Defending IKE Responders Against Denial of Service Attacks

Yoav Nir

Agenda

- The Attack
- The Stateless Cookie
- Rate Limiting
- The Puzzle
- A Strategy for Defense

The Attack

- A Denial of Service Attack on an IKE Responder involves initiating multiple sessions with an IKE Responder.
- A Distributed Denial of Service Attack involves multiple attackers. The number could be hundreds, thousands, even a hundred thousand attackers.
- In all cases, the attackers don't have valid credentials, and cannot complete the IKE_AUTH exchange.

The Attack

- One possible attack is to fill up the half-open SA database.
- A single IKE_SA_INIT request creates an entry in the HOSA-DB. An entry can take from dozens to hundreds of bytes.
- This attack can be launched from multiple spoofed addresses, creating millions of entries every second.

The Attack

- A more complex attack requires return routability. This time the attacker sends an IKE_SA_INIT request, receives a response, and sends an IKE_AUTH request.
- The IKE_AUTH request is just garbage data. The cost to the attacker is trivial, but the Responder must complete the D-H exchange, derive keys and attempt to decrypt.

The Attack

- An even more complex form of the attack has the attacker sending an IKE_AUTH request that decrypts OK.
- The decrypted packet forces the Responder to verify a chain of certificates, which takes even more resources.
- This is rarely the most efficient attack, because if requires the attacker to do a lot of work.

The Stateless Cookie

- This is the most basic defense. It's defined in RFC 7296.
- When the Responder is under load (no definition supplied), it challenges the Initiator with a stateless cookie.
- The Initiator must repeat the IKE_SA_INIT request with the cookie included.
- Prevents initiations from spoofed sources. It does not stop attacks with return routability.

Rate Limiting

- This is a defense for the HOSA-DB.
- For each source IPv4 address of IPv6 prefix, we limit the amount of concurrent half-open SAs that the Responder is willing to keep.
- The limit can be hard or soft. With a soft limit, if the limit is exceeded, a Stateless Cookie or a Puzzle is required. If it's only a Cookie for the first level, a multi-level limit is required.
- Rate Limiting only works if return routability is enforced

Retention Time

- When the Half-Open SA Database is being attacked one way to reduce the gain for the attacker is to reduce the time that a half-open SA is retained.
- We'd like a long time to allow for packet loss, so we don't want to make the retention time always short.
  - When under attack, can be reduced to whatever it takes to derive keys, generate the message + 1 RTT.

The Puzzle

- Rate Limiting and Stateless Cookies together can foil the simple attack. The attacks with an IKE_AUTH request can still overwhelm the Responder.
- That's where the puzzle comes in.
- It increases the difficulty for the attacker by requiring proof-of-work before accepting a half-open SA.

The Puzzle

- The Puzzle challenge in the current draft is based on the proof-of-work in the Bitcoin block chain.
- The Responder sends two pieces of information to the Initiator:
  - Cookie – similar to the regular stateless cookie.
  - Difficulty level – a number between 0 and 255.
    - The draft limits the number to at least 8.

The Puzzle

- The Initiator returns a Cookie notification payload, but the cookie that it sends is a bit longer. The content of the notification is as follows:

  `Notification Data = Cookie || extra bytes`

- The `extra bytes` are chosen so that the `SHA2-256` hash of the Notification Data has a number of trailing zero bits at least as high as the difficulty level.

The Puzzle - Example

- Cookie = `fdbcfa5a430d7201282358a2a034de0013cfe2ae` (probably the SHA-1 hash of something)
- Difficulty Level = 22
- Extra Bytes = `5c2880`
- SHA2-256 hash: `155319280d687074d0f78511f63c77c568a5418dd44e6467d8fc37723d800000`
- Trailing Zero Bits = 23
- Time on a laptop computer: 2.8 s single-core.

The Puzzle - Issues

- Initiators differ in their ability to perform SHA2-256 calculations.
  - >12M hashes/s for a 4-core Xeon chip
  - ~250K hashes/s for an ARM core
- You don't want to set the difficulty so high that your defense is DoS-ing the phones
- But you do want it to make a difference.
- Use the puzzles sparingly.

The Strategy

- Before an attack is detected:
  - No Cookies or puzzles for the general population.
  - Reasonable soft rate limit (~5) per-address/prefix
  - Puzzle level set for ~2 seconds (weakest client).
  - Go to the next level in two cases:
    - Overall HOSA-DB passes a certain threshold. (20%?)
    - Number of addresses/prefixes where puzzles are required exceeds some level (10-20?)

The Strategy

- First Level of Attack:
  - Enforce Cookies for everyone
  - Puzzles for suspicious prefixes and addresses.
    - Suspicious means multiple half-open SAs or multiple failed IKE_AUTH within a certain time.
    - Puzzle difficulty can be increased to ~5 seconds for the weakest legitimate Initiator.
  - Reduce HOSA-DB retention time
  - Go to next level if HOSA_DB goes beyond a certain level (80% ?) or failed IKE_AUTH rate crosses a certain level.

The Strategy

- High Level Attack
  - At this point any new connection is suspect.
  - All Initiators are required to solve a puzzle
  - Suspicious initiators are required to solve a hard puzzle.
  - Retention time is reduced to the minimum.
  - Hard Puzzle difficulty level may be increased to make it infeasible for some legitimate clients.

Comments?