OSPF WG

Security Extensions for OSPFv2 when using Manual Keying

Manav Bhatia, Alcatel-Lucent
Sam Hartman, Painless security
Dacheng Zhang, Huawei
IETF 80, Prague
Current State of Security

- OPSEC has published RFC 6039 that does an analysis on the vulnerabilities that exist in OSPFv2 despite it using the security and authentication mechanisms described in RFC 2328 and 5709

- draft-ietf-karp-ospf-analysis identifies certain gaps that remain between the current security state and those identified in draft-ietf-karp-threats-reqs
Gaps Identified

- **Replay Protection**
  - OSPFv2 uses Cryptographic Sequence numbers to prevent intra-session replay attacks
  - Does not help in protecting against inter-session replay attacks

- **IP Header Unprotected**
  - OSPFv2 uses the source IP to identify the neighbor in some cases
  - IPv4 Header is not protected by the authentication digest
So what does this draft do?

- It fixes the issues identified during the OSPFv2 gap analysis
- Proposes two mechanisms to prevent inter-session replay attacks
  - Extends the Authentication Sequence Number space
  - Introduces the concept of Session ID and Nonce
- Fixes the IP header issue by factoring in the source IP address when computing the crypto digest - thus attacks which change this, will not be successful now
Inter-Session Replay Attack

Router A accepts the packet and brings down the adjacency with B!

Router A

OSPF Hdr: Sequence Num = 10001
OSPF HELLO: Neighbor = B;

OSPF Hdr: Sequence Num = 10001
OSPF HELLO: Neighbor = B;

OSPF Hdr: Sequence Num = 10010
OSPF HELLO: Neighbor = 0

OSPF Hdr: Sequence Num = 10011
OSPF HELLO: Neighbor = B

OSPF Hdr: Sequence Num = 50000
OSPF HELLO: Neighbor = 0

Router B

OSPF Hdr: Sequence Num = 50001
OSPF HELLO: Neighbor = A;

OSPF Hdr: Sequence Num = 1
OSPF HELLO: Neighbor = 0

OSPF Hdr: Sequence Num = 2
OSPF HELLO: Neighbor = A

OSPF Hdr: Sequence Num = 50000
OSPF HELLO: Neighbor = 0

Router B goes down!
So how do we fix this? (1/2)

- OSPF authentication mechanism is stateless and oblivious to the session information
  - Router A for example doesn’t remember that it once had an OSPF session with B and the last cryptographic sequence number seen from B was 50001
  - Highly un-scalable and also requires B to keeping updating the non-volatile memory each time it increments a sequence number so that it can continue from there.
So how do we fix this? (2/2)

- Change the crypto sequence number generation algorithm at the sender side so that it always generates an increasing number (for both planned and unplanned restarts)
- Implement some algorithm that guarantees freshness of packets
- We describe both in the draft
Changing the crypto sequence number algorithm

- Currently the sequence number is a 32-bit monotonically increasing entity
- Expand this to 64 bits where:
  - most significant 32-bits increment each time the router cold boots.
  - last 32-bits remain unchanged
- The final sequence number is a concatenation of the above two numbers
So does this help?

Router A

Router B

Router B goes down!

Router A rejects this as sequence number < 11:2
So where are we?

- We believe it solves the inter-session replay attacks with OSPF.
- This solution does NOT guarantee packet freshness, i.e., you still don’t know if you are speaking to a live router or if somebody is playing out the entire conversation.
- If you want to fix this then the draft spells out the challenge/response mechanism using the Session IDs and Nonces.
Benefits

- Easy to implement - very minimal changes to the OSPF running code
- Consider this as part of the KARP infrastructure that even other routing protocols can use
- Minimal changes required in the OSPF packet encoding
Router A

C/R Solution (1)

Router B

OSPF Hdr: Session ID = X1; Nonce = N1
OSPF HELLO: Neighbor = 0

OSPF Hdr: Session ID = X2; Nonce = N2
OSPF HELLO: Neighbor = 0

OSPF Hdr: Session ID = X1; Nonce = N1
OSPF HELLO: Neighbor = B;
Session ID = X2; Nonce = N2

OSPF Hdr: Session ID = X1; Nonce = N1
Normal OSPF DD packet

OSPF Hdr: Session ID = X2; Nonce = N2
Normal OSPF DD packet

OSPF Hdr: Session ID = X1; Nonce = N1
Normal OSPF Link State Packet

OSPF Hdr: Session ID = X1; Nonce = N1
OSPF HELLO: Neighbor = B;
Session ID = X2; Nonce = N2

Router B reaches the 2way state and can now initiate the DD exchange. All Hellos from B will now contain A's Session ID and the nonce value.

The OSPF header carries the Session ID and the Nonce value. There is no change in the DD packet. Router A will accept any packet from B as long as the header carries the same Session ID and the nonce value that it sees in its HELLOs.

Router A keeps sending its HELLO listing B as its neighbor along with the last Session ID and the nonce value.

Scenario 1: Two Routers coming up..

C/R Solution (2)

Scenario 2: Another Router C comes up on that LAN

Router A

OSPF Hdr: Session ID = X1; Nonce = N1
 OSPF HELLO: Neighbor = B;
 Session ID = X2; Nonce = N2
 Neighbor2 = C;
 Session ID = X3; Nonce = N3

OSPF Hdr: Session ID = X1; Nonce = N1'
 Normal OSPF DD packet

OSPF Hdr: Session ID = X2; Nonce = N2
 OSPF HELLO: Neighbor = A;
 Session ID = X1; Nonce = N1'
 Normal OSPF DD packet

OSPF Hdr: Session ID = X1; Nonce = N1
 Normal OSPF Link State Packet

Router C

OSPF Hdr: Session ID = X3; Nonce = N3
 OSPF HELLO: Neighbor = 0

Router C's blank HELLO with its Session ID and nonce value

Upon hearing a new router, A uses a new Nonce N1'. Router B, updates its data structures to use the new value. It knows its correct since the HELLOs are listing Router B's correct Session ID and nonce values

The OSPF header carries the Session ID and the Nonce value. There is no change in the DD packet. Router A will accept any packet from C as long as the header carries the same Session ID and the nonce value that it sees in its HELLOs
Scenario 3: Router A reboots

Router B sees the new HELLO with its new nonce value, updates its internal state of A with its new session ID X4 and nonce value N4. It carries these new values in its subsequent packets.

Router B receives new HELLO -- makes note of it and changes its nonce value. Continues listing A with the earlier session ID and nonce values.

Router A reboots and uses a new session ID and nonce values.
Next Steps

- We need people who understand OSPF to look at this mechanism and see if they find some holes in it.
- If they think this is fool-proof then we can remove the Session ID and the Nonce stuff that currently exists in the draft.
- Accept this as a WG document since there has been a lot of discussion on the mailing list and people have taken it positively there!
Feedback!
Protecting the source IP address

1. OSPF Packet replayed and source IP changed from X to X'.

2. B computes the digest assuming the source IP as X'.

3. B rejects the packet as the computed digest does NOT match the digest carried in the packet!

Authentication has been computed assuming source IP as X.