Mitigating Spoofing Attacks in MPLS-VPNs using Label-hopping

Shankar Raman*, Gaurav Raina†
India-UK Advanced Technology Centre of Excellence in Next Generation Networks
**Department of Computer Science and Engineering, †Department of Electrical Engineering*
Indian Institute of Technology - Madras
Chennai - 600 036, India
Email: mjsraman@cse.iitm.ac.in, gaurav@ee.iitm.ac.in

***Abstract*—In certain models of inter-provider Multi-Protocol Label Switching (MPLS) based Virtual Private Networks (VPNs) spoofing attack against VPN sites is a key concern. For example, MPLS-based VPN inter-provider model "C" is not favoured, owing to security concerns in the data-plane, even though it can scale with respect to maintenance of routing state. Since the inner labels associated with VPN sites are not encrypted during transmission, a man-in-the-middle attacker can spoof packets to a specific VPN site. In this paper, we propose a label-hopping technique which uses a set of randomized labels and a method for hopping amongst these labels using the payload of the packet. To prevent the attacker from identifying the labels in polynomial time, we also use an additional label. The proposed technique can be applied to other variants of inter-provider MPLS based VPNs where Multi-Protocol exterior-BGP (MP-eBGP) multi-hop is used. As we address a key security concern, we can make a case for the deployment of MPLS based VPN inter-provider model "C".**

***Keywords*-MPLS; VPN; Model C; Spoofing attacks; Label-hopping;**

I. INTRODUCTION

Multi-Protocol Label Switching (MPLS) [6] technology uses fixed size labels to forward data packets between routers. By stacking labels, specific customer services such as Layer 3 Virtual Private Networks (L3-VPNs) based on Border Gateway Protocol (BGP) extensions are widely deployed in the Internet. BGP-based MPLS L3-VPN services are provided either on a single Internet Service Provider (ISP) core or across multiple ISP cores. The latter cases are known as inter-provider MPLS VPNs which are broadly categorized and referred to as models: "A", "B" and "C" [10].

Model "A" uses back-to-back VPN Routing and Forwarding (VRF) connections between Autonomous System Border Routers (ASBRs). Model "B" uses eBGP redistribution of labelled VPN-IPv4 routes from Autonomous Systems (AS) to neighbouring AS. Model "C" uses multi-hop MP-eBGP redistribution of labelled VPN-IPv4 routes and eBGP redistribution of IPv4 routes from an AS to a neighbouring AS. Model "C" is more scalable for maintaining routing states and hence preferred for deployment in the Internet; refer to [2] for more details. Security issues in MPLS, especially MPLS-based VPNs has attracted attention [1].

The security of model "A" matches the single-AS standard proposed in [9]. Model "B" can be secured well on the control-plane, but on the data-plane the validity of the outer-most label (Label Distribution or Resource Reservation Protocol label) is not checked. This weakness could be exploited to inject crafted packets from inside an MPLS network core. A solution for this problem is proposed in [2]. Model "C" can be secured on the control-plane but has a security weakness on the data-plane. The Autonomous System Border Routers (ASBRs) do not have any VPN information and hence the inner-most label cannot be validated. In this case, the solution used for Model "B" cannot be applied. An attacker can exploit this weakness to send unidirectional packets into the VPN sites connected to the other AS. Therefore, ISPs using model "C" must either trust each other or not deploy it [4].

Control plane security issue in model "C" can be resolved by using IPSec. If IPSec is used in the data-plane then configuring and maintaining key associations could be extremely cumbersome. Even though model "C" is highly scalable for carrying VPN Routing and Forwarding (VRF) routes, the vulnerability of the data-plane renders it unusable. The current recommendation is that model "C" must not be used. A simple solution to this problem is to filter all IP traffic with the exception of the required eBGP peering between the ASBRs, thereby preventing a large number of potential IP traffic-related attacks. However, controlling labelled packets is difficult. In model "C", there are at least two labels for each packet: the Provider Edge (PE) label, which defines the Label Switched Path (LSP) to the egress PE, and the VPN label, which defines the VPN associated with the packet on the PE.

