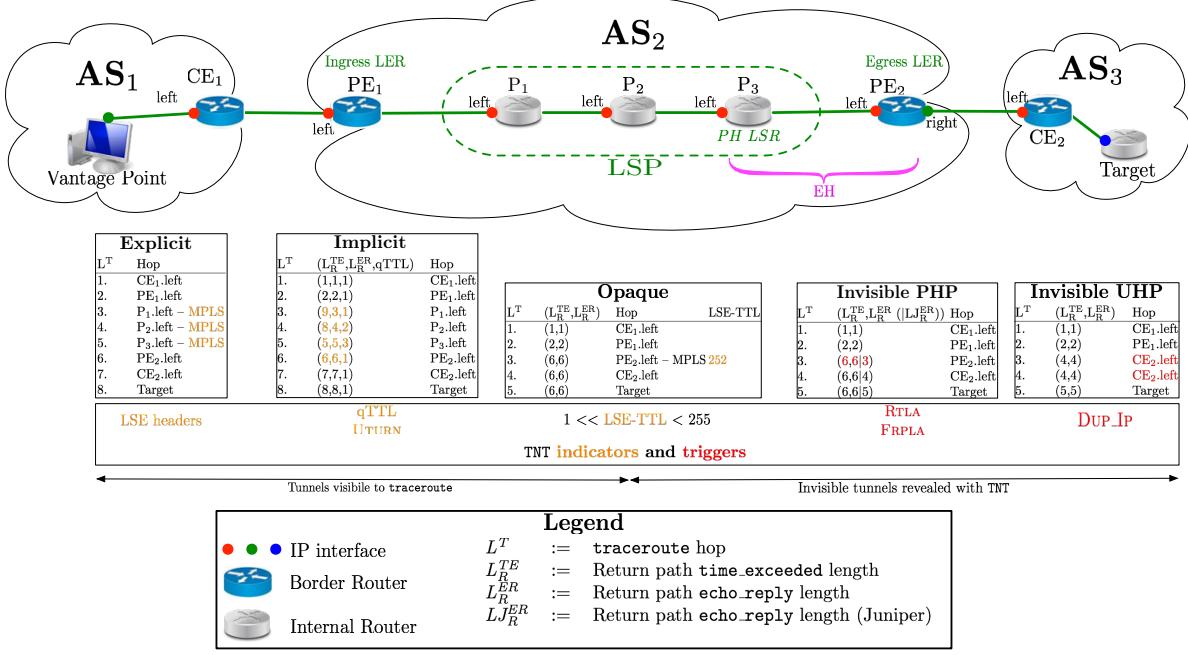


Fig. 2: Illustration of MPLS vocabulary and relationship between MPLS and traceroute. The figure is made of three parts. The upper part represents the network topology we use, throughout the paper to illustrate concepts. In particular, with respect to MPLS, P<sub>1</sub> is the LSP First Hop (FH), while P<sub>3</sub> is the Penultimate Hop (PH). In case of PHP, P<sub>3</sub> is the Ending Hop (EH) and is responsible for removing the LSE. In case of UHP, the LSE is removed by the Egress LER (PE<sub>2</sub>). The middle part of the figure presents the MPLS Tunnel classification, as observed with traceroute (this classification is a revision of the original one proposed by Donnet et al.) Finally, the bottom part of the figure provides triggers and indicators of an MPLS tunnel presence when probing with TNT. The relationship between the trigger/indicator and the observation made with probing is provided in red. Additional information (such as time\_exceeded path length) are provided. This is used in Sec. IV for illustrating TNT.

2) *LSP Internal Behavior:* Upon an MPLS packet arrival, an LSR decrements its LSE-TTL. If it does not expire, the LSR looks up the label in its LFIB. It then swaps the top LSE with the one provided by the LFIB. The operation is actually a swap only if the outgoing label returned by the LFIB is neither Implicit NULL nor empty<sup>5</sup>. Otherwise, it is a POP operation as described in the next subsection. Finally, the packet is sent to the outgoing interface of the LSR with a new label, both according to the LFIB.

If the LSE-TTL expires, the LSR, in the fashion of any IP router, forges an ICMP time\_exceeded that is sent back to the packet originator. It is worth to notice that a LSR may implement RFC 4950 [24] (as it should be the case in all recent OSes). If so, it means that the LSR will quote the full MPLS LSE stack of the expired packet in the ICMP time\_exceeded message.

ICMP processing in MPLS tunnels varies according to the ICMP type of message. ICMP *Information messages* (e.g., echo\_reply) are directly sent to the destination (e.g., originator of the echo-request) if the IP FIB allows for it (otherwise no replies are generated). On the contrary, ICMP *Error*

<sup>5</sup>In practice the actual label used for the forwarding is then greater than or equal to 0 (this specific value being reserved for Explicit NULL tunnel ending, i.e. for UHP) but excluding by design the reserved value 3 that is dedicated for Implicit NULL.

Value	Meaning	Mechanism
0	IPv4 Explicit NULL	downstream LSR should pop label immediately (UHP – popped packet is an IPv4 datagram)
1	Router Alert	deliver to control plane, do not forward
2	IPv6 Explicit NULL	downstream LSR should pop label immediately (UHP – popped packet is an IPv6 datagram)
3	Implicit NULL	pop immediately and treat as an IPv4 packet (PHP)

TABLE III: Reserved label values in MPLS and their meanings.

messages (e.g., time\_exceeded) are generally forwarded to the Egress LER that will be in charge of forwarding the packet through its IP plane [8]. Differences between Juniper and Cisco OS and configurations are discussed in details in Sec. IV-D.

3) *LSP Exit Behavior:* At the MPLS packet arrival, the EH decrements the LSE-TTL. If this TTL does not expire, the EH then pops the LSE stack after having determined the new IP-TTL.

Applying PHP comes with the advantage of reducing the load on the Egress LER, especially if it is the root of a large LSP-tree. This means that, when using PHP, the last MPLS operation (i.e., POP) is performed one hop before the

Egress LER, on the PH. On the contrary, UHP is generally used only when the ISP implements more sophisticated traffic engineering operations or wants to make the tunnel content and semantics more transparent to the customers.<sup>6</sup>

When leaving a tunnel, the router has to decide which is the correct TTL value (IP-TTL or LSE-TTL) to copy in the IP header. If the Ingress LER has activated the `no-ttl-propagate` option, the EH should pick the IP-TTL of the incoming packet while the LSE-TTL should be selected otherwise. This way, the resulting outgoing TTL cannot be greater than the incoming one: in the former case, internal hops are not counted because the tunnel is hidden while they are for the latter case. In both cases, the TTL behavior remains monotonic. In order to synchronize both ends of the tunnel without any message exchange, two mechanisms might be used for selecting the IP-TTL at the EH:

- 1) applying a `MIN(IP-TTL, LSE-TTL)` operation (solution implemented for Cisco PHP configurations [23]);
- 2) assuming the Ingress configuration (`ttl-propagate` or not) is the same as the local configuration (solution implemented by some JunOS and also in some Cisco UHP configuration).

Applying the `MIN(IP-TTL, LSE-TTL)` is the best option because it correctly supports heterogeneous `ttl-propagate` configurations in any case while, at the same time, mitigating forwarding loop without exchanging signalization messages.

This `MIN(IP-TTL, LSE-TTL)` behavior might be used for detecting the presence of hidden MPLS tunnels [9]. Indeed, it is likely that the EH generating the ICMP `time-exceeded` message will use the same MPLS cloud back to reply to the vantage point. In that case, when the reply will leave the MPLS cloud, the returning EH ( $P_1$  in Fig. 2) will choose to copy the LSE-TTL in the IP-TTL, as the IP-TTL has been initialized at its maximum value on the Egress of the forward tunnel (255 for a Cisco router – see Sec. II-A). As a consequence, while the forward path hides the MPLS cloud because the `MIN(IP-TTL, LSE-TTL)` operated on the forward PH ( $P_3$ ) selects the IP-TTL which is lower, the return path indicates its presence because the returning PH ( $P_1$ ), on the contrary, selects the LSE-TTL. In general, a sufficient condition for this pattern to occur is if the returning Ingress, which is the forward EH, re-uses the MPLS cloud back.

In practice, it is interesting to mention that this MPLS behavior is strongly dependent on the implementation and the configuration. For instance, on some Juniper OS routers (at least with JunOS Olive) or when the UHP option is activated on some Cisco IOS (at least with the 15.2 version), the `MIN(IP-TTL, LSE-TTL)` operation is not – systematically – applied. The EH assumes that the propagation configuration is homogeneous among LERs. When it is not the case (`ttl-propagate` at one end of the tunnel and `no-ttl-propagate` at the other end), the PH (for PHP routers without `MIN(IP-TTL, LSE-TTL)`) or the Egress LER (for the Cisco UHP configuration) will use the IP-

TTL instead of the LSE-TTL, leading so to a so-called *jump* effect with `traceroute` (i.e., as many hops as the LSP length are skipped after the tunnel). Except when explicitly stated, we will consider homogeneous configurations (e.g., `ttl-propagate` on the whole tunnel) in the remainder of the paper. Finally, it is worth noticing that mixing UHP and PHP (hybrid configurations) can also result in uncommon behaviors.<sup>7</sup>

### III. REVISITING MPLS TUNNELS TAXONOMY

According to whether LSRs implement RFC4950 or not (Sec. II-C2) and whether they activate the `ttl-propagate` option or not (Sec. II-C1), MPLS tunnels can be revealed to `traceroute` following Donnet et al. [8] taxonomy.

*Explicit* tunnels are those with RFC4950 and the `ttl-propagate` option activated (this is the default configuration). As such, they are fully visible with `traceroute`, including labels along the LSP. *Implicit* tunnels activate the `ttl-propagate` option but do not implement the RFC4950. No IP information is missed but LSRs are viewed as ordinary IP routers, leading to a lack of “semantic” in the `traceroute` output. *Opaque* tunnels are obscured from `traceroute` as the `ttl-propagate` option is disabled while the RFC4950 is implemented and, more decisive, the EH that pops the last label has not received an *Explicit* or *Implicit* NULL proposal for the given FEC (making the LSP end in a non controlled fashion). Consequently, the EH can be seen while the remainder of the tunnel is hidden. Finally, *Invisible* tunnels are hidden as the `no-ttl-propagate` option is activated (RFC4950 may be implemented or not).

As illustrated in Fig. 2 (middle part), *Explicit* tunnels are the ideal case as all the MPLS information comes natively with `traceroute`. For *Implicit* tunnels, Donnet et al. [8] have proposed techniques for identifying the tunnel based on the way LSRs process ICMP messages (see Sec. II-C2 – the so-called UTURN) and the IP-TTL quoted in the `time-exceeded` message (the so-called qTTL) that is increased by one at each subsequent LSR of the LSP due to the `ttl-propagate` option (ICMP `time-exceeded` are generated based on the LSE-TTL while the IP-TTL of the probe is left unchanged within the LSP and, thus, quoted as such in the ICMP `time-exceeded`).

*Opaque* tunnels are only encountered with Cisco LSPs and are a consequence of the way labels are distributed with LDP (see Sec. II-B). Indeed, a label proposal may be sent to all neighbors without ensuring that the LSP is enabled up to the Egress LER, leading so to *Opaque* tunnels because an LSP can end abruptly without reaching the Egress LER (where the prefix is injected in the IGP) that should bind an *Explicit* (UHP) or *Implicit* NULL label (PHP). As illustrated in Fig. 2, *Opaque* tunnels and their length can be identified thanks to the LSE-TTL. These LSPs end without a standard terminating label (*Implicit* or *Explicit* NULL) and so they *break* with the last MPLS header of the neighbor that may not be an MPLS speaker. Thanks to our large scale campaign

<sup>6</sup>The UHP feature has been recently made available on Juniper routers when LSPs are set with LDP. However, PHP remains the rule on Juniper [25, Chap. 1].

<sup>7</sup>Those behaviors are described and discussed in details in the Appendix, at the end of this paper.

and cross-validation with our emulation platform, we realized that the vast majority of Opaque tunnels seems to be caused by Carrier-of-Carriers VPN [26] or similar technologies. Indeed, they provoke an abrupt tunnel ending (the bottom label is necessarily carried up to the end of the tunnel to determine the correct outgoing VPN), and unfortunately lead to non revealable tunnels as we will show later.

The traceroute behavior, for Invisible tunnel, is different according to the way the LSE is popped from the packet (i.e., UHP or PHP), as illustrated in Fig. 2. Invisible tunnels are problematic, as they lead to a false vision of the Internet topology, creating false links, and spoiling graph metrics, such as the node degree distribution [9]. In this paper, we revisit the original taxonomy by doing a clear distinction between Invisible tunnels produced with PHP and UHP. In Donnet et al. [8], the class “Invisible” only covered PHP. Vanaubel et al. [9] have since proposed techniques for revealing the content of Invisible MPLS tunnels only in the case of PHP.

With Invisible UHP tunnels, the behavior is clearly different, at least for Cisco routers using the 15.2 IOS<sup>8</sup>. Upon reception of a packet with IP-TTL of 1, the Egress LER does not decrement this TTL, but, rather, forwards the packet to the next hop ( $CE_2$  in the example), so that the Egress does not show up in the trace. In contrast, the next hop will appear twice: once for the probe that should have expired at the Egress and once at the next probe. UHP indeed provokes a surprising pattern, a duplicated IP at two successive hops, illustrated as “Invisible UHP” in Fig. 2. This duplicated IP addresses might be misunderstood as a forwarding loop.

On the contrary, PHP moves the POP function at the PH, one hop before the tunnel end. This PH does not decrement the IP-TTL whatever its value is. Except for some JunOS, the packet is still MPLS switched because the LSE-TTL has not expired on it. It is somehow surprising because for Explicit and Implicit tunnels, the PH replies on its own. It is because, in this cases, the LSE-TTL has also expired. In Fig. 2, we can see that there is no more asymmetry in path length for router  $P_3$  proving so its reply does not follow a UTURN via the Egress. On the contrary, any other LSR on the LSP builds a time-exceeded message when the LSE-TTL expires and then continues to MPLS switch their reply error packet to the Egress LER unless the `mpls ip ttl-expiration pop <stack size>` command has been activated for Cisco routers. It seems to be just an option for Juniper routers with the `icmp-tunneling` command.

Note that Opaque and Invisible UHP tunnels are Cisco tunnels (signature  $< 255, 255 >$ ) due to specific implementations. Invisible PHP are Juniper (signature  $< 255, 64 >$ ), Linux boxes (signature  $< 64, 64 >$ ), or Cisco tunnels but they do not behave exactly the same as we will explain later.

Sec. IV extends techniques for revealing MPLS tunnels by proposing and implementing integrated measurement techniques for all tunnels (i.e., Explicit, Implicit, Opaque, and both UHP and PHP Invisible ones) in a single tool called TNT.

<sup>8</sup>While it is now possible to enable UHP with Juniper for LDP, TNT is not able to make a distinction between the two because the visible and revealed patterns are the same.

#### IV. TNT DESIGN AND REPRODUCIBILITY

This section introduces our tool, TNT (Trace the Naughty Tunnels), able to reveal MPLS tunnels along a path. TNT is an extension to Paris Traceroute [11] so that we avoid most of the problems related to load balancing. TNT has been implemented within Scamper [16], the state-of-the-art network measurements toolbox, and is freely available.<sup>3</sup> Sec. IV-A provides an overview of TNT, while Sec. IV-B and Sec. IV-C focus on techniques for revealing hidden tunnels and how those techniques are triggered. Finally, Sec. IV-D explains how we validated TNT on a GNS-3 platform<sup>2</sup>, an emulator running the actual OS of real routers in a virtualized environment.

##### A. Overview

Listing 1: Pseudo-code for TNT

```

1 Codes := 0, None ; 1, LSE ; 2, qTTL ; 3, UTURN ; 4, LSE-TTL ;
2   5, FRPLA ; 6, RTLA ; 7, DUP_IP .
3 trace_naughty_tunnel(target):
4   prev_hop, cur_hop, next_hop = None
5
6   for (ttl=STARTING_TTL, !halt(ttl, target), ttl++)
7     state, tun_code = None
8     next_hop = trace_hop(ttl)
9
10  # first check uniform tunnel evidence with indicators
11  tun_code = check_indicators(cur_hop)
12  #possibly fires TNT with triggers or opaques tunnels
13  if (tun_code == None)
14    tun_code = check_triggers(prev_hop, cur_hop,
15      next_hop)
16  #check if cur_hop does not belong to a uniform LSP
17  if (tun_code != None)
18    #potential hidden tunnel to reveal
19    state = reveal_tunnel(prev_hop, cur_hop,
20      tun_code)
21
22  elif (tun_code == LSE-TTL)
23    #potential opaque tunnel to reveal
24    state = reveal_tunnel(prev_hop, cur_hop, tun_code)
25
26  #hop by hop and tunnel display
27  dump(cur_hop, tun_code, state)
28
29  #sliding pair of IP addresses
30  prev_hop = cur_hop #candidate ingress LER
31  cur_hop = next_hop #candidate egress LER

```

TNT is conceptually illustrated in Listing 1. At the macroscopic scale, the `trace_naughty_tunnel()` function is a simple loop that fires probes towards each processed target. TNT consists in collecting, in a hop-by-hop fashion, intermediate IP addresses (`trace_hop()` function) between the vantage point and the target. Tracing a particular destination ends when the `halt()` function returns true: the target has been reached or a gap has been encountered (e.g., five consecutive non-responding hops). TNT uses a moving window of two hops such that, at each iteration, it considers a potential Ingress LER (i.e., `prev_hop`) and a potential Egress LER (i.e., `cur_hop`) for possibly revealing an Invisible tunnel between them. Indicators allow to check if the current hop does not belong to a uniform tunnel, i.e., a visible one (see line 11).

For each pair of collected IP addresses with `trace_hop()`, TNT checks for the presence of tunnels through so called *indicators* and *triggers*. The former provides reliable indications about the presence of an MPLS tunnel without necessarily requiring additional probing.

Generally, indicators correspond to uniform tunnels (or to the last hop of an Opaque tunnel), and are, mostly, basic evidence of visible MPLS presence such, as LSEs quoted in the ICMP time-exceeded packet (see Sec. IV-B for details). Triggers are mainly unsigned values suggesting the potential presence of Invisible tunnels through a large shifting in path length asymmetry (see Sec. IV-B for details). When exceeding a given threshold  $T$ , such triggers fire path revelation methods (function `reveal_tunnel()`) between the potential Ingress and Egress LERs as developed in Sec. IV-C. If intermediate hops are found, they are stored in a global stack structure named `revealed_lsrs`.

`STARTING_TTL` is a parameter used to avoid tracing repeatedly the nodes close to the vantage point [27], usually  $STARTING\_TTL \in [3, 5]$ .

