SSH: the Terrapin attack and Strict KEX

Damien Miller
djm@openssh.com
https://bsky.app/profile/damienmiller.bsky.social

Background

SSH protocol is 1990s cryptography - AFAIK no security proofs (let along machine-checkable ones) for the protocol before standardisation. IMO despite this, it's done pretty well*

Quite a bit of analysis and retrospective proof (this list is **absolutely not exhaustive**):

A Surfeit of SSH Cipher Suites (2016) - Albrecht, Degabriele, Hansen, Paterson

Rigorous specifications of the SSH Transport Layer (2011) - Poll, Schubert

Analysis of the SSH Key Exchange Protocol (2011), Williams

* except CBC cipher modes

Terrapin attack

So we all thought it was safe

Along comes:

> *Terrapin Attack: Breaking SSH Channel Integrity By Sequence Number Manipulation*
>
> Fabian Bäumer, Marcus Brinkmann, and Jörg Schwenk, Ruhr University Bochum

Primer: SSH KEX

Early SSH packetised protocol, after banner exchange done

- Server and client: SSH_MSG_KEXINIT
- Client > server: SSH_MSG_KEX_ECDH_INIT
- Server > client: SSH_MSG_KEX_ECDH_REPLY
- Server and client: SSH_MSG_NEWKEYS

Every packet has a sequence number, which is not reset after key exchange

KEX binding

Binding the KEX output to the connection is done via the exchange hash:

```
H(client banner || server banner ||
  client KEXINIT packet || server KEXINIT packet ||
  server host key ||
  client ECDH public value || server ECDH public value ||
  ECDH shared secret)
```

Calculated by both the client and server, used for server's hostkey signature and to derive symmetric cipher and MAC keys

That captures everything in the early connection, right?

Wrong

The observant among you and/or people who have experience with TLS 1.3 will notice that this is not a *full session transcript* hash

But you can’t send any other messages at this time, right?

Wrong Wrong

`SSH_MSG_IGNORE` - “ignore this message”

`SSH_MSG_DEBUG` - “here’s some text, maybe show the user?”

But these are totally harmless not to include in the exchange hash, they don’t modify the connection state, right?

Right … ?

Wrong Wrong Wrong

These messages *increment the packet sequence number*

The packet sequence number is not reset across KEX

And it's **not checked in any way before KEX completes**

After KEX, it's still mostly implicit

The attack

```
seq=0 SSH_MSG_KEXINIT
seq=1 SSH_MSG_KEX_ECDH_REPLY
seq=2 SSH_MSG_NEWKEYS
seq=3 SSH2_MSG_EXT_INFO
seq=4 SSH2_MSG_SERVICE_ACCEPT
```

The client only starts checking sequence numbers after NEWKEYS

The attack

```
seq=0 SSH_MSG_KEXINIT
seq=1 SSH_MSG_IGNORE
seq=1 SSH_MSG_KEX_ECDH_REPLY
seq=2 SSH_MSG_NEWKEYS
seq=3 SSH2_MSG_EXT_INFO
seq=4 SSH2_MSG_SERVICE_ACCEPT
```

MITM inserts a message before NEWKEYS
MITM deletes one message after NEWKEYS
Client sees identical sequence numbers after NEWKEYS

Impact

The Terrapin attacks allows an on-path adversary (i.e. MITM) the ability to delete an arbitrary number of messages from the start of the encrypted SSH transport session.

Practically, doing more than one will break the SSH connection, as the SSH_MSG_SERVICE_ACCEPT message is needed to start user authentication. No user authentication = connection failure

So the deletion has to be of SSH_MSG_EXT_INFO

SSH_MSG_EXT_INFO

SSH_MSG_EXT_INFO is an extension mechanism introduced in RFC8303

Allows some additional parameters to be communicated in ciphertext and after user authentication has completed.

Interesting things: `server-sig-algs`, `ping@openssh.com`

SSH_MSG_EXT_INFO

`server-sig-algs` - inform the client which signature algorithms are acceptable for public key authentication

`ping@openssh.com` - inform the client that the server supports the PING extension, used by OpenSSH's keystroke timing obfuscation

Impact: possibly downgrade the signature algorithm (e.g from RSA/SHA256 to RSA/SHA1) and/or disable keystroke timing obfuscation.

Complicating the attack - the fine print

The actual attack is not *quite* that simple properties of the cipher+MAC/AEAD matter.

Stateful chaining/keystream modes are safe - deleting messages will desynchronise the keystream (CBC, CTR-EncryptAndMAC)

AES-GCM is safe - uses a separate invocation counter instead of a sequence number

`chacha20-poly1305@openssh.com` and `*-etm@openssh.com` MACs are broken because it used the sequence number as part of the per-packet authentication.

Fixing the attack: Strict KEX

Strict KEX is a minimal set of interventions in SSH kex and the binary packet protocol to avoid the Terrapin attack

1. Terminate the connection if any non-KEX message is received during KEX.
2. Reset the sequence number every time KEX completes

Fixing the attack: Strict KEX

Terminating the connection for non-KEX messages during KEX

Ensures only KEX messages are valid: MITM can’t insert IGNORE

Gives **most** of the benefits of full transcript hashing in key derivation

Not as strong a guarantee, e.g. appending data to existing messages could go unnoticed (SSH implementation dependent) but would be caught by a full transcript hash.

Fixing the attack: Strict KEX

Reset the sequence number every time KEX completes

Ensures that sequence number foolery cannot persist from before KEX to after KEX, or across re-KEX

Activating Strict KEX support

Strict KEX is signalled using a similar hack to how SSH_MSG_EXT_INFO was ~~haeked~~ retrofitted into the SSH protocol

Client appends `kex-strict-c-v00@openssh.com` to the list of KEX algorithms it supports in SSH_MSG_KEXINIT

Server does the same with `kex-strict-s-v00@openssh.com`

Protocol modifications enabled if both ends support them.

Implementation support

Most implementations added support in late 2023

Coordinated disclosure managed by team at Ruhr University Bochum

Supported in Apache Mina, AsyncSSH, Golang x/cypto/ssh, libssh, OpenSSH, Paramiko, PuTTY, russh (this list is not exhaustive)

Closing thoughts

Terrapin is cryptographically interesting, but has limited practical impact

Strict KEX is a complete fix for the Terrapin attack.

Achieved its goals: straightforward implementation, was able to be widely deployed quickly and without incident

Would be good to standardise.

If we ever revise the SSH protocol then making the key exchange use a full session transcript in its hash would great

Thanks