In [5], the authors propose encryption techniques, such as IPSec, for securing the provider edge (PE) of the network. The authors also highlight that the processing capacity could be over-burdened. Further, if an attacker is located at the core of the network, or in the network between the providers that constitute an inter-provider MPLS VPN, then spoofing attacks are possible. The vulnerability of MPLS against spoofing attacks and performance impact of IPSec has been discussed in [3]. If the inner labels that identify packets going towards a L3 VPN site are spoofed, then

 ISBN: 978-1-61208-183-0

sensitive information related to services available within the organizational servers can be compromised. As far as we know, there is no scheme available for installing an anti-spoofing mechanism for these VPN service labels.

This paper outlines a label-hopping technique that helps to alleviate the data-plane security problem in model "C". We propose a scheme that changes the inner VPN labels dynamically based on the payload. By using a mix of algorithms and randomized labels, we can guard against spoofing and related attacks. The advantage of our scheme is that it can be used wherever Multiprotocol-external BGP (MP-eBGP) multi-hop scenarios arise.

The rest of the paper is organized as follows. In Section II, we discuss the pre-requisites of our proposed scheme. In Section III, we discuss the label-hopping technique and some implementation issues. In Section IV, we discuss the preliminary simulation and implementation issues. We present our conclusions and provide avenues for future work in Section V.

II. PRE-REQUISITES FOR THE LABEL-HOPPING SCHEME

In this section, we briefly review the network topology for model "C", the PE configuration and the control-plane exchanges needed for our proposed scheme.

A. MPLS VPN model "C"

The reference MPLS-eBGP based VPN network for model "C" as described in [11] is shown in Figure 1, which also shows the control plane exchanges. The near-end PE (PE_{ne}) and far-end PE (PE_{fa}) are connected through the inter-provider MPLS core. The VPN connectivity is established through a set of routers from different Autonomous Systems (AS) and their ASBRs. In the VPN, MP-eBGP updates are exchanged for a set of Forward Equivalence Classes (FECs). These FECs, which have to be protected, originate from the prefixes behind PE_{ne} in a VPN site or a set of VPN sites.

B. PE configuration

Various configurations are needed on the PEs to implement the label hopping scheme. A set of "m" algorithms that generate collision-free labels (universal hashing algorithms) are initially implemented in the PEs. Each algorithm is mapped to an index $A = (a_1, a_2, \cdots a_m), m \geq 1$. The bit-selection pattern used by the PEs for generating the additional label is also configured. PE_{ne} must be configured for a FEC or a set of FECs represented by an aggregate label (per VRF label) which will use the label-hopping scheme. For each FEC or a set of FECs, a set of valid labels used for hopping, $K = (k_1, k_2, k_3, \cdots k_n), n > 1$ and, $k_i \neq k_j$ if $i \neq j$, is configured in PE_{ne}. In the case of bi-directional security, the roles of the PEs can be reversed.

C. Control and data-plane flow

Initially, set K and the bit-selection pattern used by the PEs are exchanged securely over the control-plane. Optionally an index from A, representing a hash-algorithm, could also be exchanged. We propose that only the index is exchanged between the PEs, as it enhances the security, for two reasons. First, the algorithm itself is masked from the attacker. Second, the algorithm can be changed frequently, and it would be difficult for the attacker to identify the final mapping that generates the label to be used for a packet. Figure 1 depicts this unidirectional exchange from PE_{ne} to PE_{fa}.

Once the secure control-plane exchanges are completed, we apply the label-hopping technique, and PE_{fa} forwards the labelled traffic towards PE_{ne} through the intermediate routers using the label-stacking technique (Figure 2). The stacked labels along with the payload are transferred between the AS and ASBRs before they reach PE_{ne}. Using the label-hopping algorithm PE_{ne} verifies the integrity of labels. Upon validation, PE_{ne} uses the label information to forward the packets to the appropriate VPN service instance or site. This data-plane exchange from PE_{fa} and PE_{ne} is depicted in Figure 3. We now present the label-hopping scheme.