Finally, at each loop iteration, the collected data is dumped into a warts file, the Scamper file format for storing IPv4/IPv6 traceroute records. This job is performed by the `dump()` function. It writes potential revealed hops (available in the global stack structure `revealed_lsrs`), and any useful information, such as tags, identifying the tunnel's type, and revelation method, if any.

## B. Indicators and Triggers

Listing 2: Pseudo-code for checking indicators

```

1  code check_indicators(hop):
2      #hop must exist
3      if (hop == None):
4          return None
5
6      if (is_mpls(hop)):
7          if ( $\mathcal{T}_{LSE\_TTL} < hop.lse\_ttl < 255$ ):
8              #opaque tunnel are both indicators and triggers
9              return LSE-TTL
10         else:
11             #explicit tunnel
12             return LSE
13
14         if (hop.qttl > 1):
15             #implicit tunnel
16             return qTTL
17
18         #retrieve path length from raw TTLs
19         L_R^TE = path_len(hop.ttl_te)
20         L_R^ER = path_len(hop.ttl_er)
21
22         #UTURN will be turned into RTLA for junOS signatures
23         if ( $|L_R^TE - L_R^ER| > \mathcal{T}_{UTURN}$  && !signature_is_junOS(hop)):
24             #implicit tunnel
25             return UTURN
26
27         return None

```

Tunnels indicators are pieces of evidence of MPLS tunnel presence and concern cases where tunnels (or parts of them) can be directly retrieved from the original traceroute. They are used for Explicit tunnels and uniform/visible tunnels in general. Explicit tunnels are indicated through LSEs directly quoted in the ICMP time-exceeded message – See line 12 in Listing 2 and traceroute output on Fig. 2. It is worth noting that Fig. 2 highlights the main patterns TNT looks for firing or not additional path revelation in a simple scenario where forward and return paths are symmetrical.

The indicator for Opaque tunnels consists in a single hop LSP with the quoted LSE-TTL not being equal to 1, due to the

way labels are distributed within Cisco routers (see Sec. II-B) or the way Cisco routers deal with VPRN tunnel ending.<sup>9</sup> This is illustrated in Fig. 2 where we get a value of 252 because the LSP is actually 3 hops long. This surprising quoted LSE-TTL is an evidence in itself. It is illustrated in lines 7 to 9 in Listing 2, where a hop is tagged as Opaque if the quoted LSE-TTL is between a minimum threshold,  $\mathcal{T}_{LSE\_TTL}$  (see Sec. V for fixing a value for the threshold) and 254 (LSE-TTL is initialized to 255 [21]). Note that this pattern resulting from an Opaque tunnel is both an indicator and a trigger: TNT passively understands the tunnel is incomplete and try to reveal its content with new active measurements.

Implicit tunnels are detected through qTTL and/or UTURN indicators [8]. First, if the IP-TTL quoted in an ICMP time-exceeded message (qTTL) is greater than one, it likely reveals the ttl-propagate option at the Ingress LER of an LSP. For each subsequent traceroute probe within the LSP, the qTTL will be one greater, resulting in an increasing sequence of qTTL values. This indicator is considered in line 14 in Listing 2. Second, the UTURN indicator relies on the fact that, by default, LSRs send ICMP time-exceeded messages to the Egress LER which, in turns, forwards the packets to the probing source. However, they reply directly to other kinds of probes (e.g., echo-request) using their own IP forwarding table, if available. As a result, return paths are generally shorter for echo-reply messages than for time-exceeded replies. Thereby, UTURN is the signature related to the difference in these lengths. This is illustrated in Fig. 2 (Implicit and Explicit tunnels follow the same behavior except for RFC4950 implementation). On  $P_1$ , we have UTURN ( $P_1 = L_R^TE - L_R^ER = 9 - 3 = 6$ ). With a symmetric example, one can formalize the UTURN pattern for an LSR  $P_i$  in an LSP of length  $LL$  as follows:

$$UTURN(P_i) = 2 \times (LL - i + 1). \quad (1)$$

Due to the iBGP path heterogeneity (the IGP tie-break rule in particular), the BGP return path taken by the ICMP echo-reply message can be different from the BGP return path taken by the time-exceeded reply. This is illustrated in Fig. 3a where the two return paths in blue and red can differ even outside the AS ( $L_R^TE$  can be distinct from  $L_R^ER$ ). As a result, and because it may differ at each intermediate hop, the UTURN indicator does not necessarily follow exactly Eqn. 1. A small variation may then appear in practice. In particular, a value of 0 can hide a true Implicit hop.

For JunOS routers, the situation is quite different. It turns out that, by default (i.e., without enabling the icmp-tunneling feature – see Appendix II for details), these routers send time-exceeded replies directly to the source, without forwarding them to the egress LER. The UTURN indicator becomes then useless. Moreover, for routers having the JunOS signature, the UTURN indicator and the RTLA trigger are computed in the same way. Thus, to avoid any confusion, TNT introduces an exception for such OS signatures (line 23 in Listing 2), and first considers the difference

<sup>9</sup>Juniper routers never lead to Opaque indicators because they behave differently as discussed in Sec. IV-D

as a trigger, and then falls back to an indicator if the revelation fails (not shown in Listing 1 for clarity). In addition, when icmp-tunneling is enabled, time-exceeded replies start with a TTL of 254, implying a bigger difference with echo-request replies, as it can be seen in Fig. 2:  $UTURN(P_1) = L_J^{ER} - L_J^{TE} = 10 - 3 = 7$  instead of 6 if  $P_1$  runs a Cisco OS.

Listing 3: Pseudo-code for checking triggers

```

1 code check_triggers(prev_hop, cur_hop, next_hop):
2     #prev_hop and cur_hop must exist
3     #duplicate IP checked on cur_hop and next_hop
4     if (prev_hop == None or cur_hop == None or prev_hop ==
5         cur_hop)
6         return None
7
8     if (cur_hop == next_hop)
9         #invisible UHP tunnel
10        return DUP_IP
11
12     #retrieve path length from raw TTLs
13     LRTE = path_len(cur_hop.ttl_te)
14     LRER = path_len(cur_hop.ttl_er)
15     LRT = cur_hop.probe_ttl
16
17     if (sign_is_junOS(cur_hop))
18         #for the JunOS signature
19         if (LRTE - LRER ≥ TRTLA)
20             #invisible PHP tunnel with JunOS
21             return RTLA
22
23     else
24         #for other signatures (raw TTLs are initialized the
25         #same)
26         if (LRTE - LRT ≥ TFRPLA)
27             #invisible PHP tunnel with other known OS
28             return FRPLA
29
30     return None

```

Indicators are MPLS passive pieces of evidence that can also prevent TNT from firing new probes (with the exception of LSE-TTL that is also a trigger for Opaque tunnels). On the contrary, triggers are active patterns suggesting the presence of Invisible tunnels (both PHP and UHP) that could be revealed using additional probing (see Sec. IV-C). Listing 3 provides the pseudo-code for checking triggers.

First, we look for potential Invisible UHP tunnel (line 7). As explained in Sec. III, Invisible UHP tunnels occur with Cisco routers using IOS 15.2. When receiving a packet with an IP-TTL of 1, the Egress LER does not decrement the TTL but, rather, forwards it directly to the next hop. Consequently, the Egress LER does not appear in the trace while, on the contrary, the next hop (CE<sub>2</sub> in Fig. 2) appears twice (duplicate IP address in the trace output).

The two remaining triggers, RTLA (Return Tunnel Length Analysis [9] – see Table IV for a summary of acronyms used by TNT) and FRPLA (Forward/Return Path Length Analysis [9]), work by using three path lengths, which are  $L_R^{TE}$  (the time-exceeded path length),  $L_R^{ER}$  (the echo-reply path length), and  $L_R^T$  (the forward traceroute path length). More precisely, RTLA is the difference between the time-exceeded and the echo-reply return path lengths, while FRPLA is the difference between the forward and the return path lengths (obtained based on traceroute probe and reply messages). TNT tries to capture significative differences between these lengths to infer the presence of MPLS tunnels, relying on two common practices of LSRs, in particular the EH, developed in the previous subsection. Both triggers are

based on the idea that replies sent back to the vantage point are also likely to cross back the MPLS cloud, which will apply the  $\text{MIN}(\text{IP-TTL}, \text{LSE-TTL})$  operation at the EH of the return tunnel. These triggers respectively infer the exact (RTLA) or approximate (FRPLA) return path length. Indeed, FRPLA is subject to BGP path asymmetry (and so, to false positives or negatives) in opposition to RTLA when it applies (it may produce some false alarms but only due to ECMP). In the absence of Invisible tunnel, we expect those triggers to have a value equal or close to 0. Indeed, in such a case, we should have  $L_R^{ER} = L_R^{TE} = L_R^T = 1$  if BGP does not interfere (see Fig. 3). Therefore, any significant deviation from this value is interpreted as the potential presence of an Invisible MPLS cloud, and thus, brings TNT to trigger additional path revelation techniques (see Sec. IV-C). In practice (look at Fig. 3b), we expect to have  $L_R^{ER} = L_R^{TE} = 1$  (due to the MIN for the echo-reply return tunnel and the pipe mode for the forward tunnel) while  $L_R^T$  directly provides the actual return tunnel length (with a value  $\geq 1$ ). It is due to the MIN operation applied by the EH of the return tunnel, which selects the LSE-TTL of the time-exceeded reply, and keeps the IP-TTL for the echo-reply packet. Indeed, in the case of the time-exceeded message, the return Ingress LER (i.e., the forward Egress LER) initializes the LSE-TTL with the same value as the IP-TTL, meaning 255. For echo-reply packets, the IP-TTL is set to 64. RTLA is not subject to any BGP asymmetry because we have  $L_R^{ER} = L_R^{TE}$ , i.e. BGP return paths have the same length. Indeed, the two messages use the same physical path, the only difference being the MIN operation applied at the EH of the return tunnel, if any.

To check for those triggers, we first extract the three key distances thanks to the reply IP-TTLs received by the vantage point (lines 11 to 13 in Listing 3). As explained by Vanaubel et al. [9], RTLA only works with JunOS routers, while FRPLA is more generic. Therefore, prior to estimate the triggers, TNT uses network fingerprinting (see Sec. II-A) to determine the router brand of the potential Egress LER (line 15 in Listing 3).

In the presence of a JunOS hardware,  $L_R^{TE}$  is compared to  $L_R^{ER}$ , as in case of an Invisible tunnel,  $L_R^{TE}$  is supposed to be greater than  $L_R^{ER}$ . Indeed, with this routing platform, time-exceeded and echo-reply packets have different initial TTL values (see Table I), and the RTLA trigger can exploit the TTL gap between those two kinds of messages caused by the  $\text{MIN}(\text{IP-TTL}, \text{LSE-TTL})$  behavior at the Egress LER (the  $L_R^{ER}$  appears longer than  $L_R^{TE}$  as the MIN operation results in a different pick). This difference represents the number of LSRs in the return LSP, and is compared to a pre-defined threshold  $T_{RTLA}$  (line 17 in Listing 3). This threshold (see Sec. V for the parameter calibration) filters all the LSPs shorter than the limit it defines. In the case depicted in Fig. 2:  $RTLA(PE_2) := L_R^{ER} - L_R^{TE} = L_R^{ER} - L_R^{TE} = 6 - 3 = 3$ . Indeed, for the echo-reply message, we have

$$\begin{aligned} \text{IP-TTL} &= 64 \\ &= \text{MIN}(\text{IP-TTL} = 64, \text{LSE-TTL} = 252) \end{aligned}$$

instead of

$$\begin{aligned} \text{IP-TTL} &= 252 \\ &= \text{MIN}(\text{IP-TTL} = 255, \text{LSE-TTL} = 252) \end{aligned}$$

for the time-exceeded reply. Note that an invisible

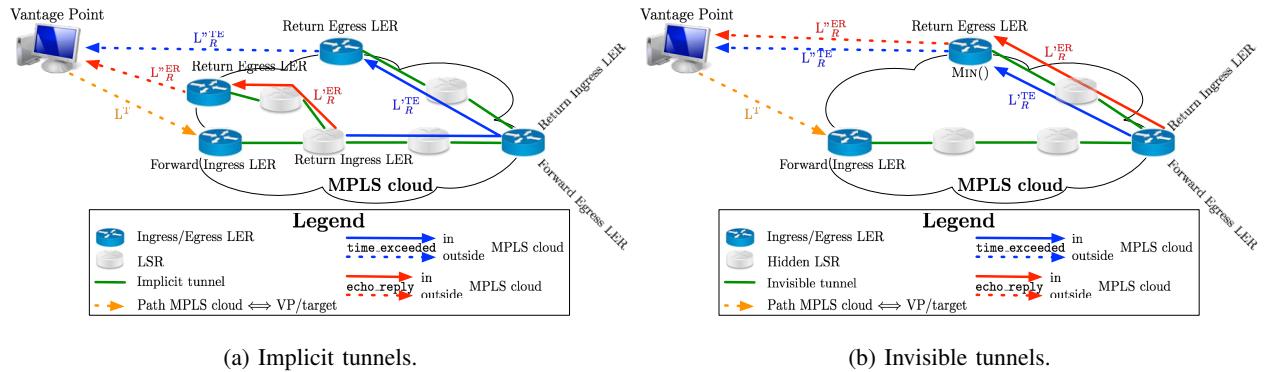


Fig. 3: Indicators and triggers illustration for Implicit and Invisible tunnels. Notations  $L'_y^x$  and  $L''_y^x$  refer to a given sub-length of an ICMP packet  $x$  on the  $y$  path ( $y$  being the forward or return path and  $x$  being a echo-reply or traceroute ICMP packet, see Fig. 2). For example,  $L_R^{TE}$  gives the return path of the time-exceeded within the MPLS cloud, while  $L''_R^{TE}$  is the return path of the time-exceeded between the MPLS cloud and the vantage point. Consequently, we have  $L_R^{TE} = L'_R^{TE} + L''_R^{TE}$ .

shadow effect also applies for RTLA after the Invisible tunnel, as the trigger will still be positive for a few nodes after the egress LER.

FRPLA is more generic and applies thus to any configuration. FRPLA allows to compare, at the AS granularity, the length distribution of forward (i.e.,  $L^T$ ) and return paths (i.e.,  $L_R^{TE}$ ). Return paths are expected to be longer than forward ones, as the tunnel hops are not counted in the forward paths while they are taken into account in the return paths (due to the MIN(IP-TTL, LSE-TTL) behavior at the return Egress LER). Then, we can statistically analyze their length difference and check if a shift appears (see Line 22 in Listing 3). This is illustrated in Fig. 2 (“Invisible PHP”) in which  $L^T$  is 3 while  $L_R^{TE}$  is equal to 6, leading so to an estimation of the return tunnel length of 3. In general, when no IP hops are hidden, we expect that the resulting distribution will look like a normal distribution centered in 0 (i.e., forward and return paths have, on average, a similar length). If we rather observe a significant and generalized shift towards positive values, it means the AS makes probably use of the no-ttl-propagate option. In order to deal with path asymmetry, TNT uses a threshold,  $T_{FRPLA}$  (see Sec. V for calibrating this parameter), greater than 0 to avoid generating too much false positives (revelation attempt with no tunnel). The MIN effect also results in an invisible shadow after the hidden LSP:  $FRPLA(CE_2) = 2$  and  $FRPLA(CE_3) = 1$ , etc until the situation returns to normal. Note that the RTLA and FRPLA shadows are the reasons why TNT does not look for consecutive Invisible tunnels in a trace. Finally, for Invisible UHP, one can observe that no MIN shift applies on the return path, as only the duplicate effect is visible.

Threshold calibration will be discussed in details in Sec. V. The optimal calibration can provide a 80/20 % success/error rates (errors being due to the BGP and ECMP noises). Moreover, the order in which TNT considers indicators and triggers, their codes, reflects their reliability, and so, their respective success rates (and their resulting states): the lower the code (i.e. the higher its priority), the more reliable (and higher the

Acronym	Meaning	Usage
FRPLA	<u>Forward/Return Path Length Analysis</u>	
RTLA	<u>Return Tunnel Length Analysis</u>	Trigger
DPR	<u>Direct Path Revelation</u>	
BRPR	<u>Backward Recursive Path Revelation</u>	Path Revelation

TABLE IV: Summary of acronyms used by TNT.

revelation success rate). Thus, if a hop matches simultaneously multiple triggers (RTLA and FRPLA for example), it is tagged with the one having the highest priority (i.e., RTLA in our example).

### *C. Hidden Tunnels Revelation*

Listing 4: Pseudo-code for revealing Invisible tunnels

```

state reveal_tunnel(ingress, egress, tun_code):
    #ingress and egress hops must exist
    if (ingress == None or egress == None)
        return None
    buddy_bit = False
    #standard traceroute towards the candidate egress
    target = egress
    route = trace(REV_STARTING_TTL, target)

    if (last_hop(route) != egress)
        #the target does not respond (revelation is not
        #possible)
        return TARGET_NOT_REACHED
    else if (ingressnotin route)
        #the forwarding path differs (revelation is not
        #possible)
        return ING_NOT_FOUND
    else if (distance(ingress, egress, route) > 1)
        #path segment revelation with \dpr
        push_segment_to_revelation_stack(ingress, egress, route
            )
    return DPR
else
    ttl = ingress.probe_ttl + 1
    revealed_ip = extract_hop(ttl, route)

    for iTR=0;;
        if (revealed_ip == target)
            if (tun_code != DUP_IP || buddy_bit)
                #no more progression in the revelation
                break
            else
                #try with the buddy for the DUP IP trigger

```

```

31     target = buddy(revealed_ip)
32     buddy_bit = True
33   else
34     # a new hop has been revealed
35     iTR++
36     push_hop_to_revelation_stack(revealed_ip)
37     target = revealed_ip
38     buddy_bit = False
39
40     revealed_ip = traceHop(ttl, target)
41
42   if (iTR == 0)
43     #no revelation (fail)
44     return NOTHING_TO_REVEAL
45   if (iTR == 1)
46     #single hop revealed LSP (\dpr ≈ \brpr)
47     return 1HOP_LSP
48   else
49     #hop by hop revelation with \brpr
50     return BRPR

```

Listing 4 offers a simplified view of the TNT tunnel revelation. The first step consists in launching a standard traceroute towards the candidate Egress<sup>10</sup> (line 8 in Listing 4). REV\_STARTING\_TTL is the starting TTL used for the revelation, which corresponds to 2 hops before the candidate Ingress hop, by default. During this first attempt, TNT may fail to reach the candidate Egress (line 12), and/or the candidate Ingress (line 15) when collecting the active data. Otherwise, TNT may reveal a tunnel and four additional output states can arise:

- an LSP composed of at least 2 LSRs is revealed in the first trace towards the Egress (line 19 – DPR, Direct Path Revelation [9] – Table IV for a summary of acronyms used by TNT);
- an LSP having more than one LSR is revealed using several iterations (line 50 – BRPR, Backward Recursive Path Revelation [9]);
- nothing is revealed, the candidate Ingress and Egress are still consecutive IP addresses in the trace towards the candidate Egress (line 44);
- a single-hop LSP is revealed (line 47) although several iterations have been tried: DPR and BRPR cannot be distinguished for one hop LSPs.

