IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN XXXX-XXXX, Vol. 0, No. 0, pp. 1–7. DOI:XXXXXXXX

TLS and SSH Downgrade Attacks and Protections

Paul Cordellier

Hochschule Albstadt-Sigmaringen, Albstadt-Ebingen, Germany, cordellierp@gmail.com

Abstract. As cryptographic systems grow older, vulnerabilities are found in older systems. Unfortunately, some servers or clients aren't updated accordingly. SSL 2.0, a version of TLS' predecessor, is considered deprecated and unsafe since 2011. In 2020, 1.2 million web servers supported this version, and that number decreased to 450 thousand in 2023. [Kop23] So we can infer that the unsafe version SSL 3.0 is still widely used.
For communication software developers, this means that making backwards compatible software is important, so that the user can still access websites that are less maintained. Therefore, a system must be implemented so that both client and server understand the secure tools they can use for the communication to be as safe as possible. In a downgrade attack, this system is exploited so that a more vulnerable mode of operation is used.
This paper will present downgrade attacks present in OpenSSH version 9.4p1 and 9.5p1, and in older TLS version, that enable the attacker to downgrade the communication. This paper also informs of what the industry came up with to fix those issues.

Keywords: Downgrade attack · OpenSSH · TLS · SSL · Man-in-the-middle

1 Introduction

The handshake is a procedure in which two devices or programs attempt to initiate communication and establish their communication parameters. In the following attacks, the client and the server agree on encryption and authentication procedures during the handshake. The attacker plays the role of the man-in-the-middle, altering data during the handshake process.

2 Man-in-the-middle

At its core, the man-in-the-middle secretly relays the communications, and alters them. In the context of our attacks, the goal is to convince the server that the man-in-the-middle is the client, and vice versa. So the attacker doesn't get spotted, most of the packets are sent back to their intended devices. Here are examples of man-in-the-middle attacks:

- **ARP Spoofing**: the ARP (Address Resolution Protocol) links the MAC address and the IP address of the device. The scope of this attack is on a local network. The attacker sends lot of fake ARP information, to convince the victims that the data should be sent to them. This is how the man-in-the-middle fakes its identity. [Nid19] Ethernet is a protocol that was designed without any authentication technology. For example, LAN is based on this protocol, so it is relatively easy to perform this attack on this kind of network. [Gib05]

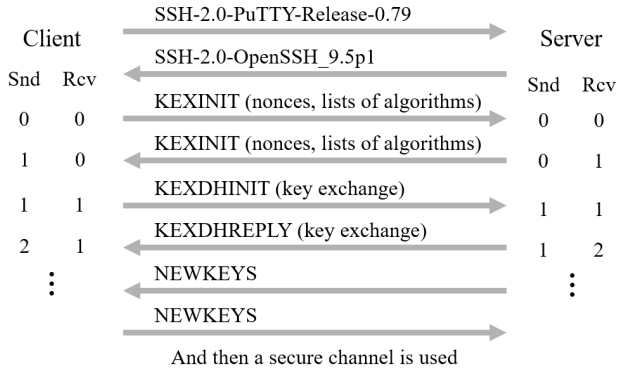


Figure 1: Typical SSH Handshake before the secure channel is used. The numbers on the sides count the number of packets sent and received.

- **DNS Spoofing**: The attacker exploits a vulnerability in a DNS server and injects fake DNS entries. If it works, the domain name will redirect to the attacker's device. This attack often has a larger scale and is harder to perform. [Nid19]
- **Evil twin attack**: The attacker creates a fake Wi-Fi access point, hoping that victims will connect to it. The attacker should be able to read the data of this fake access point and send any requests they want. This attack can be set up in places with public Wi-Fi, like airports or coffee shops. [Moh24]
- **Man in the Browser**: The victim's browser is exploited, and the attacker can take the position of a man-in-the-middle. This can be set up through a malware or unsafe browser extensions. [Gri25]

3 SSH handshake

The handshake is displayed on Figure 1. The Binary Packet Protocol (BPP) is used to perform the SSH handshake. Most information on this paper about the SSH handshake and SSH downgrade is from [FB24]. The main purpose of the handshake is to exchange information to set up the secure channel. In the secure channel, each packet sent is encrypted and with Message Authentication Code (MAC). This MAC is verified by the receiver to make the exchange safer. Here is a description of the following steps of the handshake:

1. Both parties start by exchanging their versions.
2. The KexInit messages contain nonces. (In cryptography, a nonce is a random generated number, that make the encrypted message harder to decrypt, because it is less predictable.) Those messages also contain lists of algorithms, e.g. two for the encryption (for both directions), one for the MAC. Which algorithm is actually used depends on the order of both lists.
3. The KexDhInit and KexDhReply messages are used to perform a finite-field Diffie-Hellman key exchange, or an alternative method to exchange keys.
4. The NewKeys are exchanged.
5. After that, the packages are sent through the secure channel.