III. LABEL-HOPPING TECHNIQUE

In this section, we describe the label-hopping technique and discuss some implementation aspects.

Once a data packet destined to the PE_{ne} arrives at the PE_{fa} a selected number of bytes from the payload is chosen as input to the hashing algorithm. The hash-digest obtained as a result is used to obtain the first label for the packet. The agreed bit-selection pattern is then applied on the hash-digest to obtain an additional label, which is then concatenated with the first label. Once PE_{ne} receives these packets it verifies both the labels.

The implementation steps for the control-plane at the PE_{ne} and PE_{fa} are given by Algorithms 1 and 2. The implementation steps for the data-plane at the PE_{fa} and PE_{ne} are given by Algorithms 3 and 4.

Note: The values in K need not be contiguous and can be randomly chosen from a pool of labels to remove coherence

Algorithm 1 Control-plane PE_{ne} algorithm

Require: FEC[] Forward Equivalence Classes, K[] valid labels, A[i] hash algorithm instance, I[] the bit-selection pattern chosen for the inner label.

```
Begin
packet = makepacket(FEC,K, A[i], I);
CP-SendPacket(PE_fa, MP-eBGP, packet);
End
```

 ISBN: 978-1-61208-183-0

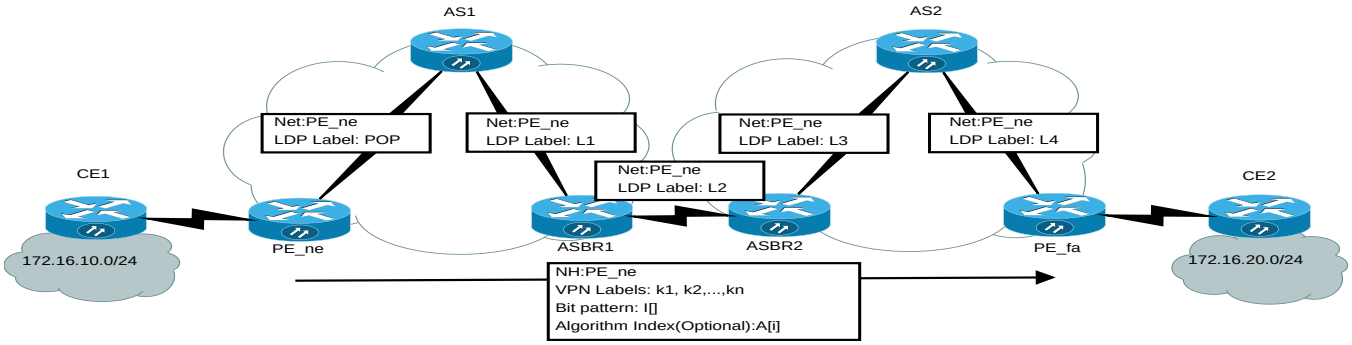

Figure 1: Control-plane exchanges for model C [11]

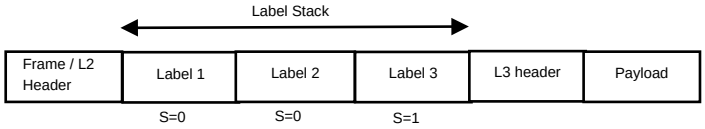

Figure 2: Label stack using scheme outlined for Model "C"