With the default UHP configuration on Cisco IOS 15.2, an additional test, called *buddy* (line 31), is required to retrieve the outgoing IP interface of the Egress LER (the right interface, in green, on PE<sub>2</sub> in Fig. 2), and thanks to its retrieval, force replies from its incoming IP interface (the left one, in red, on PE<sub>2</sub> in Fig. 2). The *buddy()* function assumes a point-to-point connection between the Egress LER and the next hop (IP addresses on this point-to-point link are called *buddies*). In most cases, the corresponding IP addresses belong to a /31 or a /30 prefix [4], [28]. Note that according to the IP address submitted to *buddy()*, this function may require additional probing to infer the correct prefix in use. Besides, specific UDP probing is necessary in order to provoke destination-unreachable messages. Such error messages, as *time-exceeded* ones, enable to get the incoming interface of the targeted router (instead of *echo-reply* ones that are indexed with the target IP).

<sup>10</sup>We use the term *candidate* as, at this point, we are not completely sure an MPLS tunnel is hidden there.

DPR (Direct Path Revelation) works when there is no MPLS tunneling for internal IGP prefixes other than loopback addresses, i.e., the traffic to internal IP prefixes is not MPLS encapsulated (default Juniper configuration but can also be easily configured on Cisco devices – see Sec. II-B). With PHP, BRPR (Backward Recursive Path Revelation) works because the target (PE<sub>2</sub>.left on Fig. 2) belongs to a prefix being also advertised by the PH. Thus, the probe is popped one hop before the PH (P<sub>3</sub> on Fig. 2), and it appears in the trace towards the Egress incoming IP interface, e.g., PE<sub>2</sub>.left on Fig. 2. BRPR is then applied recursively on the newly discovered interface until no new IP address is revealed. BRPR works also natively with UHP on IOS 12.4 (i.e., without the *buddy()* function), for the same reason as for PHP: the prefix is local and shifts the end of the tunnel one hop before and, in this implementation, the EH replies directly. On the contrary, TNT needs to use the *buddy()* function at each step for IOS 15.2 enabling UHP, because the EH silently forwards the packet one hop ahead. Vanaubel et al. [9] provides more details on DPR and BRPR.

#### D. Reproducibility and Practical BGP Configurations

We use the GNS3 emulation environment for several purposes. First, we aim at verifying that the inference assumptions we considered in the wild are correct and reproducible in a controlled environment. Second, some of the phenomena we exploit to reveal tunnels in the wild have been directly discovered in our testbed. Indeed, using our testbed we reverse-engineered the TTL processing (considering many MPLS configurations, we study the POP operation in particular) of some common OSes used by many real routers. Finally, it is also useful for debugging TNT to test its features in this controllable environment. Generally speaking, we aim at reproducing with GNS3 all common behaviors observed in the wild, and, on the opposite, we also expect to encounter in the wild all basic behaviors (based on standard MPLS and BGP configurations) we build and setup within GNS3.

In practice, we have considered four distinct router OSes: two Cisco standard IOS (12.4 and 15.2), and two virtualized versions of JunOS (Olive and VMX, the only Juniper OSes we succeeded to emulate within GNS3). We envision in a near future to also test the IOS XR and some other Juniper OSes, if possible, but we believe that our tests are already representative enough of most behaviors existing in the wild.

In our emulations, topologies (see Fig. 2) are configured as follows. We assumed that LERs are AS Provider-Edge (PE) routers, i.e., AS border routers of the ISP running (e)BGP sessions. Two main configurations are then possible to enable transit tunneling at the edges. Either the BGP next-hop can be the loopback IP address of the PE itself (with *next hop self* command), or it belongs to the eBGP neighbor – and in that case the connected subnet or the IP address should be redistributed in the ISP. In both cases, there exists a LDP mapping, at each Ingress LER and for any transit forwarding equivalent class (FEC) between the BGP next-hop, the IGP next-hop, and the local MPLS label to be pushed. According to the configuration at the Egress LER, when the Ingress LER is in pipe mode (see Sec. II-C1), distinct kinds of tunnels emerge: Opaque, UHP Invisible, or PHP Invisible.

We consider the simplest possible configurations, i.e., homogeneous in terms of OS and MPLS+BGP configurations. They are consistent and symmetric MPLS configurations both in terms of signaling (LDP with the independent model using all IGP connected prefix – Cisco default mode – xor the ordered model using only loopback addresses – Juniper default mode)<sup>11</sup> and the propagation operation in use (pipe xor uniform)<sup>12</sup> at the domain scale. Using heterogeneous configurations, we discovered many intriguing corner cases that are discussed in Appendix I-D. Some of them may result in incorrect TTL processing and other in hiding even more the tunnel to TNT. In some rare cases, only the Brute Force option of TNT is able to fire the path revelation that exposes tunnels.

The BGP configuration is also standard: the Egress LER enables the next-hop-self feature and so the transit traffic is tunneled via this IP address. All LSRs also have a global IGP routing table thanks to a route reflector (they can answer natively to ping requests) or a redistribution in the IGP routing control plane. The AS scale BGP prefix is advertised using a global aggregation and the BGP inter-domain link is addressed by the neighbor but can be redistributed in the IGP as a connected one.

Opaque tunnels show up when enabling the neighbor <IP> `ebgp-multihop <#hops>` command towards the BGP neighbor whose IP address is redistributed statically in the IGP. DPR works also with Cisco IOS when enabling the `mpls ldp label allocate global host-routes` command. Eventually, the command `mpls ldp explicit-null [for prefix-acl]` allows for revealing UHP tunnels without the use of the buddy.

Table V provides a summary of TNT capacities considering several MPLS usages. In particular, it provides many information about the way TNT is able to collect information about tunnels in default cases (i.e., standard configurations). For example, it shows that TNT is able to discriminate between Cisco Invisible UHP and PHP tunnels while it is not the case for Juniper routers (the  $\square$  is not colored). Indeed, for both UHP/PHP Juniper configurations, the trigger and the revelation methods are the same (RTLA and DPR). Moreover, we also show when our basic set of techniques need to be extended for enabling revelation and distinction among different classes: we use the symbol `++` to enforce these new requirements. In particular, for revealing UHP Cisco tunnels, TNT extends BRPR with additional `buddy()` function and UDP probing.

Another kind of MPLS technology that conduct to surprising patterns is VPRN tunnels. They also lead to Opaque tunnels for Cisco routers and twisted traces with Juniper routers.

As shown by Table V, TNT is able to reveal the content of all basic point-to-point (P2P) MPLS circuits (i.e., LDP or RSP-TE tunnels). However, things are not as easy when point-to-multipoint (P2MP) tunnels join the party (second part of Table V), leading typically to Opaque tunnels. In such a situation, TNT is only able to detect their presence but

Configurations	Pop	Cisco iOS15.2	Juniper VMX
P2P circuits (e.g. LDP or RSVP-TE tunnels)	PHP UHP	FRPLA, BRPR DUP_IP, BRPR <code>++</code> $\square$	RTLA, DPR RTLA, DPR $\square$
P2MP overlays (e.g. VPRN: CsC or VPN BGP-MPLS)	PHP UHP	LSE-TTL, - LSE-TTL++, - $\square$	RTLA++, - N/A $\square$

TABLE V: TNT revelation ( $\square$ ) and detection ( $\boxtimes$ ) capacities according to the OS and the MPLS tunnel flavor (i.e. the MPLS L3 tunneling underlying technologies for a given usage). In particular, this table provides the default indicator/trigger and the default path revelation method (when it applies).

without revealing their content. Indeed, such tunnels may result from various network configurations, e.g., heterogeneous routing devices (a combination of Juniper and Cisco devices in particular), specific BGP edge configurations as already introduced, or VPRN (Virtual Private Routed Network) [14]. As for the two former configurations, the Opaque pattern arises with VPRN because of an abrupt tunnel ending, i.e., the LSP ends without a standard ending label (Implicit or Explicit NULL). Indeed, the last hop towards the Egress contains at least one label (two with UHP, the top label then being 0). The inner label is used to identify the VPN and the associated VRF containing routes. By definition, the VPN label value is neither Explicit NULL nor Implicit NULL. Upon receiving this non-terminating label, the Egress behaves as if the tunnel did not end in a controlled fashion.

Using our GNS-3 platform, it appears that VPRN content cannot be revealed with TNT, while other Opaque tunnels configurations (i.e., routing devices heterogeneity, BGP edge configuration) can. The mechanism behind the absence of content revelation can be explained by the IP address collected by TNT from the source IP field in the ICMP reply. Usually, the collected address is the one assigned to the physical incoming interface of the Egress PE. In the VPRN case, the collected IP is the one assigned to the interface on which the VRF is attached. In practice, this corresponds to the outgoing interface towards the VPN at the customer's side. Said otherwise, TNT collects the outgoing address instead of the incoming one. Because the incoming address is the only one that enables a successful revelation, this type of Opaque tunnels cannot be revealed yet. Table VII in Sec. VI will show that, in the wild, VPRNs are the most prevalent case of Opaque tunnels.

Whatever the kind of probes sent to or through the VPRN, the IP address visible to TNT (or traceroute in general) is the outgoing address. Despite its expired TTL, it is likely that the probe arriving on the Egress PE will be pushed to the VRF of the VPN and its associated interface before generating the error message (the VPN being identified with the MPLS label contained in the packet). Then, the interface where the packet actually expires is the one associated to the VRF. However, as shown in Table V, we are able to distinguish UHP and PHP configurations (thanks to LSE-TTL`++`), because the bottom label is equal to 255 for UHP and lower with PHP.

With Juniper VPN, there is no Opaque indicator resulting from VPRN or any other configurations. A first explanation

<sup>11</sup>See Sec. II-B

<sup>12</sup>See Sec. II-C1

is that Juniper routers, on the contrary to the independent mode enabled by default with Cisco routers, do not inject the whole IGP in LDP, but only their loopback address using the ordered mode (see Sec. II). This mode limits the probability to face a non-controlled tunnel ending. However, with VPRN configurations, a Juniper Egress LER deals with the same packet level situation as with Cisco routers. Up to the end of the tunnel, the packet is still MPLS encapsulated with the end-to-end VPN non terminating label at the bottom of the stack. Juniper routers do not, however, produce an Opaque indicator in that situation. Indeed, packets destined to the VPN are handled in a specific way with Juniper devices: they are IP packets forwarded directly to the next-hop without looking at or manipulating the IP-TTL whatever its value.

The outcome of such a sliding packet is twofold. Firstly, the Egress hop is hidden in the transit trace, as with Cisco UHP but without the duplicated IP. Secondly, when performing a direct trace (even with UDP) targeting the first address of the path within the VPN, i.e. the IP interface of the Egress LER belonging to the VPN, one can see that this address and its buddy appear in the wrong order. Indeed, in the trace, the two addresses are switched, meaning that the CE IP address appears before the Egress one. Being forwarded without inspecting the IP-TTL, probes targeting IP addresses belonging to the VPN are automatically forwarded to the CE router, where they expire. The next probe, having a greater TTL, follows the same path as the one before, but can be forwarded back to the Egress LER by the CE router before expiring. This loop results in the two addresses being switched regarding their actual location in the path. Finally, one can infer the loop because two additional artifacts compared to RTLA (RTLA++) are visible: the TTL that deviates from its monotony and subsequent IP addresses also raise alarms due to potential conflicting allocation.

Additional details on the validation through GNS3 emulation can be found in the Appendix, at the end of this paper. All topologies and scripts developed are available for reproducibility.<sup>3</sup>

## V. TNT CALIBRATION AND PROBING COST

Sec. IV shows that TNT relies mainly on four parameters when looking for tunnels indicators or triggers:  $\mathcal{T}_{LSE\_TTL}$  for Opaque tunnels,  $\mathcal{T}_{UTURN}$  for Implicit tunnels, and  $\mathcal{T}_{RTLA}$  and  $\mathcal{T}_{FRPLA}$  for PHP/UHP Invisible tunnels. This section aims at calibrating those parameters (Sec. V-B), as well as evaluating the probing cost associated to TNT (Sec. V-C).

### A. Measurement Setup

We deployed TNT on three vantage points (VPs) over the Archipelago infrastructure [17]. VPs were located in Europe (Belgium), North America (San Diego), and Asia (Tokyo).

TNT was run on April 6<sup>th</sup>, 2018 towards a set of 10,000 destinations (randomly chosen among the whole set of Archipelago destinations list). Each VP had its own list of destinations, without any overlapping.

From indicators and triggers described in Sec. IV-B (see Listing 2 and 3), it is obvious that  $UTURN$  is equivalent

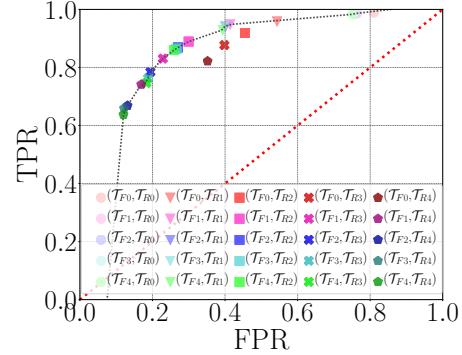


Fig. 4: Receiver operating characteristic (ROC) curve providing the efficiency of TNT according to values for Invisible tunnels parameters.  $\mathcal{T}_{R_x}$  refers to  $\mathcal{T}_{RTLA}$  with the value  $x$ , while  $\mathcal{T}_{F_y}$  to  $\mathcal{T}_{FRPLA}$  with the value  $y$ .

to  $RTLA$  for Juniper routers. However, the  $\mathcal{T}_{UTURN}$  will not have the same value than  $\mathcal{T}_{RTLA}$ .  $\mathcal{T}_{UTURN} = 0$  by design as any difference between echo-reply and time-exceeded replies for the Cisco router signature indicates LSE-/IP-TTL shifting. In practice, we reinforce the condition by looking for at least two consecutive hops having a cumulated  $UTURN \geq 3$ . Besides, we have observed that abnormal<sup>13</sup> LSE-TTL values oscillate between 236 and 254, the main proportion being located between 250 and 254. It suggests thus that, in the majority of the cases, Opaque tunnels are rather short. Consequently, a value of 236 for  $\mathcal{T}_{LSE\_TTL}$  would be enough for detecting the presence of an opaque tunnel and launching additional measurements for revealing its content.

For our tests, we varied  $\mathcal{T}_{RTLA}$  and  $\mathcal{T}_{FRPLA}$  between 0 and 4. A full measurement campaign was launched for each pair of parameter value (thus, a total of 25 measurement runs). Moreover for each pair, if no trigger is pulled, a so called brute force revelation is undertaken: DPR/BRPR are launched (with the use of the buddy if required). This brute force data is used as a basis to evaluate the quality and cost of each threshold value.

### B. Calibration

With the help of well calibrated thresholds, the results associated to FRPLA and RTLA triggers allows for a binary classification. These triggers provide a prediction, while the results of additional probing gives the true facts when some conditions apply (see resulting states of Listing 4), i.e., being or not a tunnel. With that in mind, one can assess the performance of FRPLA and RTLA triggers through the analysis of True Positive Rate (TPR) and False Positive Rate (FPR): we plot the results on a Receiver Operating Characteristic (ROC) curve in Fig. 4. We define TPR as the ratio of TNT success to the number of links being actually MPLS tunnels (having a length greater than 1): TNT triggers additional probing and actually reveals Invisible tunnels (we have  $TPR + FNR = 1$ ,

<sup>13</sup>Abnormal here means “different from 1” which is the LSE-TTL value that should be obtained in ICMP time-exceeded messages. More details can be found in our technical report [29].

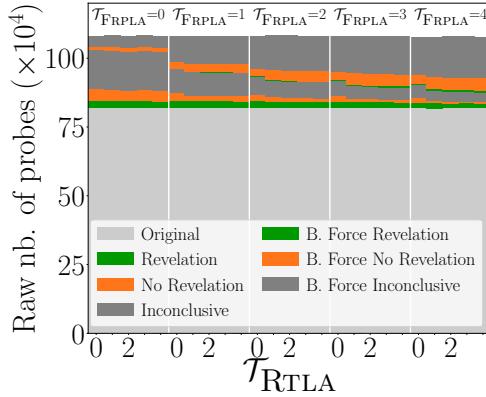


Fig. 5: Probing cost associated to TNT according to  $T_{\text{FRPLA}}$  and  $T_{\text{RTLA}}$  thresholds.

i.e., when adding to False Negative Rate, we obtain all links being long enough tunnels). FPR is defined as the ratio of TNT failure to the amount of standard IP links: it triggers for additional probing but without revealing anything (we have  $FPR + TNR = 1$ , i.e., when adding to True Negative Rate, we obtain all IP links without tunnels). Here, our brute force data gives the ground data that we consider reliable (i.e., revelation is fired at each hop and if nothing is revealed, we consider that there is no tunnel – we do not consider inconclusive cases where we obtain states ING\_NOT\_FOUND or TARGET\_NOT\_REACHED – see Listing 4). The ROC curve is obtained by varying the  $T_{\text{RTLA}}$  and  $T_{\text{FRPLA}}$  parameters between 0 and 4. The red dotted diagonal provides the separation between positive results for TNT (above part of the graph) and negative results (below part of the graph). Finally, the black dotted line is the interpolation of measurement results (at the exception of  $T_{R_0}$  values which appear as being outliers, as expected).

We observe that the results are essentially positive for TNT. Some results, between  $(T_{R_1}, T_{F_3})$  and  $(T_{R_2}, T_{F_3})$ , are even reasonably close to the perfect classification (upper left corner) and, thus, are considered as the best choice for defining our thresholds  $T_{\text{RTLA}}$  and  $T_{\text{FRPLA}}$ . We obtain a compromise close to 80%-20%: while we expect to reveal at least 80% of existing tunnels (MPLS links), TNT has a controlled overhead of 20%, i.e., it fires useless additional probing for an average limited to two actual IP links on ten.

### C. Probing Cost

Fig. 5 illustrates the probing cost associated to TNT. In particular, it focuses on additional measurements triggered by RTLA or FRPLA for revealing Invisible tunnels. The light grey zone (labeled as “Original” on Fig. 5) corresponds to probes associated to standard traceroute. The green, orange, and dark grey zones correspond to probes sent when additional measurements are triggered by RTLA or FRPLA. In particular, the green zone corresponds to additional measurements that were able to reveal the content of an Invisible tunnel. On the contrary, the orange zone refers to additional measurements that failed, i.e., no Invisible tunnel content was revealed.

