Internship Report

Aryan Gupta

June 2020

1 Objective

The internship aimed at finding a way to implement IPSec in SCION Architecture using PISKES key as a pre-shared key for authentication.

2 IPSec and PISKES

As mentioned in RFC4306, (IPsec) provides confidentiality, data integrity, access control, and data source authentication to IP datagrams. These services are provided by maintaining shared state between the source and the sink of an IP datagram. IKE is a protocol for establishing this shared state.
Originally IKE had two modes to choose from:

1. Main mode- It involved sharing of 3 pairs of messages. One message involved sharing of the choice of Diffie Hellman group.

2. Aggressive mode- It involved sharing of 2 pairs of messages. This is very insecure when used with pre shared key and some routers e.g. CISCO routers have by default blocked the use of PSK with Aggressive mode

IKEv2 does not offer choice of modes.

2.1 A quick overview of IKE

It involves sharing of two pairs of messages to establish an IKE SA and a CHILD SA (i.e. AH or ESP SA). First pair of messages are called IKE_SA_INIT and the second pair is called IKE_AUTH. IKE_SA_INIT establishes a shared secret, which is used as an input to derive the key which is used for securing further messages. The IKE_AUTH messages authenticate the previous messages.In IKE exchange, it is the responsibility of the initiator to ensure reliable delivery of messages. If the response is not delivered within a time frame, then the initiator needs to send the query again.

IKE_SA_INIT:
Initiator sends a message which contains a Header(HDR), a SA payload(SA_{i1}), a Diffie Hellman key(KE_i), an initiator's nonce(N_i).
The SA_{i1} payload contains a mix and match list of cryptographic protocols, which are acceptable to the initiator.

The responder responds with the response which has a HDR, SA_{i2}, KE_r and N_r. If the responder needs certain certificates to validate/authenticate the initiator, it can put CERTREQ payload requesting the initiator to send the required certificates in the IKE_AUTH messages.

Through the SA_{i2} payload, the responder chooses one combination from the list sent by the initiator in SA_{i1}.

After this exchange both the parties are able to generate a shared secret called SKEYSEED. This is used to obtain keys for this IKE_SA. These keys are: SK_e_i and SK_a_i for encryption and integrity protection of the messages sent by the initiator, SK_e_r and SK_a_r for the responder and SK_d for use in derivation of the keys for the future child SAs.

The messages from here on are encrypted and integrity protected (represented by SK)

IKE_AUTH:

The initiator sends a message which contains

HDR, SK ID_i, [CERT], [CERTREQ], [ID_r], AUTH, SA_{i2}, TS_i, TS_r.

The initiator sends the details for the establishment of the child SA in the SAi2 payload, claims its identity with IDi, proves it with AUTH payload and optionally sends and requests certificates.

The responder sends a message which contains

HDR, SK ID_r, [CERT], AUTH, SA_{r2}, TS_i, TS_r.

2.2 Authentication in IPSec

IPSec allows four ways to authenticate:

1. Pre-shared Keys
2. Digital Signatures
3. Public Key Encryption
4. External Authentication

For pre shared keys, the auth value is computed as:

"AUTH=prf (prf (Shared secret, "Key pad for IKEv2"), $< messageoctets >$)"

Here key pad for IKEv2 is 17 ASCII characters without the null termination. The shared secret can be of variable length. The keypad is used to provide randomness even if the user chooses a weak password/shared secret.

2.3 DoS resistance in IPSec

Usual approach of IPSec for the prevention of DOS attacks is as follows:

1. An expected attack against IKE is state and CPU exhaustion, where the target is flooded with session initiation requests from forged IP addresses. This attack can be made less effective if an implementation of a responder uses minimal CPU and commits no state to an SA until it knows the initiator can receive packets at the address from which it claims to be sending them. To accomplish this, a responder SHOULD – when it detects a large number of half-open IKE_SAs – reject initial IKE messages unless they contain a Notify payload of type

COOKIE. It SHOULD instead send an unprotected IKE message as a response and include COOKIE Notify payload with the cookie data to be returned. Initiators who receive such responses MUST retry the IKE_SA_INIT with a Notify payload of type COOKIE containing the responder supplied cookie data as the first payload and all other payloads unchanged.

$$Cookie =< VersionIDofSecret > |Hash(N_i|IP_i|SPI_i| < secret >)$$

where $< secret >$ is a randomly generated secret known only to the responder and periodically changed and — indicates concatenation. $< VersionIDofSecret >$ should be changed whenever $< secret >$ is regenerated.

2. There is a Denial of Service attack on the initiator of an IKE_SA that can be avoided if the initiator takes the proper care. Since the first two messages of a SA setup are not cryptographically protected, an attacker could respond to the initiator's message before the genuine responder and poison the connection setup attempt. To prevent this, the initiator MAY be willing to accept multiple responses to its first message, treat each as potentially legitimate, respond to it, and then discard all the invalid half-open connections when it receives a valid cryptographically protected response to any one of its requests. Once a cryptographically valid response is received, all subsequent responses should be ignored whether or not they are cryptographically valid.

Using pre shared keys for authentication in IKE demands:

1. The pre-shared key SHOULD contain as much unpredictability as the strongest key being negotiated. Not sufficient randomness/entropy is what leads to passwords being not recommended as a pre shared key. But since people may use passwords nevertheless, instead of storing the password in plain text, the IKE implementation can store the value prf (Shared secret, "Key pad for IKEv2"), which could not be used as a password equivalent for protocols other than IKEv2

2. The management interface by which shared secret is provided MUST accept ASCII strings of at least 64 octets and MUST NOT add a null terminator before using them as shared secret. It MUST also accept a HEX encoding of the shared secret

3. If the negotiated prf takes a fixed size key, the shared secret must be of that fixed size

2.4 Use of PISKES KEYS

We can use PISKES keys as pre-shared keys to authenticate the source and the destination. If we can configure the PISKES mechanism to fulfill the properties required for the use of a preshared key , **we can use it as a preshared key for authentication in the regular manner, in IKE_AUTH messages.**

Besides using it in the IKE_AUTH messages for authentication, we can do an authentication check in the first message itself. This can increase DoS resilience. We can add a tag,

$$\text{tag} = \text{MAC}_{K^p{}_{\text{B:S}_\text{B}\text{->A:H}_\text{A}}}(\text{pkt'})$$

to the IKE_SA_INIT where pkt' contains the immutable parts of the IP datagram. This is on the same principle as mentioned in the PISKES paper. Since every new first request is different (at least the nonce changes), this cannot be replayed. This can help in preventing DOS attacks. For this, an option can be made available which helps the user choose whether he wants to use cookies or a tag from the PISKES key.

Using a PISKES key in this manner can provide the following advantages-

1. This will provide a relatively stronger guarantee than using cookies.
2. The responder doesn't have to worry about false half open connections since the source has been verified in the first transaction.
3. The user can still choose to use other means of authentication for IKE_AUTH step to provide a security (for eg. Certificates)

The possible drawback that can occur with using PISKES is that you need to trust that the AS will not misbehave and as mentioned in the PISKES paper, there are solutions to that problem which involve blacklisting.