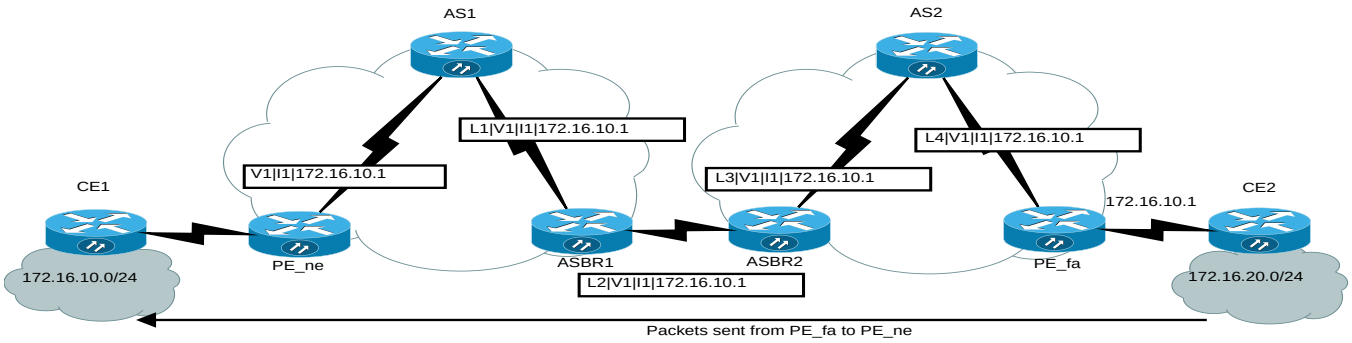

Figure 3: Data-plane flow for model C [11]

Algorithm 2 Control-plane PE_{fa} algorithm

Require: None

Begin
packet = CP-ReceivePacket(PE_{ne}); // from PE_{ne}
FEC[] = ExtractFEC(packet); // extract FECs
K[] = ExtractLabels(packet); // extract the labels
selectHashAlgorithm(A[i]); // hash algorithm to use
RecordValues(FEC); // information for PE_{fa}
RecordValues(K);
RecordValues(I); // bit-selection pattern to be used
End

Algorithm 3 Data-plane PE_{fa} algorithm

Require: None

Begin
packet = DP-ReceivePacket(Interface);
match = CheckFEC(packet); // Is the algorithm enabled?
if $match == 0$ **then**
  return; // no match
end if
hash-digest = calculateHash(A[i],packet);
first-label = hash-digest % $|K|$;
additional-label = process(hash-digest,I)
DP-SendPacket(PE_{ne}, first-label, additional-label, packet);
End

in the label space. Also the algorithms used could be either vendor dependent or a set of standard algorithms mapped the same way by the PE_{ne} and PE_{fa}. If the two PEs involved are from different vendors we assume that a set of standard algorithms are used. In order to avoid too many processing cycles in the line cards of PE_{ne} and PE_{fa}, the hash-digest is calculated over a predefined size of the payload. An additional inner label is further added to enhance protection against spoofing attacks. With an increased label size, an

Algorithm 4 Data-plane PE_{ne} algorithm

Require: None

```
Begin
packet = DP-ReceivePacket(Interface);
match = CheckFEC(packet);
if match == 0 then
   return; //no match
end if
label-in-packet=extractPacket(packet, LABEL);
inner-label=extractPacket(packet, INNER-LABEL);
hash-digest=calculateHash(A[i],packet);
first-label=hash-digest % |K|;
additional-label = process(hash-digest,I)
if label-in-packet ≠ first-label then
   error(); return;
end if
if inner-label ≠ additional-label then
   error(); return;
end if
DP-SendPacket(CE1, NULL, NULL, packet);
End
```

attacker spends more than polynomial time to guess the VPN instance label for the site behind PE_{ne}. There could be two hash-digests that generate the same label. In this case, the two hash-digests is differentiated using the additional label. Collisions can be avoided by re-hashing or any other suitable techniques that are proposed in the literature [8]. If collisions exceed a certain number, then Algorithms 1 and 2 can be executed with a set of new labels.