Finally, the dark grey zone refers to inconclusive revelation: the trigger has led to additional measurements but TNT was unable to reach the potential Egress LER (i.e., the IP address that engaged the trigger – `cur_hop` in Listing 1 – generally due to unresponsive IP interface) or TNT was unable to reach again the candidate Ingress LER (i.e., `prev_hop` in Listing 1) because the destination has changed (ECMP or BGP routing noises).

If the amount of probes sent for actually revealing the content of an Invisible tunnel remains almost stable whatever the values for  $T_{\text{FRPLA}}$  and  $T_{\text{RTLA}}$  are, one can observe a very slow decrease meaning that there are less revealed tunnels for high values. Further, the additional traffic generated by erroneous trigger (orange) or by inconclusive revelation (dark grey) clearly decreases while  $T_{\text{FRPLA}}$  increases. This result is aligned with Sec. V-B in which the best values for  $T_{\text{FRPLA}}$  are between 2 and 3. Note that FRPLA is more generic but less reliable than other triggers. On the contrary, the  $T_{\text{RTLA}}$  threshold has a minor effect on the amount of probes sent because it is more specific and more reliable.

Hatched zones (orange, dark grey, and green) correspond to the amount of probes sent using brute force. First, on the contrary to normal behavior (i.e., revelation launched according to triggers), the amount of probes sent increases with  $T_{\text{FRPLA}}$  (the impact of  $T_{\text{RTLA}}$  is quite negligible), as well as the amount of inconclusive revelation. Second, the amount of probes having revealed an Invisible tunnel is low compared to standard behavior.

Generally speaking, one can observe that the overhead of TNT is quite limited compared to a basic active campaign and considering the information gathered. In particular, if using correct parameters to limit both useless probes and missed tunnels (e.g.,  $T_{R_1}$ ,  $T_{F_3}$ ), our tool generates less than 10% of additional probing compared to the underlying campaign for reaching a satisfying compromise where 80% of tunnels are revealed.

## VI. TUNNELS QUANTIFICATION WITH TNT

This section aims at discussing how TNT and its features behave in the wild Internet. In particular, it analyzes the success rate of each indicator and trigger with respect to possible revelation techniques. Sec. VI-A describes the measurement setup, while Sec. VI-B discusses the results obtained.

### A. Measurement Setup

We deployed TNT on the Archipelago infrastructure [17] on April 23<sup>rd</sup>, 2018 with parameters  $T_{\text{FRPLA}}$  fixed to 3 and  $T_{\text{RTLA}}$  to 1, according to results discussed in Sec. V-B.

TNT has been deployed over 28 vantage points, scattered all around the world: Europe (9), North America (11), South America (1), Asia (4), and Australia (3). The overall set of destinations, nearly 2,800,000 IP addresses, is inherited from the Archipelago dataset and spread over the 28 vantage points to speed up the probing process.

A total of 522,049 distinct IP addresses (excluding traceroute targets) has been collected, with 28,350 being non publicly routable addresses (and thus excluded from our

Status		# probes		
	attempt	traceroute	ping	buddy
original	revealed	63,559,385	7,109,075	—
	no revelation	2,190,275	206,842	19,181
	TARGET_NOT_REACHED	1,640,224	—	556
	ING_NOT_FOUND	4,174,404	—	9,888
attempt	ING_NOT_FOUND	1,790,900	—	7,326

TABLE VI: Raw number of probes sent by TNT over the set of 28 vantage points.

dataset). Each collected routable IP address has been pinged, only once per vantage point, allowing us to collect additional data for fingerprinting (see Sec. II-A). Our dataset and our post-processing scripts are freely available.<sup>3</sup>

### B. Results

Table VI provides the amount of probes sent by traceroute-like probing in TNT, ping, and buddy bit exploration. The row “original” refers to standard traceroute based revelation (i.e., nothing to reveal, Explicit, or Implicit tunnels).

The main results from Table VI is the amount of probes involved in inconclusive revelation, split between TARGET\_NOT\_REACHED (TNT was unable to reach the potential Egress LER) and ING\_NOT\_FOUND (TNT did not cross the potential Ingress LER). In particular, TARGET\_NOT\_REACHED involved twice more probes than revealed tunnels. Those particular inconclusive revelations might be explained by ICMP rate limiting between the traceroute probe and additional probing (both ping and BRPR/DPR). Another explanation is that those potential Egress LERs respond to initial traceroute with an IP address that is not globally announced. As such, additional probing following traceroute will fail as no route is available to reach them.

Table VII provides the number of MPLS tunnels discovered by TNT, per tunnel type as indicated in the first column. The indicators/triggers are provided, as well as the additional revelation technique used. Without any surprise, Explicit tunnels are the most present category (76% of tunnels discovered).

Implicit tunnels represent 5% of the whole dataset, with the UTURN indicator providing more results than qTTL. However, those results must be taken with care as UTURN is subject to false positive (implicit UTURN tunnels are likely to be overestimated because of possible confusion with RTLA for Juniper routers), while qTTL is much more reliable [30]. Compared to previous works, it is clear that this class is not as prevalent as expected at the time (because both we correct and improve our methodology and the RFC4950 is likely to be more and more deployed).

Opaque tunnels are less prevalent (1.7% of tunnels discovered). This is somewhat expected as Opaque tunnels are the results of particular label distribution within Cisco MPLS clouds. This confirms previous empirical results [8, Sec. 7.2]. It is also worth noticing that additional revelation techniques (DPR or BRPR) does not perform well with such tunnels (content of 98% of Opaque tunnels cannot be revealed). Indeed, as already

discussed earlier, this result can be explained because Carrier-of-Carriers VPN [26] or similar VPRN are not possible to reveal but can only be detected. We deduce from our campaign results that the vast majority of Opaque tunnels seems to arise from Cisco VPRN.<sup>14</sup>

The proportion of Invisible tunnels is not negligible (16% of tunnels in our dataset). Those measurements clearly contradicts our previous work suggesting that Invisible tunnels were probably 40 to 50 times less numerous than Explicit ones [8, Sec. 8]. More precisely, Invisible PHP is the most prominent configuration (87% of Invisible tunnels belongs to the Invisible PHP category), confirming so our past survey [9]. RTLA appears as being the most efficient trigger. This is partially due to the order of triggers in the TNT code because it favors high ranked trigger compared to low ranked (in case both apply, we prefer to use the most reliable, i.e., the less subject to any interference such a BGP asymmetry). As indicated in Listing 3 (Sec. IV-B), we first check for RTLA as it is more reliable than FRPLA. DPR works better than BRPR, which is obvious as it is triggered by RTLA (Juniper routers). For Invisible UHP, it is worth noticing that the buddy bit, prior to BRPR or DPR revelation, was required in nearly 25% of the cases. In other cases, a simple BRPR or DPR revelation was enough to get the tunnel content. UHP seems to be often filtered for a particular FEC, e.g., only /32 host loopback addresses are advertised in LDP with UHP while other FEC are advertised with PHP (BRPR) or are not injected at all (DPR).

The column labeled “mix” corresponds to tunnels partially revealed thanks to BRPR and partially with DPR. Typically, it comes from *heterogeneous MPLS clouds*. For instance, operators may deploy both Juniper and Cisco hardware without any homogeneous prefixes distribution (i.e., local prefix for Juniper, all prefixes for Cisco – See Sec. II-B for details). Note that it is also possible that the UHP and PHP label popping techniques co-exist when using our backward recursive path revelation (BRPR). Although not explained in Sec. IV for clarity reasons, TNT can deal with those more complex situations, making the tool quite robust to pitfalls encountered in the wild Internet (5% of the Invisible tunnels encountered).

Finally, the column labeled “1HOP\_LSP” corresponds to one hop tunnels where DPR and BRPR cannot be distinguished. This large proportion (20%) of very short Invisible tunnels is aligned with previous works that already noticed the proportion of short Explicit tunnels [8], [10], [31].

## VII. RELATED WORK

For years now, traceroute has been used as the main tool for discovering the Internet topology [1]. Multiple extensions have been provided to circumvent traceroute limits.

Doubletree [27], [32] has been proposed for improving the cooperation between scattered traceroute vantage points, reducing so the probing redundancy. Paris traceroute [11] has been developed for fixing issues related to IP load balancing, avoiding so false links between IP interfaces. tracebox [5]

<sup>14</sup>In this paper, TNT does not look for Juniper VPRN as its indicator, RTLA++, is less reliable (See Sec. IV-D).

Tunnel Type	Indicator/Trigger	Revelation Technique				# Tunnels
		DPR	BRPR	1HOP_LSP	Mix	
Explicit	LSE headers	-	-	-	-	150,036
Implicit	qTTL	-	-	-	-	2,689
	UTURN	-	-	-	-	7,216
Opaque	LSE-TTL	22	17	43	-	3,346
Invisible PHP	RTLA	11,268	1,191	2,595	279	15,333
	FRPLA	5,903	2,555	3,260	1,012	12,730
Invisible UHP	DUP_IP	1,609	1,531	686	296	4,122
<b>Total</b>		18,802	5,294	6,584	1,587	195,525

TABLE VII: Raw number of tunnels discovered by TNT per tunnel type (see Sec. III). Color code for indicators/triggers is identical to Fig. 2. No additional revelation technique is necessary for Explicit and Implicit tunnels.

extends traceroute for revealing the presence of middleboxes along a path. YARPP [33] provides techniques for speeding up the traceroute probing process. Reverse traceroute [34] is able to provide the reverse path (i.e., from the target back to the vantage point). Passenger [35] and Discarte [36] extend traceroute with the IP record route option. Marchetta et al. [37] have proposed to use the ICMP Parameter Problem in addition to Record Route option in traceroute. Finally, tracenet [38] mimics traceroute for discovering subnetworks.

TNT is also in the scope of the *hidden router* issue, i.e., any device that does not decrement the TTL causing the device to be transparent to traceroute probing. Discarte and Passenger, through the use of IP Record Route Option, allows, to some extent, to reveal hidden routers along a path. DRAGO [39] considers the ICMP Timestamp for also detecting hidden routers. TNT goes beyond those solutions as it does not rely on ICMP messages and IP option that are, generally, filtered by operators either locally (i.e., the option/message is turned off on the router) or for transit packets (i.e., edge routers do not forward those particular packets).<sup>15</sup> TNT only relies on standard messages (echo-request/echo-reply and time-exceeded) that are implemented and used by the vast majority of routers and, as such, has the potential to reveal much more information.

MPLS tunnels discovery has been the subject of several researches those last years. In particular, Sommers et al. [10] examined the characteristics of MPLS deployments that are explicitly identified using RFC4950 extensions, as observed in CAIDA’s topology data. Donnet et al. [8] proposed the first classification of MPLS tunnels according to the relationship between MPLS and traceroute. This paper is a revision of Donnet et al.’s work in light of a deeper understanding of MPLS mechanisms, in particular for hidden tunnels (Opaque, Invisible PHP and UHP). More recently, Vanaubel et al. [9] have proposed techniques for inferring and possibly revealing hidden tunnels: FRPLA, RTLA, BRPR, and DPR. FRPLA and RTLA were initially not used as triggers for measurements (as we are doing in this paper with TNT by extending those techniques in many aspects) but rather as a way to infer an hidden tunnel length. Vanaubel et al. directed their measurements towards pre-identified high degree routers with

<sup>15</sup>It has been, however, demonstrated recently that IP Record Route option might still find a suitable usage in Internet measurements if used with prudence [40].

the ITDK dataset used as a source for triggering specific measurements (as they were suspected to be the exit point of a large number of hidden MPLS tunnels). As such, Vanaubel et al. did not provide any integrated measurement tool, on the contrary to TNT, in which MPLS tunnels are discovered on the fly (TNT does not rely on any kind of external dataset).

## VIII. CONCLUSION

In this paper, we revise the MPLS classification proposed by Donnet et al. [8] and introduce TNT (Trace the Naughty Tunnels) that is an extension to Paris traceroute for revealing MPLS tunnels along a path. As such, TNT has the potential to reveal more complete information on the exact Internet topology. We provide accurate IP level tracing functions leading so to better Internet models. For instance, it has been shown that Invisible tunnels have an impact on Internet basic graph properties [9]). Our fully integrated tool reveals, or at least detect, all kind of tunnels in two simple stages: first, it uses indicators and triggers to respectively classify and possibly tag tunnels as hidden, second it reveals the tagged tunnel content if any. TNT has the capacity to unveil the MPLS ecosystem deployed by operators. Recent works on MPLS discovery have revealed that MPLS is largely deployed by most ISP [8], [31], [10]. By running TNT on a daily (or nearly daily) basis from the Archipelago platform, we expect to see numerous researches using our tool and data to mitigate the impact of MPLS on the Internet topology. TNT has been developed with a reproducibility perspective. As such, it is freely available, as well as our dataset and scripts used for processing data.<sup>3</sup>

## ACKNOWLEDGMENTS

Authors would like to thank kc claffy and her team at CAIDA for letting them deploying TNT on the Archipelago infrastructure. In addition, part of Mr. Vanaubel’s work was supported by an internship at CAIDA, under the direction of Young Hyun.

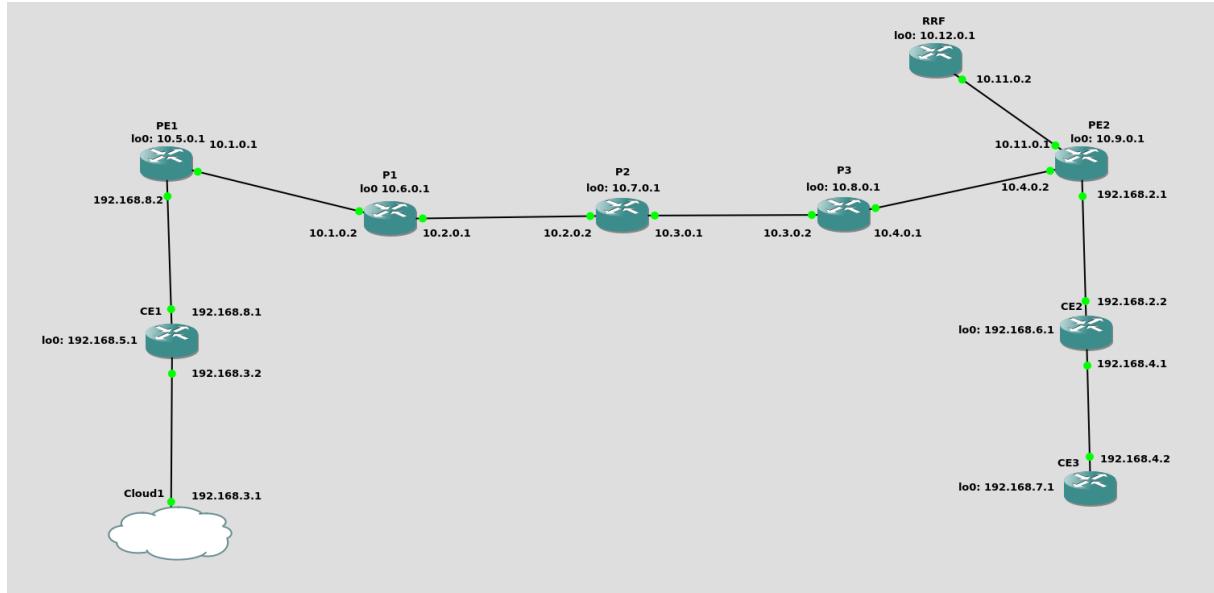
## REFERENCES

- [1] B. Donnet and T. Friedman, “Internet topology discovery: a survey,” *IEEE Communications Surveys and Tutorials*, vol. 9, no. 4, pp. 2–15, December 2007.
- [2] H. Haddadi, G. Iannaccone, A. Moore, R. Mortier, and M. Rio, “Network topologies: Inference, modeling and generation,” *IEEE Communications Surveys and Tutorials*, vol. 10, no. 2, pp. 48–69, April 2008.
- [3] R. Pastor-Satorras and A. Vespignani, *Evolution and Structure of the Internet: A Statistical Physics Approach*. Cambridge University Press, 2004.

