Mitigating spoofing and replay attacks in MPLS-VPNs using label-hopping with TicToc

Shankar Raman
Balaji Venkat
Gaurav Raina
Outline

Security issues in MPLS-VPN models
Label-hopping algorithms
Simulation and Implementation
MPLS VPN Security Issues

Model A: Highly secure, misconfiguration of ASBR can compromise security

Model B: Secure control plane, data plane security by adding an extra ASBR

Model C: Secure control plane, no data plane security, ISPs must trust each other

Service labels can be spoofed
## Router Configuration

<table>
<thead>
<tr>
<th>Algorithm 1</th>
<th>Algorithm 2</th>
<th>Algorithm m</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>m</td>
</tr>
</tbody>
</table>

Algorithm indices are exchanged

<table>
<thead>
<tr>
<th>K11,.., K1i</th>
<th>K21,.., K2j</th>
<th>Kn1,.., Knk</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEC 1</td>
<td>FEC 2</td>
<td>FEC n</td>
</tr>
</tbody>
</table>

Keys for FECs

| Bits Chosen 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31|

16 bits chosen out of 32 bits
Secure Control Plane Exchange
Label hopping applied to data plane

Illustration of data plane transfer

Label 1 = Outermost label
Label 2 = Label generated by the hash digest
Tic-Toc based Scheme

- Timing over IP Connection and Transfer of Clock (Tic-Toc) (IEEE 1588),
- Exchange Time slices,
- Exchange labels during these time slices,
Control plane algorithms for PEne

Require:

* FEC[] Forward Equivalence Classes,
* K[] valid labels,
* TS[] valid time slices,
* A[i] hash algorithm instance,
* I[] the bit-selection pattern chosen for the inner label.
* Random seed "Rseed" which is used for generating the index into set K (set of labels).
* PTP port and PTP LSP information

Begin
packet = makepacket(FEC, K, TS, A[i], I, Rseed);
CP-SendPacket(PEfa, MP-eBGP, packet);
End
Control plane algorithms for PEfa

Require: None
Begin
packet = CP-ReceivePacket(PEne); // from PEne
FEC[] = ExtractFEC(packet); // extract FECs
K[] = ExtractLabels(packet); // extract the labels
TS[] = ExtractTimeSlices(packet); // extract the time slices
Rseed = ExtractRandomSeed(packet); // extract the Rseed value.
selectHashAlgorithm(A[i]); // hash algorithm to use
RecordValues(FEC); // information for PEfa
RecordValues(K);
RecordValues(TS);
RecordValues(I); // bit-selection pattern to be used
RecordValue(Rseed);
End
Data Plane Algorithm for PEfa

Begin
Initialization:

One Time Init:
BeginInit

CurrentTimeSliceIndex = 0;

CurrentMasterClock = PTP LSP Master Clock Timestamp;

CurrentTimeInstant = CurrentMasterClock;

NextTimeInstant = CurrentMasterClock + TS[CurrentTimeSliceIndex];

EndInit

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);
if (CurrentTimeInstant <= NextTimeInstant ((+ or -) configured seconds)) then
    // do nothing;
else
    CurrentTimeSliceIndex++;
    if CurrentTimeSliceIndex == n then // check to wrap around
        CurrentTimeSliceIndex = 0;
    end if
    CurrentTimeInstant = NextTimeInstant;
    NextTimeInstant = CurrentTimeInstant + TS[CurrentTimeSliceIndex];
end if
first-label = K[GenerateRandom(Rseed) MOD n(K)];
end if
additional-label = process(hash-digest,I)
DP-SendPacket(PEne, first-label, additional-label, packet);
End
Data Plane Algorithm for PEne

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);
if (CurrentTimeInstant <= NextTimeInstant ((+ or -) configured seconds)) then
  // do nothing;
else
  CurrentTimeSliceIndex++;
  // Save the old RseedIndex into set K
  OldRseedIndex = RseedIndex;
  RseedIndex = (GenerateRandom(Rseed) MOD n(K));
  NextRseedIndex =
    LookAheadRseedIndex(GenerateRandom(Rseed) MOD n(K));
  RollbackRseed(Rseed by 1);
  if CurrentTimeSliceIndex == n then // check to wrap around
    CurrentTimeSliceIndex = 0;
  end if
  CurrentTimeInstant = NextTimeInstant;
  NextTimeInstant = CurrentTimeInstant + TS[CurrentTimeSliceIndex];
end if
// Check if label used before in the previous | current or future
// time slot can be used
// Check with OldRseedIndex, RseedIndex and NextRseedIndex
first-label-range = K[RseedIndex (+or- 1)];
additional-label = process(hash-digest,I)
if label-in-packet ! in first-label-range then
  error(); return;
end if
Avoiding replay attacks

- Exchange the seed
- Use Pseudo Random Number Generation algorithm
- Use the Random Number generated to choose the labels at various time slices
Simulation and Implementation

- Quagga open source software on an desktop machine,
- Payload is used as the random source was a concern,
- Fragmentation of packets - path discovery MTU.
Conclusion

- Security solution for MPLS-VPN Model “C” using label-hopping
- We make a case of deployment of Model “C”
Thank you

QUESTIONS?