Illustration: We now briefly illustrate the label-hopping scheme. In Figure 1, using Algorithms 1 and 2, a set of labels are forwarded from PE_{ne} to PE_{fa}. The roles of PE_{ne} and PE_{fa} are interchanged for reverse traffic. Figure 2 shows a packet from the data-plane for model "C", with the proposed scheme. In the figure, "Label 1" refers to the outermost label, while "Label 2" refers to the label generated from the hash-digest and "Label 3" refers to an additional label generated as in Algorithm 3. This additional label has bottom of stack bit (denoted by S in Figure 2) set. These labels are stacked immediately onto the packet and the path labels for routing the packets to appropriate intermediary PEs are added. Figure 3 also shows these path labels used by the data packet to reach PE_{ne}. When the packet passes through the core of an intermediary AS involved in model "C", or through the network connecting the intermediary AS, the intruder or the attacker has the capability to inspect the labels and the payload. However, the proposed scheme prevents the attacker from guessing the right combination of the labels. We can increase the size of the additional inner-labels thereby reducing threats from polynomial time attacks.

IV. Simulation and Implementation

In this section, we present the preliminary simulation results on performance, comparing the label-hopping technique with deep packet inspection where we encrypt and decrypt the complete packet. We also briefly highlight some implementation issues.

A. Simulation

Implementing the label-hopping scheme for all set of FECs belonging to any or all VPN service instances may cause throughput degradation. This is because the hash-digest computation and derivation of the inner-label / additional inner label calculation can be computation intensive. We therefore compared our technique by choosing a part of the payload as input to our hashing algorithm.

We simulated our algorithm on a 2.5 GHz processor Intel dual processor quad core machine. We compared the performance of the label-hopping technique with a deep packet inspection technique where the complete packet was encrypted before transmission and decrypted on reception. The performance figures are shown in Figure 4. These simulation figures indicate that we were able to process 10 million packets per second when we used 64-byte for hashing on a payload of size 1024 bytes. For a hash using 128-byte, we were able to process about 6.3 million packets per second. However with a deep packet inspection where we encrypted and decrypted the complete packet, we were able to process only about 1 million packets per second.

In cases where performance becomes a bottleneck, this label-hopping scheme can be applied to specific traffic which are mission-critical, sensitive and most likely need to be protected as they travel from the PE_{fa} to the PE_{ne}. Selective application of this service which could be offered as a premium for a selected set of FECs is a suitable option, there by protecting the traffic of organizations that are paranoid about the integrity of the switched traffic into their VPN sites.

B. Implementation

We are modifying the open source Quagga router software on Linux to implement our scheme. One of the concerns in the scheme is the use of payload for generating the random source. If the payload does not vary between two packets then the control-plane exchanges have to be renegotiated with a different set of labels for the second packet. The other concern in the scheme is to tackle the problem of fragmentation that can occur along the path from PE_{fa} to PE_{ne}. We can fragment the packet at PE_{fa} and ensure that the size of the packet is fixed before transmission. We could also employ the Path Maximum Transfer Unit (Path-MTU) discovery process so that packets do not get

 ISBN: 978-1-61208-183-0

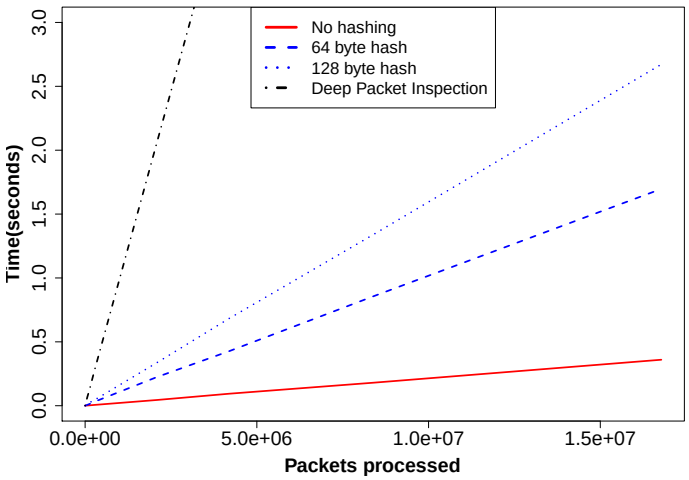

Figure 4: Performance comparison of complete packet encryption and decryption with a 64, 128 byte hash on a payload of size 1024 bytes.

split into multiple fragments. If packets are fragmented this scheme fails. However, networks usually employ the Path-MTU discovery process to prevent fragmentation and hence this problem may not occur.