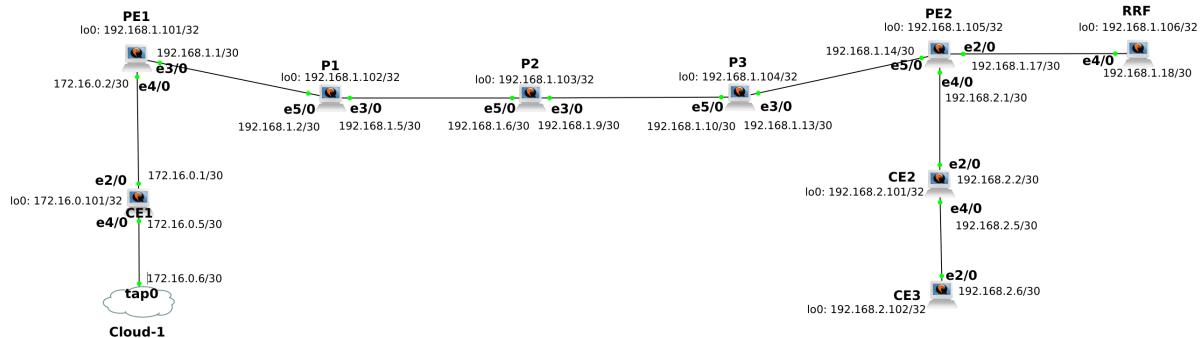
- [4] P. Mérindol, B. Donnet, O. Bonaventure, and J.-J. Pansiot, "On the impact of layer-2 on node degree distribution," in *Proc. ACM Internet Measurement Conference (IMC)*, November 2010.
- [5] G. Detal, b. Hesmans, O. Bonaventure, Y. Vanaubel, and B. Donnet, "Revealing middlebox interference with tracebox," in *Proc. ACM Internet Measurement Conference (IMC)*, October 2013.
- [6] K. Edeline and B. Donnet, "A first look at the prevalence and persistence of middleboxes in the wild," in *Proc. International Teletraffic Congress (ITC)*, September 2017.
- [7] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," Internet Engineering Task Force, RFC 3031, January 2001.
- [8] B. Donnet, M. Luckie, P. Mérindol, and J.-J. Pansiot, "Revealing MPLS tunnels obscured from traceroute," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 2, pp. 87–93, April 2012.
- [9] Y. Vanaubel, P. Mérindol, J.-J. Pansiot, and B. Donnet, "Through the wormhole: Tracking invisible MPLS tunnels," in *In Proc. ACM Internet Measurement Conference (IMC)*, November 2017.
- [10] J. Sommers, B. Eriksson, and P. Barford, "On the prevalence and characteristics of MPLS deployments in the open Internet," in *Proc. ACM Internet Measurement Conference (IMC)*, November 2011.
- [11] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira, "Avoiding traceroute anomalies with Paris traceroute," in *Proc. ACM Internet Measurement Conference (IMC)*, October 2006.
- [12] L. Andersson, I. Minei, and T. Thomas, "LDP specification," Internet Engineering Task Force, RFC 5036, October 2007.
- [13] D. Awduch, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP tunnels," Internet Engineering Task Force, RFC 3209, December 2001.
- [14] B. Gleeson, A. Lin, J. Heinanen, G. Armitage, and A. Malis, "A framework for IP based virtual private networks," Internet Engineering Task Force, RFC 2764, February 2000.
- [15] Center for Applied Data Analysis, "The CAIDA UCSD internet topology data kit," March 2016, see <http://www.caida.org/data/internet-topology-data-kit>.
- [16] M. Luckie, "Scamper: a scalable and extensible packet prober for active measurement of the Internet," in *Proc. ACM Internet Measurement Conference (IMC)*, November 2010.
- [17] H. Y. claffy, kc, K. Keys, M. Fomenkov, and D. Krioukov, "Internet mapping: from art to science," in *Proc. IEEE Cybersecurity Application and Technologies Conference for Homeland Security (CATCH)*, March 2009.
- [18] Y. Vanaubel, J.-J. Pansiot, P. Mérindol, and B. Donnet, "Network fingerprinting: TTL-based router signature," in *Proc. ACM Internet Measurement Conference (IMC)*, October 2013.
- [19] L. Andersson and R. Asati, "Multiprotocol label switching (MPLS) label stack entry: EXP field renamed to traffic class field," Internet Engineering Task Force, RFC 5462, February 2009.
- [20] E. Rosen, D. Tappan, G. Fedorkow, Y. Rekhter, D. Farinacci, T. Li, and A. Conta, "MPLS label stack encoding," Internet Engineering Task Force, RFC 3032, January 2001.
- [21] P. Agarwal and B. Akyol, "Time-to-live (TTL) processing in multiprotocol label switching (MPLS) networks," Internet Engineering Task Force, RFC 3443, January 2003.
- [22] D. Aydin, "CISCO vs. Juniper MPLS," June 2014, see <http://monsterdark.com/cisco-vs-juniper-mpls/>.
- [23] L. De Ghein, *MPLS Fundamental: A Comprehensive Introduction to MPLS (Theory and Practice)*. CISCO Press, November 2006.
- [24] R. Bonica, D. Gan, D. Tappan, and C. Pignataro, "ICMP extensions for multiprotocol label switching," Internet Engineering Task Force, RFC 4950, August 2007.
- [25] T. Fiola and J. Panagos, *This Week: Deploying MPLS*, ser. Junos Networking Technologies Series. Juniper Networks Books, April 2011.
- [26] E. Rosen and Y. Rekhter, "BGP/MPLS IP virtual private networks (VPNs)," Internet Engineering Task Force, RFC 4364, February 2006.
- [27] B. Donnet, P. Raoult, T. Friedman, and M. Crovella, "Efficient algorithms for large-scale topology discovery," in *Proc. ACM SIGMETRICS*, June 2005.
- [28] J.-F. Grailet, F. Tarissan, and B. Donnet, "TreeNET: Discovering and connecting subnets," in *Proc. Traffic Monitoring and Analysis Workshop (TMA)*, April 2016.
- [29] M. P. Vanaubel, Y. J.-J. Pansiot, and B. Donnet, "Mpls tech rep," arXiv, cs.NI 1234, May 2018.
- [30] G. Davila Revelo, M. A. Ricci, B. Donnet, and J. I. Alvarez-Hamelin, "Unveiling the MPLS structure on Internet topology," in *Proc. Traffic Monitoring and Analysis Workshop (TMA)*, April 2016.
- [31] Y. Vanaubel, P. Mérindol, J.-J. Pansiot, and B. Donnet, "MPLS under the microscope: Revealing actual transit path diversity," in *Proc. ACM Internet Measurement Conference (IMC)*, October 2015.
- [32] R. Beverly, A. Berger, and G. Xie, "Primitives for active Internet topology mapping: Toward high-frequency characterization," in *Proc. ACM Internet Measurement Conference (IMC)*, November 2010.
- [33] R. Beverly, "Yarrr'ing the Internet: Randomized high-speed active topology discovery," in *Proc. ACM Internet Measurement Conference (IMC)*, November 2016.
- [34] E. Katz-Bassett, H. Madhyastha, V. Adhikari, C. Scott, J. Sherry, P. van Wesep, A. Krishnamurthy, and T. Anderson, "Reverse traceroute," in *Proc. USENIX Symposium on Networked Systems Design and Implementations (NSDI)*, June 2010, see <https://www.revtr.ccs.neu.edu>.
- [35] R. Sherwood and N. Spring, "Touring the internet in a TCP sidecar," in *Proc. ACM Internet Measurement Conference (IMC)*, October 2006.
- [36] R. Sherwood, A. Bender, and N. Spring, "Discarte: a disjunctive Internet cartographer," in *Proc. ACM SIGCOMM*, August 2008.
- [37] P. Marchetta, W. de Donato, V. Persico, and A. Pescapé, "Experimenting with alternative path tracing solutions," in *Proc. IEEE Symposium on Computers and Communications (ISCC)*, July 2015.
- [38] M. E. Tozal and K. Sarac, "TraceNET: an Internet topology data collector," in *Proc. ACM Internet Measurement Conference (IMC)*, November 2010.
- [39] P. Marchetta and A. Pescapé, "DRAGO: Detecting, quantifying and locating hidden routers in traceroute IP paths," in *Proc. Global Internet Symposium (GI)*, April 2013.
- [40] B. J. Goodchild, Y.-C. Chiu, R. Hansen, H. Lua, M. Calder, M. Luckie, W. Lloyd, D. Choffnes, and E. Katz-Bassett, "The record route option is an option!" in *In Proc. ACM Internet Measurement Conference (IMC)*, November 2017.

# Appendix

This appendix illustrates the validation of TNT through GNS-3 emulations. Multiple configurations have been tested (and even more are proposed on the website<sup>3</sup> and can be setup using the scripts and the data online). Note that we use the version 2.1.5 of GNS3 to export the so-called portable configurations. TNT is able to deal with all those configurations (both in the wild and with the ones emulated in GNS3), making it a pretty robust tool. However, in this report we use another version of our tool to simplify the output. The output of TNT slightly differ but the conclusions are the same.



(a) Cisco topology. PE1 is the Ingress LER, PE2 the Egress LER, the LSP is set up between PE1 and the EH (P3 or PE2). The TNT target (i.e., the argument of `trace_naughty_tunnel()` function – See Listing 1) is the loopback address of CE3.



(b) Juniper topology. PE1 is the Ingress LER, PE2 the Egress LER, the LSP is set up between PE1 and the EH (P3 or PE2). The TNT target (i.e., the argument of `trace_naughty_tunnel()` function – See Listing 1) is the loopback address of CE3.

Fig. 6: Topology used for GNS-3 tests

## I. P2P CIRCUITS

### A. Explicit Tunnels Validation

We first review Explicit tunnels, i.e., tunnels with RFC4950 and `ttl-propagate` enabled (see Sec. III).

In the following, we distinguish Cisco (Appendix I-A1) and Juniper IP topologies (Appendix I-A2) and configurations. In particular, with Cisco configurations, PHP (LSE popped by P3) is distinguished from UHP (LSE popped by Egress LER).

For each case, we provide the configuration of routers as well as the simplified TNT output. Indicators and triggers (see Sec. IV-B) are provided, as well as raw ICMP time-exceeded and ICMP echo-reply TTLs.

1) *Cisco Explicit Configurations*: All configurations presented here were run on the IP topology provided by Fig. 6a.

The first example provides an Explicit tunnel deployed with PHP, under Cisco IOS 15.2. The TNT behavior is the one expected.

#### IOS 15.2 – Explicit PHP

```

1 PE1
2 version 15.2
3 mpls label protocol ldp
4 router bgp 3333
5 redistribute connected
6 redistribute ospf 10
7 neighbor 10.12.0.1 remote-as 3333
8 neighbor 10.12.0.1 next-hop-self
9 neighbor 192.168.8.1 remote-as 1024
10 neighbor 192.168.8.1 next-hop-self
11
12 PE2
13 version 15.2
14 mpls label protocol ldp
15 router bgp 3333
16 redistribute connected
17 redistribute ospf 10
18 neighbor 10.12.0.1 remote-as 3333
19 neighbor 10.12.0.1 next-hop-self
20 neighbor 192.168.2.2 remote-as 2048
21 neighbor 192.168.2.2 next-hop-self
22
23 P1
24 version 15.2
25 mpls label protocol ldp
26 router bgp 3333
27 neighbor 10.12.0.1 remote-as 3333
28
29 P2
30 version 15.2
31 mpls label protocol ldp
32 router bgp 3333
33 neighbor 10.12.0.1 remote-as 3333
34
35 P3
36 version 15.2
37 mpls label protocol ldp
38 router bgp 3333
39 neighbor 10.12.0.1 remote-as 3333

```

#### TNT running over IOS 15.2 – Explicit PHP

```

1 Launching TNT: 192.168.7.1 (192.168.7.1)
2
3 1 left.CE1 (192.168.3.2) <255,255> [frpla = 0][qttl = 1][uturn = 0] 27.083 ms
4 2 left.PE1 (192.168.8.2) <254,254> [frpla = 0][qttl = 1][uturn = 0] 19.895 ms
5 3 left.P1 (10.1.0.2) <247,253> [frpla = 6][qttl = 1][uturn = 6][MPLS LSE | Label : 19 | LSE-TTL : 1] 80.598 ms
6 4 left.P2 (10.2.0.2) <248,252> [frpla = 4][qttl = 2][uturn = 4][MPLS LSE | Label : 20 | LSE-TTL : 1] 69.875 ms
7 5 left.P3 (10.3.0.2) <251,251> [frpla = 0][qttl = 1][uturn = 0][MPLS LSE | Label : 20 | LSE-TTL : 1] 68.98 ms
8 6 left.PE2 (10.4.0.2) <250,250> [frpla = 0][qttl = 1][uturn = 0] 78.17 ms
9 7 left.CE2 (192.168.2.2) <249,249> [frpla = 0][qttl = 1][uturn = 0] 78.957 ms
10 8 192.168.4.2 (192.168.4.2) <248,248> [frpla = 0][qttl = 1][uturn = 0] 110.598 ms

```

The next two configurations illustrate UHP with both IOS 12.4 and IOS 15.2. TNT works as expected and shows two examples of MPLS TTL processing specifically with UHP. With the 12.4 IOS, we see the null label while it is hidden with the 15.2 IOS. In addition, we can see that UHP tunnels show a UTURN signature different from PHP tunnels. This difference results from the way time-exceeded messages are handled by the LSRs. In both cases, the time-exceeded message is forwarded to the EH which replies using its own IP forwarding table. The EH changes depending on the configuration: P3 for PHP (here the EH is the PH), and PE2 for UHP (here the EH is the Egress LER). Indeed, we can see that the UTURN difference disappears at the respective EH.

#### IOS 12.4 – Explicit UHP

```

1 PE1
2 version 12.4
3 mpls label protocol ldp
4 mpls ldp explicit-null
5 router bgp 3333
6 redistribute connected
7 redistribute ospf 10
8 neighbor 10.12.0.1 remote-as 3333

```

```

9 neighbor 10.12.0.1 next-hop-self
10 neighbor 192.168.8.1 remote-as 1024
11 neighbor 192.168.8.1 next-hop-self
12
13 PE2
14 version 12.4
15 mpls label protocol ldp
16 mpls ldp explicit-null
17 router bgp 3333
18 redistribute connected
19 redistribute ospf 10
20 neighbor 10.12.0.1 remote-as 3333
21 neighbor 10.12.0.1 next-hop-self
22 neighbor 192.168.2.2 remote-as 2048
23 neighbor 192.168.2.2 next-hop-self
24
25 P1
26 version 12.4
27 mpls label protocol ldp
28 mpls ldp explicit-null
29 router bgp 3333
30 neighbor 10.12.0.1 remote-as 3333
31
32 P2
33 version 12.4
34 mpls label protocol ldp
35 mpls ldp explicit-null
36 router bgp 3333
37 neighbor 10.12.0.1 remote-as 3333
38
39 P3
40 version 12.4
41 mpls label protocol ldp
42 mpls ldp explicit-null
43 router bgp 3333
44 neighbor 10.12.0.1 remote-as 3333

```

### TNT running over IOS 12.4 – Explicit UHP

```

1 Launching TNT: 192.168.7.1 (192.168.7.1)
2
3 1 left.CE1 (192.168.3.2)  <255,255> [frpla = 0][qttl = 1][uturn = 0] 22.651 ms
4 2 192.168.8.2 (192.168.8.2)  <254,254> [frpla = 0][qttl = 1][uturn = 0] 230.326 ms
5 3 left.P1 (10.1.0.2)  <247,253> [frpla = 6][qttl = 1][uturn = 6][MPLS LSE | Label : 22 | LSE-TTL : 1] 263.686 ms
6 4 left.P2 (10.2.0.2)  <248,252> [frpla = 4][qttl = 2][uturn = 4][MPLS LSE | Label : 22 | LSE-TTL : 1] 358.238 ms
7 5 left.P3 (10.3.0.2)  <249,251> [frpla = 2][qttl = 3][uturn = 2][MPLS LSE | Label : 16 | LSE-TTL : 1] 374.214 ms
8 6 left.PE2 (10.4.0.2)  <250,250> [frpla = 0][qttl = 1][uturn = 0][MPLS LSE | Label : 0 | LSE-TTL : 1] 418.696 ms
9 7 left.CE2 (192.168.2.2)  <249,249> [frpla = 0][qttl = 1][uturn = 0] 655.848 ms
10 8 192.168.4.2 (192.168.4.2)  <248,248> [frpla = 0][qttl = 1][uturn = 0] 513.054 ms

```

### IOS 15.2 – Explicit UHP

```

1 PE1
2 version 15.2
3 mpls label protocol ldp
4 mpls ldp explicit-null
5 router bgp 3333
6 redistribute connected
7 redistribute ospf 10
8 neighbor 10.12.0.1 remote-as 3333
9 neighbor 10.12.0.1 next-hop-self
10 neighbor 192.168.8.1 remote-as 1024
11 neighbor 192.168.8.1 next-hop-self
12
13 PE2
14 version 15.2
15 mpls label protocol ldp
16 mpls ldp explicit-null
17 router bgp 3333
18 redistribute connected
19 redistribute ospf 10
20 neighbor 10.12.0.1 remote-as 3333
21 neighbor 10.12.0.1 next-hop-self
22 neighbor 192.168.2.2 remote-as 2048
23 neighbor 192.168.2.2 next-hop-self
24
25 P1
26 version 15.2
27 mpls label protocol ldp
28 mpls ldp explicit-null
29 router bgp 3333
30 neighbor 10.12.0.1 remote-as 3333
31
32 P2
33 version 15.2

```

```

34 mpls label protocol ldp
35 mpls ldp explicit-null
36 router bgp 3333
37 neighbor 10.12.0.1 remote-as 3333
38
39 P3
40 version 15.2
41 mpls label protocol ldp
42 mpls ldp explicit-null
43 router bgp 3333
44 neighbor 10.12.0.1 remote-as 3333

```

#### TNT running over IOS 15.2 – Explicit UHP

```

1 Launching TNT: 192.168.7.1 (192.168.7.1)
2
3 1 left.CE1 (192.168.3.2) <255,255> [frpla = 0][qttl = 1][uturn = 0] 7.64 ms
4 2 left.PE1 (192.168.8.2) <254,254> [frpla = 0][qttl = 1][uturn = 0] 39.87 ms
5 3 left.P1 (10.1.0.2) <247,253> [frpla = 6][qttl = 1][uturn = 6][MPLS LSE | Label : 19 | LSE-TTL : 1] 100.632 ms
6 4 left.P2 (10.2.0.2) <248,252> [frpla = 4][qttl = 2][uturn = 4][MPLS LSE | Label : 20 | LSE-TTL : 1] 80.453 ms
7 5 left.P3 (10.3.0.2) <249,251> [frpla = 2][qttl = 3][uturn = 2][MPLS LSE | Label : 20 | LSE-TTL : 1] 100.815 ms
8 6 left.PE2 (10.4.0.2) <250,250> [frpla = 0][qttl = 1][uturn = 0] 109.089 ms
9 7 left.CE2 (192.168.2.2) <249,249> [frpla = 0][qttl = 1][uturn = 0] 98.817 ms
10 8 192.168.4.2 (192.168.4.2) <248,248> [frpla = 0][qttl = 1][uturn = 0] 119.842 ms

```

2) *Juniper Explicit Configurations:* All configurations presented here were run on the topology provided by Fig. 6b.

For Explicit tunnels, Juniper Olive and VMX behave the same. We first provide the configuration and TNT output for Explicit tunnels without UTURN effect.

#### VMX – Explicit PHP (default configuration)

```

1 PE1
2 propagate ttl
3
4 PE2
5 propagate ttl
6
7 P1
8 propagate ttl
9
10 P2
11 propagate ttl
12
13 P3
14 propagate ttl

```

#### TNT running over VMX - Explicit PHP (default configuration)

```

1 Launching TNT: 192.168.2.102 (192.168.2.102)
2
3 1 CE1 ( 172.16.0.5) <255,64> [frpla = 0][qttl = 1][uturn = 0] 2.682 ms
4 2 PE1 ( 172.16.0.2) <254,63> [frpla = 0][qttl = 1][uturn = 0] 4.603 ms
5 3 left.P1 (192.168.1.2) <253,62> [frpla = 0][qttl = 1][uturn = 0][MPLS LSE | Label : 299824 | LSE-TTL : 1] 6.362 ms
6 4 left.P2 (192.168.1.6) <252,61> [frpla = 0][qttl = 1][uturn = 0][MPLS LSE | Label : 299792 | LSE-TTL : 1] 8.451 ms
7 5 left.P3 (192.168.1.10) <251,60> [frpla = 0][qttl = 1][uturn = 0][MPLS LSE | Label : 299792 | LSE-TTL : 1] 8.557 ms
8 6 left.PE2 (192.168.1.14) <250,59> [frpla = 0][qttl = 1][uturn = 0] 8.285 ms
9 7 CE2 (192.168.2.2) <249,58> [frpla = 0][qttl = 1][uturn = 0] 8.09 ms
10 8 CE3 (192.168.2.102) <248,57> [frpla = 0][qttl = 1][uturn = 0] 8.142 ms

```

On the contrary to Cisco configuration, Juniper does not exhibit the UTURN effect. When the LSE-TTL of a packet expires, the LSR does not send the ICMP time-exceeded to the EH which then forwards the packets on its own to the probing source, it replies the same with respect to other probes (e.g., echo-request) using its own IP forwarding table if available – resulting in general in a shorter return path (see Sec. IV-B). The configuration must be explicitly stated with the `icmp-tunneling` as provided below.

#### VMX – Explicit PHP (icmp-tunneling configuration)

```

1 PE1
2 propagate ttl
3 icmp-tunneling
4
5 PE2
6 propagate ttl
7 icmp-tunneling
8
9 P1
10 propagate ttl
11 icmp-tunneling
12
13 P2
14 propagate ttl
15 icmp-tunneling
16
17 P3
18 propagate ttl

```

```
19 icmp-tunneling
```

```
TNT running over VMX – Explicit PHP (icmp-tunneling configuration)
1 Launching TNT: 192.168.2.102 (192.168.2.102)
2
3 1 CE1 ( 172.16.0.5) <255,64> [frpla = 0][qttl = 1][uturn = 0] 2.034 ms
4 2 PE1 ( 172.16.0.2) <254,63> [frpla = 0][qttl = 1][uturn = 0] 4.646 ms
5 3 left.P1 (192.168.1.2) <246,62> [frpla = 7][qttl = 1][uturn = 7][MPLS LSE | Label : 299824 | LSE-TTL : 1] 11.424 ms
6 4 left.P2 (192.168.1.6) <247,61> [frpla = 5][qttl = 1][uturn = 5][MPLS LSE | Label : 299824 | LSE-TTL : 1] 7.994 ms
7 5 left.P3 (192.168.1.10) <251,60> [frpla = 0][qttl = 1][uturn = 0][MPLS LSE | Label : 299824 | LSE-TTL : 1] 6.252 ms
8 6 left.PE2 (192.168.1.14) <250,59> [frpla = 0][qttl = 1][uturn = 0] 8.585 ms
9 7 CE2 (192.168.2.2) <249,58> [frpla = 0][qttl = 1][uturn = 0] 9.369 ms
10 8 CE3 (192.168.2.102) <248,57> [frpla = 0][qttl = 1][uturn = 0] 9.232 ms
```

### B. Opaque Tunnels Validation (Cisco only)

Opaque tunnels only occur with Cisco routers, in some particular configuration (see Sec. III for details). The topology used for GNS-3 emulation is the one provided by Fig. 6a. We only show tests for IOS 15.2 as the situation is the same with IOS 12.4. In our example, we were able to reveal the content of the Opaque tunnel through BRPR, on the contrary to in the wild TNT deployment where Opaque tunnels revelation did not work that much (see Sec. VI). We see thus here a difference between theory and practice.

```
IOS 15.2 – Opaque PHP
```

```
1 PE1
2 version 15.2
3 mpls label protocol ldp
4 no propagate-ttl
5 router bgp 3333
6 redistribute connected
7 redistribute ospf 10
8 neighbor 10.12.0.1 remote-as 3333
9 neighbor 192.168.8.1 remote-as 1024
10
11 PE2
12 version 15.2
13 mpls label protocol ldp
14 no propagate-ttl
15 router bgp 3333
16 redistribute connected
17 redistribute ospf 10
18 neighbor 10.12.0.1 remote-as 3333
19 neighbor 192.168.6.1 remote-as 2048
20 neighbor 192.168.6.1 ebgp-multihop 2
21
22 P1
23 version 15.2
24 mpls label protocol ldp
25 no propagate-ttl
26 router bgp 3333
27 neighbor 10.12.0.1 remote-as 3333
28
29 P2
30 version 15.2
31 mpls label protocol ldp
32 no propagate-ttl
33 router bgp 3333
34 neighbor 10.12.0.1 remote-as 3333
35
36 P3
37 version 15.2
38 mpls label protocol ldp
39 no propagate-ttl
40 router bgp 3333
41 neighbor 10.12.0.1 remote-as 3333
```

```
TNT running over IOS 15.2 – Opaque PHP
```

```
1 Launching TNT: 192.168.7.1 (192.168.7.1)
2
3 1 left.CE1 (192.168.3.2) <255,255> [frpla = 0][qttl = 1][uturn = 0] 25.164 ms
4 2 left.PE1 (192.168.8.2) <254,254> [frpla = 0][qttl = 1][uturn = 0] 40.06 ms
5
6 OPAQUE | Length estimation : 3 | Revealed : 3 (difference : 0)
7 2.1 [REVEALED] left.P1 (10.1.0.2) <253,253> [frpla = 0][qttl = 1][uturn = 0] 40.008 ms - step 2
8 2.2 [REVEALED] left.P2 (10.2.0.2) <252,252> [frpla = 0][qttl = 1][uturn = 0] 40.058 ms - step 1
9 2.3 [REVEALED] left.P3 (10.3.0.2) <251,251> [frpla = 0][qttl = 1][uturn = 0] 90.301 ms - step 0
10
11 3 left.PE2 (10.4.0.2) <250,250> [frpla = 3][qttl = 1][uturn = 0][MPLS LSE | Label : 16 | LSE-TTL : 252] 110.408 ms
12 4 left.CE2 (192.168.2.2) <250,250> [frpla = 2][qttl = 1][uturn = 0] 80.195 ms
13 5 192.168.4.2 (192.168.4.2) <250,250> [frpla = 1][qttl = 1][uturn = 0] 132.331 ms
```

### C. Invisible Tunnels Validation

This section discusses Invisible tunnels, i.e., tunnels with the `no-ttl-propagate` option enabled (see Sec. III).

We do a distinction between Cisco (Appendix I-C1) and Juniper configurations (Appendix I-C2). PHP (LSE popped by P3) is also distinguished from UHP (LSE popped by Egress LER).

For each case, we provide the configuration of routers as well as the TNT output. Indicators and triggers (see Sec. IV-B) are provided, as well as ICMP `time-exceeded` and ICMP `echo-reply` TTLs.

1) *Invisible Cisco Configurations*: All configurations presented here were run on the topology provided by Fig. 6a.

#### IOS 15.2 – Invisible PHP

```

1 PE1
2 version 15.2
3 mpls label protocol ldp
4 no propagate-ttl
5 router bgp 3333
6 redistribute connected
7 redistribute ospf 10
8 neighbor 10.12.0.1 remote-as 3333
9 neighbor 10.12.0.1 next-hop-self
10 neighbor 192.168.8.1 remote-as 1024
11 neighbor 192.168.8.1 next-hop-self
12
13 PE2
14 version 15.2
15 mpls label protocol ldp
16 no propagate-ttl
17 router bgp 3333
18 redistribute connected
19 redistribute ospf 10
20 neighbor 10.12.0.1 remote-as 3333
21 neighbor 10.12.0.1 next-hop-self
22 neighbor 192.168.2.2 remote-as 2048
23 neighbor 192.168.2.2 next-hop-self
24
25 P1
26 version 15.2
27 mpls label protocol ldp
28 no propagate-ttl
29 router bgp 3333
30 neighbor 10.12.0.1 remote-as 3333
31
32 P2
33 version 15.2
34 mpls label protocol ldp
35 no propagate-ttl
36 router bgp 3333
37 neighbor 10.12.0.1 remote-as 3333
38
39 P3
40 version 15.2
41 mpls label protocol ldp
42 no propagate-ttl
43 router bgp 3333
44 neighbor 10.12.0.1 remote-as 3333

```

#### TNT running over IOS 15.2 – Invisible PHP

```

1 Launching TNT: 192.168.7.1 (192.168.7.1)
2
3 1 left.CE1 (192.168.3.2) <255,255> [frpla = 0][qttl = 1][uturn = 0] 7.52 ms
4 2 left.PE1 (192.168.8.2) <254,254> [frpla = 0][qttl = 1][uturn = 0] 29.927 ms
5
6 FRPLA | Length estimation : 3 | Revealed : 3 (difference : 0)
7 2.1 [REVEALED] left.P1 (10.1.0.2) <253,253> [frpla = 0][qttl = 1][uturn = 0] 50.051 ms - step 2
8 2.2 [REVEALED] left.P2 (10.2.0.2) <252,252> [frpla = 0][qttl = 1][uturn = 0] 60.102 ms - step 1
9 2.3 [REVEALED] left.P3 (10.3.0.2) <251,251> [frpla = 0][qttl = 1][uturn = 0] 59.876 ms - step 0
10
11 3 left.PE2 (10.4.0.2) <250,250> [frpla = 3][qttl = 1][uturn = 0] 80.38 ms
12 4 left.CE2 (192.168.2.2) <250,250> [frpla = 2][qttl = 1][uturn = 0] 69.89 ms
13 5 192.168.4.2 (192.168.4.2) <250,250> [frpla = 1][qttl = 1][uturn = 0] 99.833 ms

```

The configuration for running standard Cisco Invisible UHP tunnels is provided below. Such a configuration might be revealed through BRPR thanks to the `DUP_IP` trigger.

#### IOS 15.2 – Invisible UHP

```

1 PE1
2 version 15.2
3 mpls label protocol ldp
4 no propagate-ttl
5 mpls ldp explicit-null
6 router bgp 3333
7 redistribute connected
8 redistribute ospf 10

```

```

9 neighbor 10.12.0.1 remote-as 3333
10 neighbor 10.12.0.1 next-hop-self
11 neighbor 192.168.8.1 remote-as 1024
12 neighbor 192.168.8.1 next-hop-self
13
14 PE2
15 version 15.2
16 mpls label protocol ldp
17 no propagate-ttl
18 mpls ldp explicit-null
19 router bgp 3333
20 redistribute connected
21 redistribute ospf 10
22 neighbor 10.12.0.1 remote-as 3333
23 neighbor 10.12.0.1 next-hop-self
24 neighbor 192.168.2.2 remote-as 2048
25 neighbor 192.168.2.2 next-hop-self
26
27 P1
28 version 15.2
29 mpls label protocol ldp
30 no propagate-ttl
31 mpls ldp explicit-null
32 router bgp 3333
33 neighbor 10.12.0.1 remote-as 3333
34
35 P2
36 version 15.2
37 mpls label protocol ldp
38 no propagate-ttl
39 mpls ldp explicit-null
40 router bgp 3333
41 neighbor 10.12.0.1 remote-as 3333
42
43 P3
44 version 15.2
45 mpls label protocol ldp
46 no propagate-ttl
47 mpls ldp explicit-null
48 router bgp 3333
49 neighbor 10.12.0.1 remote-as 3333

```

#### TNT running over IOS 15.2 – Invisible UHP

```

1 Launching TNT: 192.168.7.1 (192.168.7.1)
2
3 1 left.CE1 (192.168.3.2) <255,255> [frpla = 0][qttl = 1][uturn = 0] 3.157 ms
4 2 left.PE1 (192.168.8.2) <254,254> [frpla = 0][qttl = 1][uturn = 0] 29.92 ms
5
6 Duplicate IP (Egress : 192.168.2.2) | Length estimation : 1 | Revealed : 4 (difference : 3)
7 2.1 [REVEALED] left.P1 (10.1.0.2) <253,253> [frpla = 0][qttl = 1][uturn = 0] 50.043 ms - step 4 (Buddy used)
8 2.2 [REVEALED] left.P2 (10.2.0.2) <253,253> [frpla = 0][qttl = 1][uturn = 0] 49.778 ms - step 3 (Buddy used)
9 2.3 [REVEALED] left.P3 (10.3.0.2) <253,253> [frpla = 0][qttl = 1][uturn = 0] 69.834 ms - step 2 (Buddy used)
10 2.4 [REVEALED] left.PE2 (10.4.0.2) <253,253> [frpla = 0][qttl = 1][uturn = 0] 80.594 ms - step 1 (Buddy used)
11
12 3 left.CE2 (192.168.2.2) <252,252> [frpla = 1][qttl = 1][uturn = 0] 80.08 ms
13 4 left.CE2 (192.168.2.2) <252,252> [frpla = 0][qttl = 1][uturn = 0] 89.891 ms
14 5 192.168.4.2 (192.168.4.2) <251,251> [frpla = 0][qttl = 1][uturn = 0] 107.579 ms

```

With Cisco routers, it is possible to mimic an Invisible UHP tunnel with a Juniper per loopback configuration (i.e., by filtering addresses to /32 border prefixes), meaning that the tunnel content might be revealed through DPR, thanks to the DUP\_IP trigger. Such a configuration is achieved with the `allocate global host-routes` command.

#### IOS 15.2 – Invisible UHP (allocate global host route configuration)

```

1 PE1
2 version 15.2
3 mpls label protocol ldp
4 no propagate-ttl
5 mpls ldp explicit-null
6 mpls ldp label
7 allocate global host-routes
8 router bgp 3333
9 redistribute connected
10 redistribute ospf 10
11 neighbor 10.12.0.1 remote-as 3333
12 neighbor 10.12.0.1 next-hop-self
13 neighbor 192.168.8.1 remote-as 1024
14 neighbor 192.168.8.1 next-hop-self
15
16 PE2
17 version 15.2
18 mpls label protocol ldp
19 no propagate-ttl
20 mpls ldp explicit-null

```

```

21 mpls ldp label
22  allocate global host-routes
23 router bgp 3333
24 redistribute connected
25 redistribute ospf 10
26 neighbor 10.12.0.1 remote-as 3333
27 neighbor 10.12.0.1 next-hop-self
28 neighbor 192.168.2.2 remote-as 2048
29 neighbor 192.168.2.2 next-hop-self
30
31 P1
32 version 15.2
33 mpls label protocol ldp
34 no propagate-ttl
35 mpls ldp explicit-null
36 mpls ldp label
37  allocate global host-routes
38 router bgp 3333
39 neighbor 10.12.0.1 remote-as 3333
40
41 P2
42 version 15.2
43 mpls label protocol ldp
44 no propagate-ttl
45 mpls ldp explicit-null
46 mpls ldp label
47  allocate global host-routes
48 router bgp 3333
49 neighbor 10.12.0.1 remote-as 3333
50
51 P3
52 version 15.2
53 mpls label protocol ldp
54 no propagate-ttl
55 mpls ldp explicit-null
56 mpls ldp label
57  allocate global host-routes
58 router bgp 3333
59 neighbor 10.12.0.1 remote-as 3333

```

#### TNT running over IOS 15.2 – Invisible UHP (allocate global host route configuration)

```

1 Launching TNT: 192.168.7.1 (192.168.7.1)
2
3 1 left.CE1 (192.168.3.2) <255,255> [frpla = 0][qttl = 1][uturn = 0] 8.091 ms
4 2 left.PE1 (192.168.8.2) <254,254> [frpla = 0][qttl = 1][uturn = 0] 39.867 ms
5
6 Duplicate IP (Egress : 10.1.0.2) | Length estimation : 1 | Revealed : 4 (difference : 3)
7 2.1 [REVEALED] left.P1 (10.1.0.2) <253,253> [frpla = 0][qttl = 1][uturn = 0] 39.788 ms - step 2
8 2.2 [REVEALED] left.P2 (10.2.0.2) <253,253> [frpla = 0][qttl = 1][uturn = 0] 49.573 ms - step 2
9 2.3 [REVEALED] left.P3 (10.3.0.2) <253,253> [frpla = 0][qttl = 1][uturn = 0] 70.094 ms - step 2
10 2.4 [REVEALED] left.PE2 (10.4.0.2) <253,253> [frpla = 0][qttl = 1][uturn = 0] 89.171 ms - step 1 ( Buddy used )
11
12 3 left.CE2 (192.168.2.2) <252,252> [frpla = 1][qttl = 1][uturn = 0] 120.546 ms
13 4 left.CE2 (192.168.2.2) <252,252> [frpla = 0][qttl = 1][uturn = 0] 89.892 ms
14 5 192.168.4.2 (192.168.4.2) <251,251> [frpla = 0][qttl = 1][uturn = 0] 117.301 ms

```

It is also possible to build Invisible UHP tunnel in which the buddy mechanism is not necessary (as we discover in the wild). Simply running BRPR will make the tunnel content visible. This configuration might be achieved with the `ip access-list` command to enable Ultimate Hop Popping for external destinations only:

#### IOS 15.2 – Invisible UHP (mpls ldp explicit-null [for prefix-acl] configuration)

```

1 PE1
2 version 15.2
3 mpls label protocol ldp
4 no propagate-ttl
5 router bgp 3333
6 redistribute connected
7 redistribute ospf 10
8 neighbor 10.12.0.1 remote-as 3333
9 neighbor 10.12.0.1 next-hop-self
10 neighbor 192.168.8.1 remote-as 1024
11 neighbor 192.168.8.1 next-hop-self
12
13 PE2
14 version 15.2
15 mpls label protocol ldp
16 no propagate-ttl
17 mpls ldp explicit-null for BRPR-wo-buddy
18 router bgp 3333
19 redistribute connected
20 redistribute ospf 10
21 neighbor 10.12.0.1 remote-as 3333
22 neighbor 10.12.0.1 next-hop-self

```

```

23 neighbor 192.168.2.2 remote-as 2048
24 neighbor 192.168.2.2 next-hop-self
25 ip access-list standard BRPR-wo-buddy
26 permit 10.9.0.1
27 deny any
28
29 P1
30 version 15.2
31 mpls label protocol ldp
32 no propagate-ttl
33 router bgp 3333
34 neighbor 10.12.0.1 remote-as 3333
35
36 P2
37 version 15.2
38 mpls label protocol ldp
39 no propagate-ttl
40 router bgp 3333
41 neighbor 10.12.0.1 remote-as 3333
42
43 P3
44 version 15.2
45 mpls label protocol ldp
46 no propagate-ttl
47 router bgp 3333
48 neighbor 10.12.0.1 remote-as 3333

```

#### TNT running over IOS 15.2 – Invisible UHP (mpls ldp explicit-null [for prefix-acl] configuration)

```

1 Launching TNT: 192.168.7.1 (192.168.7.1)
2
3 1 192.168.3.2 (192.168.3.2) <255,255> [frpla = 0][qttl = 1][uturn = 0] 7.299 ms
4 2 192.168.8.2 (192.168.8.2) <254,254> [frpla = 0][qttl = 1][uturn = 0] 14.921 ms
5
6 Duplicate IP (Egress : 10.4.0.2) | Length estimation : 3 | Revealed : 4 (difference : 1)
7 2.1 [REVEALED] 10.1.0.2 (10.1.0.2) <253,253> [frpla = 0][qttl = 1][uturn = 0] 36.443 ms - step 3
8 2.2 [REVEALED] 10.2.0.2 (10.2.0.2) <252,252> [frpla = 0][qttl = 1][uturn = 0] 35.879 ms - step 2
9 2.3 [REVEALED] 10.3.0.2 (10.3.0.2) <251,251> [frpla = 0][qttl = 1][uturn = 0] 66.288 ms - step 1
10 2.4 [REVEALED] 10.4.0.2 (10.4.0.2) <250,250> [frpla = 0][qttl = 1][uturn = 0] 64.19 ms - step 0
11
12 3 CE2 (192.168.2.2) <250,250> [frpla = 3][qttl = 1][uturn = 0] 116.643 ms
13 4 CE2 (192.168.2.2) <250,250> [frpla = 2][qttl = 1][uturn = 0] 99.93 ms
14 5 192.168.4.2 (192.168.4.2) <250,250> [frpla = 1][qttl = 1][uturn = 0] 94.185 ms

```

2) *Juniper Invisible Configurations*: All configurations presented here were run on the topology provided by Fig. 6b.