V. Conclusion and Future Work

In this paper, we proposed a label-hopping scheme for inter-provider BGP-based MPLS VPNs that employ MP e-BGP multi-hop control-plane exchanges. In such an environment, without label-hopping, the data-plane is subject to spoofing attacks.

The technique proposed uses a payload-based label-hopping scheme to prevent attackers from easily deciphering labels and their respective VPNs. The scheme is less computationally intensive than encryption-based methods. It prevents the spoofed packets from getting into a VPN site even if the attacker is in the core or at an intervening link between ISPs. In our scheme, we chose the payload of the packet as the variable component since the use of encryption or IPSec to secure the inner labels are time intensive strategies. Instead of using the payload as a random source, other options like time-of-the-day could be used. This requires the use of time synchronization mechanism. Such mechanisms like "Timing over IP Connection and Transfer of Clock (TicToc)" are receiving much attention from the IETF. This will be the subject of our future study.

Acknowledgements

The authors would like to acknowledge the UK EP-SRC Digital Economy Programme and the Government of India Department of Science and Technology (DST) for funding given to the IU-ATC.

References

[1] S. Alouneh, A. En-Nouaary and A. Agarwal, "MPLS security: an approach for unicast and multicast environments", Annals of Telecommunications, Springer, vol. 64, no. 5, June 2009, pp. 391–400, doi:10.1007/s12243-009-0089-y.

[2] M. H. Behringer and M. J. Morrow, "MPLS VPN security", Cisco Press, June 2005, ISBN-10: 1587051834.

[3] B. Daugherty and C. Metz, "Multiprotocol Label Switching and IP, Part 1, MPLS VPNS over IP Tunnels", IEEE Internet Computing, May–June 2005, pp. 68–72, doi: 10.1109/MIC.2005.61.

[4] L. Fang, N. Bita, J. L. Le Roux and J. Miles, "Interprovider IP-MPLS services: requirements, implementations, and challenges", IEEE Communications Magazine, vol. 43, no. 6, June 2005, pp. 119–128, doi: 10.1109/MCOM.2005.1452840.

[5] C. Lin and W. Guowei, "Security research of VPN technology based on MPLS", Proceedings of the Third International Symposium on Computer Science and Computational Technology (ISCSCT 10), August 2010, pp. 168–170, ISBN-13:9789525726107.

[6] Y. Rekhter, B. Davie, E. Rosen, G. Swallow, D. Farinacci and D. Katz, "Tag switching architecture overview", Proceedings of the IEEE, vol. 85, no. 12, December 1997, pp. 1973–1983, doi:10.1109/5.650179.

[7] E. Rosen and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, Standard Track, February, 2006.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, "Introduction to algorithms", 3rd edition, MIT Press, September 2009, ISBN-10:0262033844.

[9] C. Semeria, "RFC 2547bis: BGP/MPLS VPN fundamentals", Juniper Networks white paper, March 2001.

[10] Advance MPLS VPN Security Tutorials [Online], Available: "http://etutorials.org/Networking/MPLS+VPN+security/ Part+II+Advanced+MPLS+VPN+Security+Issues/", [Accessed: 10th December 2011]

[11] Inter-provider MPLS VPN models [Online], Available: "http://mpls-configuration-on-cisco-ios-software.org.ua/1587051990/ ch07lev1sec4.html", [Accessed 10th December 2011]

 ISBN: 978-1-61208-183-0