Juniper, with Olive OS, does not apply the MIN(IP-TTL, LSE-TTL) at the exit of the MPLS cloud. As such, the FRPLA trigger does not provide the return tunnel length but is equal to 1 because the ingress LER process the incoming IP TTL in a distinct way with respect to the origin of the packet (locally generated or not). Invisible PHP tunnel can, then, be revealed through DPR. Juniper LSR can be configured as followed:

#### JunOS Olive – Invisible PHP

```

1 PE1
2 no-propagate-ttl
3
4 PE2
5 no-propagate-ttl
6
7 P1
8 no-propagate-ttl
9
10 P2
11 no-propagate-ttl
12
13 P3
14 no-propagate-ttl

```

#### TNT running over JunOS Olive – Invisible PHP

```

1 Launching TNT: 192.168.2.102 (192.168.2.102)
2
3 1 CE1 ( 172.16.0.5) <255,64> [frpla = 0][qttl = 1][uturn = 0] 0.638 ms
4 2 PE1 ( 172.16.0.2) <254,63> [frpla = 0][qttl = 1][uturn = 0] 1.898 ms
5
6 FRPLA | Length estimation : 1 | Revealed : 3 (difference : 2)
7 2.1 [REVEALED] left.P1 (192.168.1.2) <253,62> [frpla = 0][qttl = 1][uturn = 0] 3.039 ms - step 0
8 2.2 [REVEALED] left.P2 (192.168.1.6) <252,61> [frpla = 0][qttl = 1][uturn = 0] 3.951 ms - step 0
9 2.3 [REVEALED] left.P3 (192.168.1.10) <252,61> [frpla = 0][qttl = 1][uturn = 0] 4.906 ms - step 0
10
11 3 left.PE2 (192.168.1.14) <252,61> [frpla = 1][qttl = 1][uturn = 0] 7.043 ms
12 4 CE2 (192.168.2.2) <252,61> [frpla = 0][qttl = 1][uturn = 0] 6.891 ms
13 5 CE3 (192.168.2.102) <251,60> [frpla = 0][qttl = 1][uturn = 0] 8.978 ms

```

On the contrary to Olive, VMX applies the MIN(IP-TTL, LSE-TTL) function. As such, the behavior observed is the theoretical one. It

is worth noting that configuring Juniper VMX for Invisible MPLS tunnels is identical than with Olive. Invisible tunnels are, now, revealed through DPR, with the RTRA trigger.

#### JunOS VMX – Invisible PHP

```

1 PE1
2 no-propagate-ttl
3
4 PE2
5 no-propagate-ttl
6
7 P1
8 no-propagate-ttl
9
10 P2
11 no-propagate-ttl
12
13 P3
14 no-propagate-ttl

```

#### TNT running over JunOS VMX – Invisible PHP

```

1 Launching TNT: 192.168.2.102 (192.168.2.102)
2
3 1 CE1 ( 172.16.0.5) <255,64> [frpla = 0][qttl = 1][uturn = 0] 0.96 ms
4 2 PE1 ( 172.16.0.2) <254,63> [frpla = 0][qttl = 1][uturn = 0] 1.66 ms
5
6 RTRA | Length estimation : 3 | Revealed : 3 (difference : 0)
7 2.1 [REVEALED] left.P1 (192.168.1.2) <253,62> [frpla = 0][qttl = 1][uturn = 0] 8.8 ms - step 0
8 2.2 [REVEALED] left.P2 (192.168.1.6) <252,62> [frpla = 0][qttl = 1][uturn = 0] 2.134 ms - step 0
9 2.3 [REVEALED] left.P3 (192.168.1.10) <251,62> [frpla = 0][qttl = 1][uturn = 0] 3.352 ms - step 0
10
11 3 left.PE2 (192.168.1.14) <250,62> [frpla = 3][rtl = 3(3)][qttl = 1][uturn = 3] 4.569 ms
12 4 CE2 (192.168.2.2) <250,61> [frpla = 2][rtl = 2(-1)][qttl = 1][uturn = 2] 4.625 ms
13 5 CE3 (192.168.2.102) <250,60> [frpla = 1][rtl = 1(-1)][qttl = 1][uturn = 1] 4.355 ms

```

#### D. Corner Cases: Heterogeneous Propagation Configuration

This section discusses corner cases, i.e., unlikely configurations that may arise when MPLS is not homogeneously configured throughout the tunnel. TNT, like traceroute, cannot deal with those situations, but these abnormal shiftings have not been clearly encountered in practice.

1) *Cisco Jumpy Configurations*: The following Cisco configuration (for IOS 15.2) is supposed to build an UHP Invisible tunnel. However, on the contrary to the configuration provided in Appendix I-C1, the management of LSE-TTL is heterogeneous over the tunnel. Indeed, in this case, the Ingress LER is not configured with the `no-ttl-propagate` (on the contrary to the Egress LER and other routers in the tunnel). As such, the `MIN(IP-TTL, LSE-TTL)` operation is not – systematically – applied on the Egress while it is expected to be from the Ingress. The EH assumes that the propagation configuration is homogeneous among LERs, which is not the case here. Therefore, the Egress LER will use the IP-TTL instead of the LSE-TTL when popping the LSE. As consequence, and as shown by the TNT output, we observe that

- 1) the MPLS tunnel is actually Explicit;
- 2) a number of hops equal to the tunnel length after the MPLS tunnel are missing (here, only CE2 is missing as the platform is too short – see Fig. 6a for the Cisco topology we use), leading to a so-called *jump* effect.

We call such a configuration *Explicit Jump* and it can be observed in the qTTL of the last hop (2 instead of one plus the skipped hop).

#### IOS 15.2 – Explicit Jump (heterogeneous configuration)

```

1 PE1
2 version 15.2
3 mpls label protocol ldp
4 mpls ldp explicit-null
5 router bgp 3333
6 redistribute connected
7 redistribute ospf 10
8 neighbor 10.12.0.1 remote-as 3333
9 neighbor 10.12.0.1 next-hop-self
10 neighbor 192.168.8.1 remote-as 1024
11 neighbor 192.168.8.1 next-hop-self
12
13 PE2
14 version 15.2
15 mpls label protocol ldp
16 no propagate-ttl
17 mpls ldp explicit-null
18 router bgp 3333
19 redistribute connected
20 redistribute ospf 10
21 neighbor 10.12.0.1 remote-as 3333
22 neighbor 10.12.0.1 next-hop-self
23 neighbor 192.168.2.2 remote-as 2048
24 neighbor 192.168.2.2 next-hop-self
25
26 P1

```

```

27 version 15.2
28 mpls label protocol ldp
29 no propagate-ttl
30 mpls ldp explicit-null
31 router bgp 3333
32 neighbor 10.12.0.1 remote-as 3333
33
34 P2
35 version 15.2
36 mpls label protocol ldp
37 no propagate-ttl
38 mpls ldp explicit-null
39 router bgp 3333
40 neighbor 10.12.0.1 remote-as 3333
41
42 P3
43 version 15.2
44 mpls label protocol ldp
45 no propagate-ttl
46 mpls ldp explicit-null
47 router bgp 3333
48 neighbor 10.12.0.1 remote-as 3333

```

#### TNT running over IOS 15.2 – Explicit Jump (heterogeneous configuration)

```

1 Launching TNT: 192.168.7.1 (192.168.7.1)
2
3 1 left.CE1 (192.168.3.2) <255,255> [frpla = 0][qttl = 1][uturn = 0] 8.407 ms
4 2 left.PE1 (192.168.8.2) <254,254> [frpla = 0][qttl = 1][uturn = 0] 29.477 ms
5 3 left.P1 (10.1.0.2) <250,253> [frpla = 3][qttl = 1][uturn = 3][MPLS LSE | Label : 19 | LSE-TTL : 1] 79.929 ms
6 4 left.P2 (10.2.0.2) <250,252> [frpla = 2][qttl = 2][uturn = 2][MPLS LSE | Label : 20 | LSE-TTL : 1] 80.573 ms
7 5 left.P3 (10.3.0.2) <250,251> [frpla = 1][qttl = 3][uturn = 1][MPLS LSE | Label : 20 | LSE-TTL : 1] 109.577 ms
8 6 left.PE2 (10.4.0.2) <250,250> [frpla = 0][qttl = 1][uturn = 0] 79.766 ms
9 7 192.168.4.2 (192.168.4.2) <250,250> [frpla = -1][qttl = 2][uturn = 0] 109.357 ms

```

2) *Juniper Jumpy Configurations*: In the fashion of Cisco, Juniper with the Olive OS (this is not possible with VMX) allows to configure an Explicit Jump tunnel with PHP. The configuration provided below shows such an MPLS tunnel. The EH is configured with the `no-ttl-propagate` option, while other routers are configured with `ttl-propagate`. As such, P3 will not apply the `MIN(IP-TTL, LSE-TTL)` when popping the label, leading so to a jump effect that is nearly as long as the tunnel itself (the Egress LER and CE2 are missing plus the qTTL at 2 on the last hop).

#### Olive – Explicit Jump (heterogeneous configuration)

```

1 PE1
2 propagate ttl
3
4 PE2
5 propagate ttl
6
7 P1
8 propagate ttl
9
10 P2
11 propagate-ttl
12
13 P3
14 no-propagate-ttl

```

#### TNT running over Olive – Explicit (heterogeneous configuration)

```

1 Launching TNT: 192.168.2.102 (192.168.2.102)
2
3 1 CE1 (172.16.0.5) <255,64> [frpla = 0][qttl = 1][uturn = 0] 0.622 ms
4 2 PE1 (172.16.0.2) <254,63> [frpla = 0][qttl = 1][uturn = 0] 1.749 ms
5 3 left.P1 (192.168.1.2) <253,62> [frpla = 0][qttl = 1][uturn = 0][MPLS LSE | Label : 299824 | LSE-TTL : 1] 2.799 ms
6 4 left.P2 (192.168.1.6) <252,252> [frpla = 0][qttl = 1][uturn = 0][MPLS LSE | Label : 299792 | LSE-TTL : 1] 3.725 ms
7 5 left.P3 (192.168.1.10) <251,251> [frpla = 0][qttl = 1][uturn = 0][MPLS LSE | Label : 299776 | LSE-TTL : 1] 7.784 ms
8 6 CE3 (192.168.2.102) <248,57> [frpla = 2][qttl = 2][uturn = 0] 8.884 ms

```

The last configuration is Juniper Olive with an *Invisible Jump* configuration. This is somewhat equivalent to the Explicit Jump but for Invisible tunnels. In that case, when P3 (PHP is configured) will pop the LSE, it will not apply the `MIN(IP-TTL, LSE-TTL)`. As a result, TNT will see the Ingress LER (PE1) and several hops after P3 will be missed (Egress LER and CE2). The tunnel is invisible and triggers do not work. One can notice a qTTL of 250 on the last hop of our platform: it means that `traceroute` can miss an entire path of 255 minus the length of the tunnel!

#### Olive – Invisible Jump configuration (heterogeneous configuration)

```

1 PE1
2 no-propagate ttl
3
4 PE2
5 propagate ttl
6

```

```

7 P1
8 no-propagate ttl
9
10 P2
11 no-propagate-ttl
12
13 P3
14 propagate-ttl

```

#### TNT running over Olive – Invisible Jump (heterogeneous configuration)

```

1 Launching TNT: 192.168.2.102 (192.168.2.102)
2   1 CE1 ( 172.16.0.5) <255,64> [frpla = 0][qttl = 1][uturn = 0] 0.515 ms
3   2 PE1 ( 172.16.0.2) <254,63> [frpla = 0][qttl = 1][uturn = 0] 1.712 ms
4   3 CE3 (192.168.2.102) <251,60> [frpla = 2][qttl = 250][uturn = 0] 8.553 ms

```

### E. RSVP-TE (All Invisible)

In this section, we introduce PHP and UHP invisible configurations with both Cisco and Juniper OSes but considering RSVP-TE instead of LDP. There is no much to say as all the resulting behaviors are the same as with LDP configurations described previously. The conclusion is then simple: TNT is able to reveal TE tunnels as LDP ones whatever the OS and the popping configuration is (i.e. PHP or UHP).

#### 1) Cisco Config – PHP:

##### cisco IOS 15.2 – RSVP-TE PHP

```

1 PE1
2 version 15.2
3 no propagate-ttl
4 mpls traffic-eng tunnels
5
6 interface Tunnel0
7 ip unnumbered Loopback0
8 tunnel mode mpls traffic-eng
9 tunnel destination 10.9.0.1
10 tunnel mpls traffic-eng autoroute announce
11 tunnel mpls traffic-eng priority 1 1
12 tunnel mpls traffic-eng bandwidth 500
13 tunnel mpls traffic-eng path-option 1 dynamic
14
15 PE2
16 version 15.2
17 no propagate-ttl
18 mpls traffic-eng tunnels
19
20 P1
21 version 15.2
22 no propagate-ttl
23 mpls traffic-eng tunnels
24
25 P2
26 version 15.2
27 no propagate-ttl
28 mpls traffic-eng tunnels
29
30 P3
31 version 15.2
32 no propagate-ttl
33 mpls traffic-eng tunnels

```

##### TNT running over 15.2 – RSVP-TE PHP

```

1 Launching TraceTunnel: 192.168.7.1 (192.168.7.1)
2
3 1 192.168.3.2 (192.168.3.2) <255,255> [ frpla = 0 ][ qttl = 1 ][ uturn = 0 ]
4 2 192.168.8.2 (192.168.8.2) <254,254> [ frpla = 0 ][ qttl = 1 ][ uturn = 0 ]
5
6   FRPLA | Length estimation : 3 | Revealed : 3 (difference : 0)
7   2.1 [REVEALED] 10.1.0.2 ( 10.1.0.2) <253,253> [ frpla = 0 ][ qttl = 1 ][ uturn = 0 ] - step 2
8   2.2 [REVEALED] 10.2.0.2 ( 10.2.0.2) <252,252> [ frpla = 0 ][ qttl = 1 ][ uturn = 0 ] - step 1
9   2.3 [REVEALED] 10.3.0.2 ( 10.3.0.2) <251,251> [ frpla = 0 ][ qttl = 1 ][ uturn = 0 ] - step 0
10
11 3 10.4.0.2 ( 10.4.0.2) <250,250> [ frpla = 3 ][ qttl = 1 ][ uturn = 0 ]
12 4 CE2 (192.168.2.2) <250,250> [ frpla = 2 ][ qttl = 1 ][ uturn = 0 ]
13 5 192.168.4.2 (192.168.4.2) <250,250> [ frpla = 1 ][ qttl = 1 ][ uturn = 0 ]

```

The tunnel is easily revealed as for LDP tunnels.

#### 2) Juniper Config – PHP:

##### Juniper – RSVP-TE PHP

```

1 PE1
2 protocols {
3   rsvp {

```

```

4     tunnel-services;
5     interface all;
6   }
7   mpls {
8     no-propagate-ttl;
9     label-switched-path PE1-to-PE2 {
10       to 192.168.1.105;
11     }
12     interface all;
13   }
14   ospf {
15     traffic-engineering {
16       shortcuts lsp-metric-into-summary;
17     }
18   }
19 }
20
21 PE2
22 protocols {
23   rsvp {
24     tunnel-services;
25     interface all;
26   }
27   mpls {
28     no-propagate-ttl;
29     icmp-tunneling;
30     label-switched-path PE2-toPE1 {
31       to 192.168.1.101;
32     }
33     interface all;
34     interface ge-0/0/2.0;
35   }
36   ospf {
37     traffic-engineering;
38   }
39 }
40
41 P1
42 protocols {
43   rsvp {
44     interface all;
45   }
46   mpls {
47     no-propagate-ttl;
48     interface ge-0/0/1.0;
49     interface ge-0/0/3.0;
50   }
51   ospf {
52     traffic-engineering;
53   }
54 }
55
56 P2
57 protocols {
58   rsvp {
59     interface all;
60   }
61   mpls {
62     no-propagate-ttl;
63     interface ge-0/0/1.0;
64     interface ge-0/0/3.0;
65   }
66   ospf {
67     traffic-engineering;
68   }
69 }
70
71 P3
72 protocols {
73   rsvp {
74     interface all;
75   }
76   mpls {
77     no-propagate-ttl;
78     interface ge-0/0/1.0;
79     interface ge-0/0/3.0;
80   }
81   ospf {
82     traffic-engineering;
83   }
84 }

```

## TNT running over Juniper – RSVP-TE PHP

```

1 Launching TraceTunnel: 192.168.2.102 (192.168.2.102)
2 1 CE1 ( 172.16.0.5) <255,64> [ frpla = 0 ][ rtla = 0(0) ][ qttl = 1 ][ uturn = 0 ]
3 2 PE1 ( 172.16.0.2) <254,63> [ frpla = 0 ][ rtla = 0(0) ][ qttl = 1 ][ uturn = 0 ]
4
5 RTL | Length estimation : 3 | Revealed : 3 (difference : 0)
6 2.1 [REVEALED] left.P1 (192.168.1.2) <253,62> [ frpla = 0 ][ rtla = 0(0) ][ qttl = 1 ][ uturn = 0 ] - step 0
7 2.2 [REVEALED] left.P2 (192.168.1.6) <252,61> [ frpla = 0 ][ rtla = 0(0) ][ qttl = 1 ][ uturn = 0 ] - step 0
8 2.3 [REVEALED] left.P3 (192.168.1.10) <251,60> [ frpla = 0 ][ rtla = 0(0) ][ qttl = 1 ][ uturn = 0 ] - step 0
9
10 3 left.PE2 (192.168.1.14) <250,62> [ frpla = 3 ][ rtla = 3(3) ][ qttl = 1 ][ uturn = 3 ]
11 4 CE2 (192.168.2.2) <250,61> [ frpla = 2 ][ rtla = 0(-1) ][ qttl = 1 ][ uturn = 0 ]
12 5 CE3 (192.168.2.102) <250,60> [ frpla = 1 ][ rtla = 0(0) ][ qttl = 1 ][ uturn = 0 ]

```

## 3) Cisco Config – UHP:

## cisco IOS 15.2 – RSVP-TE UHP

```

1 PE1
2 version 15.2
3 no propagate-ttl
4 mpls traffic-eng tunnels
5
6 interface Tunnel0
7 ip unnumbered Loopback0
8 tunnel mode mpls traffic-eng
9 tunnel destination 10.9.0.1
10 tunnel mpls traffic-eng autoroute announce
11 tunnel mpls traffic-eng priority 1 1
12 tunnel mpls traffic-eng bandwidth 500
13 tunnel mpls traffic-eng path-option 1 dynamic
14
15 PE2
16 version 15.2
17 no propagate-ttl
18 mpls traffic-eng tunnels
19 mpls ldp explicit-null
20
21 P1
22 version 15.2
23 no propagate-ttl
24 mpls traffic-eng tunnels
25
26 P2
27 version 15.2
28 no propagate-ttl
29 mpls traffic-eng tunnels
30
31 P3
32 version 15.2
33 no propagate-ttl
34 mpls traffic-eng tunnels
35 mpls traffic-eng signalling interpret explicit-null verbatim

```

## TNT running over 15.2 – RSVP-TE UHP

```

1 Launching TraceTunnel: 192.168.7.1 (192.168.7.1)
2 1 192.168.3.2 (192.168.3.2) <255,255> [ frpla = 0 ][ qttl = 1 ][ uturn = 0 ]
3 2 192.168.8.2 (192.168.8.2) <254,254> [ frpla = 0 ][ qttl = 1 ][ uturn = 0 ]
4
5 Duplicate IP (Egress : 10.4.0.2) | Length estimation : 3 | Revealed : 4 (difference : 1)
6 2.1 [REVEALED] 10.1.0.2 ( 10.1.0.2) <253,253> [ frpla = 0 ][ qttl = 1 ][ uturn = 0 ] - step 4
7 2.2 [REVEALED] 10.2.0.2 ( 10.2.0.2) <252,252> [ frpla = 0 ][ qttl = 1 ][ uturn = 0 ] - step 3
8 2.3 [REVEALED] 10.3.0.2 ( 10.3.0.2) <251,251> [ frpla = 0 ][ qttl = 1 ][ uturn = 0 ] - step 2
9 2.4 [REVEALED] 10.4.0.2 ( 10.4.0.2) <250,250> [ frpla = 0 ][ qttl = 1 ][ uturn = 0 ] - step 1 ( Buddy used )
10
11 3 CE2 (192.168.2.2) <250,250> [ frpla = 3 ][ qttl = 1 ][ uturn = 0 ][ meta = 3, 0, 0 ]
12 4 CE2 (192.168.2.2) <250,250> [ frpla = 2 ][ qttl = 1 ][ uturn = 0 ][ meta = -1, 0, 0 ]
13 5 192.168.4.2 (192.168.4.2) <250,250> [ frpla = 1 ][ qttl = 1 ][ uturn = 0 ][ meta = 0, 0, 0 ]

```

## 4) Juniper Config – UHP:

## Juniper – RSVP-TE UHP

```

1 PE1
2 protocols {
3   rsvp {
4     tunnel-services;
5     interface all;
6   }
7   mpls {
8     no-propagate-ttl;
9     label-switched-path PE1-to-PE2 {
10       to 192.168.1.105;
11     }

```

```

12     interface all;
13 }
14 ospf {
15     traffic-engineering {
16         shortcuts lsp-metric-into-summary;
17     }
18 }
19 }
20
21 PE2
22 protocols {
23     rsvp {
24         tunnel-services;
25         interface all;
26     }
27     mpls {
28         no-propagate-ttl;
29         explicit-null;
30         icmp-tunneling;
31         label-switched-path PE2-toPE1 {
32             to 192.168.1.101;
33         }
34         interface all;
35         interface ge-0/0/2.0;
36     }
37     ospf {
38         traffic-engineering;
39     }
40 }
41
42 P1
43 protocols {
44     rsvp {
45         interface all;
46     }
47     mpls {
48         no-propagate-ttl;
49         interface ge-0/0/1.0;
50         interface ge-0/0/3.0;
51     }
52     ospf {
53         traffic-engineering;
54     }
55 }
56
57 P2
58 protocols {
59     rsvp {
60         interface all;
61     }
62     mpls {
63         no-propagate-ttl;
64         interface ge-0/0/1.0;
65         interface ge-0/0/3.0;
66     }
67     ospf {
68         traffic-engineering;
69     }
70 }
71
72 P3
73 protocols {
74     rsvp {
75         interface all;
76     }
77     mpls {
78         no-propagate-ttl;
79         interface ge-0/0/1.0;
80         interface ge-0/0/3.0;
81     }
82     ospf {
83         traffic-engineering;
84     }
85 }

```

### TNT running over Juniper – RSVP-TE UHP

```

1 Launching TraceTunnel: 192.168.2.102 (192.168.2.102)
2 1 CE1 ( 172.16.0.5) <255,64> [ frpla = 0 ][ rtla = 0(0) ][ qttl = 1 ][ uturn = 0 ]
3 2 PE1 ( 172.16.0.2) <254,63> [ frpla = 0 ][ rtla = 0(0) ][ qttl = 1 ][ uturn = 0 ]
4
5 RTL | Length estimation : 3 | Revealed : 3 (difference : 0)
6 2.1 [REVEALED] left.P1 (192.168.1.2) <253,62> [ frpla = 0 ][ rtla = 0(0) ][ qttl = 1 ][ uturn = 0 ] - step 0
7 2.2 [REVEALED] left.P2 (192.168.1.6) <252,61> [ frpla = 0 ][ rtla = 0(0) ][ qttl = 1 ][ uturn = 0 ] - step 0
8 2.3 [REVEALED] left.P3 (192.168.1.10) <251,60> [ frpla = 0 ][ rtla = 0(0) ][ qttl = 1 ][ uturn = 0 ] - step 0

```

```

9
10 3 left.PE2 (192.168.1.14) <250,62> [ frpla = 3 ][ rtla = 3(3) ][ qttl = 1 ][ uturn = 3 ]
11 4 CE2 (192.168.2.2) <250,61> [ frpla = 2 ][ rtla = 0(-1) ][ qttl = 1 ][ uturn = 0 ]
12 5 CE3 (192.168.2.102) <250,60> [ frpla = 1 ][ rtla = 0(0) ][ qttl = 1 ][ uturn = 0 ]

```

Again, as one can observe, there is no difference in the trace revelation regarding the LDP case (in any configurations).

## II. P2MP CIRCUITS (ALL INVISIBLE)

In this section, we study the VPRN case. In addition to MPLS, it requires the MP-BGP feature to build P2MP tunnels. We divide the analysis in two invisible tunnels sub-cases: the Implicit Null model (relying on the PHP option) and the Explicit Null one (enabling the UHP option). Note that we simplify a bit the given configurations for obvious readability purposes (as in the previous TE section). Moreover the IP configuration is not exactly the same as for previous cases as its is a more complex MPLS usage.

### A. VPN BGP MPLS – Implicit Null Model

#### 1) Cisco Config – PHP:

cisco IOS 15.2 – MPLS BGP VPN PHP

```

1 PE1
2 version 15.2
3 no propagate-ttl
4 mpls label protocol ldp
5
6 ip vrf VPN_Y
7 rd 1:2
8 route-target export 1:2
9 route-target import 1:2
10
11 interface GigabitEthernet4/0
12 description PEtoCEVPN
13 ip vrf forwarding VPN_Y
14
15 router rip
16 address-family ipv4 vrf VPN_Y
17 redistribute bgp 1 metric 1
18 network 10.0.0.0
19
20 router bgp 1
21 address-family vpnv4
22 neighbor 10.0.0.131 activate
23 neighbor 10.0.0.131 send-community extended
24 neighbor 10.0.0.132 activate
25 neighbor 10.0.0.132 send-community extended
26 neighbor 10.0.0.130 activate
27 neighbor 10.0.0.130 send-community extended
28
29 address-family ipv4 vrf VPN_Y
30 redistribute rip
31 no synchronization
32
33 PE2
34 version 15.2
35 no propagate-ttl
36 ip vrf VPN_Y
37 rd 1:2
38 route-target export 1:2
39 route-target import 1:2
40
41 interface GigabitEthernet4/0
42 description PEtoCEVPN
43 ip vrf forwarding VPN_Y
44
45 router rip
46 address-family ipv4 vrf VPN_Y
47 redistribute bgp 1 metric 1
48 network 10.0.0.0
49
50 router bgp 1
51 address-family vpnv4
52 neighbor 10.0.0.130 activate
53 neighbor 10.0.0.130 send-community extended
54 neighbor 10.0.0.132 activate
55 neighbor 10.0.0.132 send-community extended
56 neighbor 10.0.0.133 activate
57 neighbor 10.0.0.133 send-community extended
58
59 address-family ipv4 vrf VPN_Y
60 redistribute rip
61 no synchronization

```

## TNT running over 15.2 – VPN BGP MPLS PHP

```

1 Launching TraceTunnel: 10.0.1.103 (10.0.1.103)
2   1 192.168.3.2 (192.168.3.2) <255,255> [ frpla = 0 ][ qttl = 1 ][ uturn = 0 ]
3   2 10.0.0.54 ( 10.0.0.54) <254,254> [ frpla = 0 ][ qttl = 1 ][ uturn = 0 ]
4
5   OPAQUE | Length estimation : 3 | Revealed : 0 (difference : 3)
6
7   3 10.0.0.58 ( 10.0.0.58) <250,250> [ frpla = 3 ][ qttl = 1 ][ uturn = 0 ][MPLS LSE | Label : 28 | mTTL : 252 ]
8   4 10.0.0.57 ( 10.0.0.57) <249,249> [ frpla = 3 ][ qttl = 1 ][ uturn = 0 ]

```

The bottom label is revealed and equal to 252. This is the only visible MPLS indication. No revelation is working on such Opaque tunnels. Whatever the kind of probes sent to or through the VPRN, the IP address visible to TNT (or traceroute in general) is the outgoing address. Despite its expired TTL, it is likely that the probe arriving on the Egress PE will be pushed to the VRF of the VPN and its associated interface before generating the error message (the VPN being identified with the MPLS label contained in the packet). Then, the interface where the packet actually expires is the one associated to the VRF. However, we are able to distinguish UHP and PHP configurations (thanks to the so called LSE-TTL++), because the bottom label is equal to 255 for UHP and lower with PHP as we can see here. Note that the two last IP addresses can also trigger an alarm. In any cases, we are able to discriminate them from other opaque tunnels shown in previous P2P configurations. So we are able to identify their class as they are not possible to reveal.

## 2) Juniper Config – PHP:

## Juniper – VPN BGP MPLS PHP

```

1 PE1
2 routing-instances
3   VRF1
4     instance-type vrf;
5     interface ge-0/0/2.0;
6     route-distinguisher 192.168.1.101:1;
7     vrf-target target:65000:1;
8     protocols
9       bgp
10         group ce
11           type external;
12           peer-as 1;
13           neighbor 172.16.0.1;
14     protocols
15       mpls
16         interface all;
17         no-propagate-ttl;
18
19     bgp
20       group internal-peers
21         type internal;
22         local-address 192.168.1.101;
23         family inet-vpn
24           any;
25         export next-hop-self;
26         neighbor 192.168.1.106;
27         neighbor 192.168.1.105;
28
29 PE2
30 routing-instances
31   VRF1
32     instance-type vrf;
33     interface ge-0/0/2.0;
34     route-distinguisher 192.168.1.105:1;
35     vrf-target target:65000:1;
36     protocols
37       bgp
38         group ce
39           type external;
40           peer-as 3;
41           neighbor 192.168.2.2;
42     protocols
43       mpls
44         no-propagate-ttl;
45         icmp-tunneling;
46         interface all;
47
48     bgp
49       group internal-peers
50         type internal;
51         local-address 192.168.1.105;
52         family inet-vpn
53           any;
54         export next-hop-self;
55         neighbor 192.168.1.106;
56         neighbor 192.168.1.101;

```

## TNT running over Juniper towards the buddy – BGP MPLS VPN PHP

```

1 Launching TraceTunnel: 192.168.2.102 (192.168.2.102)
2 1 CE1 ( 172.16.0.5) <255,64> [ frpla = 0 ][ rtla = 0(0) ][ qttl = 1 ][ uturn = 0 ]
3 2 PE1 ( 172.16.0.2) <254,63> [ frpla = 0 ][ rtla = 0(0) ][ qttl = 1 ][ uturn = 0 ]
4
5 RTL | Length estimation : 3 | Revealed : 1 (difference : 2)
6 2.1 CE2 (192.168.2.2) <250,62> [ frpla = 0 ][ rtla = 0(0) ][ qttl = 1 ][ uturn = 0 ] - step 1 ( Buddy used )
7
8 3 CE2 (192.168.2.2) <250,62> [ frpla = 3 ][ rtla = 3(3) ][ qttl = 1 ][ uturn = 3 ]
9 4 CE3 (192.168.2.102) <250,61> [ frpla = 2 ][ rtla = 0(0) ][ qttl = 1 ][ uturn = 0 ]

```

We can observe that CE2 appears twice. When targeting the buddy (192.168.2.1), we *rediscover* CE2 since it now appears as if it was before the buddy in the topology. We say that we capture a *twisted IP* when targeting the buddy.

## TNT running over Juniper – BGP MPLS VPN PHP

```

1 Launching TraceTunnel: 192.168.2.1 (192.168.2.1)
2 1 CE1 ( 172.16.0.5) <255,64> [ frpla = 0 ][ rtla = 0(0) ][ qttl = 1 ][ uturn = 0 ]
3 2 PE1 ( 172.16.0.2) <254,63> [ frpla = 0 ][ rtla = 0(0) ][ qttl = 1 ][ uturn = 0 ]
4
5 RTL | Length estimation : 3 | Revealed : 1 (difference : 2)
6 2.1 CE2 (192.168.2.2) <250,62> [ frpla = 0 ][ rtla = 0(0) ][ qttl = 1 ][ uturn = 0 ] - step 1 ( Buddy used )
7
8 3 CE2 (192.168.2.2) <250,62> [ frpla = 3 ][ rtla = 3(3) ][ qttl = 1 ][ uturn = 3 ]
9 4 192.168.2.1 (192.168.2.1) <250,63> [ frpla = 2 ][ rtla = 4(0) ][ qttl = 1 ][ uturn = 4 ]

```

With Juniper VPN, there is no Opaque indicator resulting from VPRN or any other configurations. A first explanation is that Juniper routers, on the contrary to the independent mode enabled by default with Cisco routers, do not inject the whole IGP in LDP, but only their loopback address using the ordered mode (see Sec. II). This mode limits the probability to face a non-controlled tunnel ending. However, with VPRN configurations, a Juniper Egress LER deals with the same packet level situation as with Cisco routers. Up to the end of the tunnel, the packet is still MPLS encapsulated with the end-to-end VPN non terminating label at the bottom of the stack. Juniper routers do not, however, produce an Opaque indicator in that situation. Indeed, packets destined to the VPN are handled in a specific way with Juniper devices: they are IP packets forwarded directly to the next-hop without looking at or manipulating the IP-TTL whatever its value.

The outcome of such a sliding packet is twofold. Firstly, the Egress hop is hidden in the transit trace, as with Cisco UHP but without the duplicated IP. Secondly, when performing a direct trace (even with UDP) targeting the first address of the path within the VPN, i.e the IP interface of the Egress LER belonging to the VPN, one can see that this address and its buddy appear in the wrong order. Indeed, in the trace, the two addresses are switched, meaning that the CE IP address appears before the Egress one. Being forwarded without inspecting the IP-TTL, probes targeting IP addresses belonging to the VPN are automatically forwarded to the CE router, where they expire. The next probe, having a greater TTL, follows the same path as the one before, but can be forwarded back to the Egress LER by the CE router before expiring. This loop results in the two addresses being switched regarding their actual location in the path. Finally, one can infer the loop because two additional artifacts compared to RTLA (RTLA++) are visible: the TTL that deviates from its monotony and subsequent IP addresses also raise alarms due to potential conflicting allocation.

## B. VPN BGP MPLS – Explicit Null

## 1) Cisco Config – UHP:

## cisco IOS 15.2 – MPLS BGP VPN PHP

```

1 PE1
2 version 15.2
3 no propagate-ttl
4 mpls label protocol ldp
5
6 ip vrf VPN_Y
7 rd 1:2
8 route-target export 1:2
9 route-target import 1:2
10
11 interface GigabitEthernet4/0
12 description PEtoCEVPN
13 ip vrf forwarding VPN_Y
14
15 router rip
16  address-family ipv4 vrf VPN_Y
17 redistribute bgp 1 metric 1
18 network 10.0.0.0
19
20 router bgp 1
21  address-family vpnv4
22  neighbor 10.0.0.131 activate
23  neighbor 10.0.0.131 send-community extended
24  neighbor 10.0.0.132 activate
25  neighbor 10.0.0.132 send-community extended
26  neighbor 10.0.0.130 activate
27  neighbor 10.0.0.130 send-community extended
28
29  address-family ipv4 vrf VPN_Y
30 redistribute rip
31
32 PE2

```

```

33 version 15.2
34 no propagate-ttl
35 mpls ldp explicit-null
36
37 ip vrf VPN_Y
38 rd 1:2
39 route-target export 1:2
40 route-target import 1:2
41
42 interface GigabitEthernet4/0
43 description PEtoCEVPN
44 ip vrf forwarding VPN_Y
45
46 router rip
47 address-family ipv4 vrf VPN_Y
48 redistribute bgp 1 metric 1
49 network 10.0.0.0
50
51 router bgp 1
52 address-family vpnv4
53 neighbor 10.0.0.130 activate
54 neighbor 10.0.0.130 send-community extended
55 neighbor 10.0.0.132 activate
56 neighbor 10.0.0.132 send-community extended
57 neighbor 10.0.0.133 activate
58 neighbor 10.0.0.133 send-community extended
59
60 address-family ipv4 vrf VPN_Y
61 redistribute rip

```

#### TNT running over 15.2 – VPN BGP MPLS UHP

```

1 Launching TraceTunnel: 10.0.1.103 (10.0.1.103)
2   1 192.168.3.2 (192.168.3.2) <255,255> [ frpl = 0 ][ qttl = 1 ][ uturn = 0 ]
3   2 10.0.0.54 ( 10.0.0.54) <254,254> [ frpl = 0 ][ qttl = 1 ][ uturn = 0 ]
4
5     OPAQUE | Length estimation : 3 | Revealed : 0 (difference : 3)
6
7   3 10.0.0.58 ( 10.0.0.58) <250,250> [ frpl = 3 ][ qttl = 1 ][ uturn = 0 ][MPLS LSE | Label : 28 | mTTL : 255 ]
8   4 10.0.0.57 ( 10.0.0.57) <249,249> [ frpl = 3 ][ qttl = 1 ][ uturn = 0 ]

```

The bottom label is revealed and equal to 255. We can identify this tunnel as the VPRN UHP case but not reveal its content as for non VPRN opaque ones.

To conclude, it appears that VPRN content cannot be revealed with TNT, while other Opaque tunnels configurations (i.e., routing devices heterogeneity, BGP edge configuration) can. The mechanism behind the absence of content revelation can be explained by the IP address collected by TNT from the source IP field in the ICMP reply. Usually, the collected address is the one assigned to the physical incoming interface of the Egress PE. In the VPRN case, the collected IP is the one assigned to the interface on which the VRF is attached. In practice, this corresponds to the outgoing interface towards the VPN at the customer's side. Said otherwise, TNT collects the outgoing address instead of the incoming one. Because the incoming address is the only one that enables a successful revelation, this type of Opaque tunnels cannot be revealed yet.