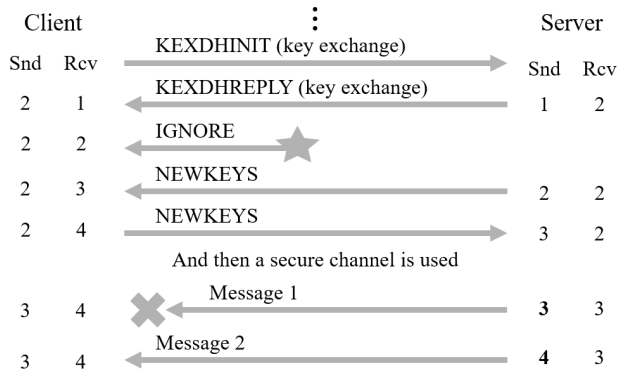


Figure 2: A basic prefix truncation in which a man-in-the-middle sends one IGNORE packet to the client during the handshake. The numbers in **bold** are used for the MAC computation.

Once in the secure channel, each packet is encrypted and has a MAC. This is a way to add verification of the packets by the client and the server. To prevent a man-in-the-middle to reorder the packets, both client and server count the number of packets received and sent, with the SND and RVC counters. The SND counter is used in the MAC computation. If a man-in-the-middle deletes a packet, a peer will realize that there is an offset in the count and will notify the other peer.

But before the secure channel, no encryption, MAC or authentication is used, and this can be exploited. The section Section 5 will talk about vulnerabilities in ChaCha20-Poly1305 and CBC-EtM, different encryption modes.

4 Prefix truncation attack

A man-in-the-middle can try to offset the counters during the SSH handshake mentioned above in Section 3. Because the secure channel has not started yet, the attacker can simply send raw a message, say the IGNORE message, and perform a prefix truncation. In the example given in Figure 2, the man-in-the-middle sends the message to the client; once the secure channel is set up, the first packet sent by the server is ignored by the client, because the number SND decoded from its MAC does not match the client's RCV counter.

In this example, the purpose of the IGNORE message is to protect against traffic analysis, but it is not supposed to have an effect during the handshake.

The way the attack succeeds highly depends on the chosen encryption mode. As soon as the secure channel starts, the way the encryption works will define how likely it is that one of the peers will detect an error.

4.1 With ChaCha20-Poly1305

ChaCha20 is a stream cipher that encrypts the packet. Then, the Poly1305 authenticator uses the encrypted packet and another key generated by ChaCha20 to create the MAC. This encryption mode only has, as an input, the packet that needs to be decrypted and the sequence number (RCV or SND). [YN18] [FB24] This means that the deletion of the first packet EXTINFO will not influence the way the other packets will be decoded. Therefore, truncation attacks are perfectly exploitable for this encryption mode.

4.2 With CBC-EaM

In this encryption mode, the whole packet is encrypted with CBC (Cypher Block Chaining) and the MAC is generated with EaM (Encrypt and MAC). This means that in order to generate the MAC, EaM uses the unencrypted packet length, the packet encryption and the sequence number (Rcv or Snd). The main problem is that, because of the CBC encryption, each packet is chained, so deleting one alters the decryption of every following packet. With EaM, the MAC verification of a corrupted packet fails very often, preventing the attacker from reaching their goeals.

4.3 With CBC-EtM

Here, the payload is still encrypted with CBC (Cypher Block Chaining) but the MAC is generated with EtM (Encrypt then MAC). So, in order to generate the MAC, EtM use the sequence number, the unencrypted packet length and the encrypted packet.

This encryption mode has the same effect as CBC-EaM, but this time the corrupted messages result in an error much less often. The way the packet is analyzed by the receiver can give results such as:

- The packet is too corrupted and the connection is closed.
- The packet has the right padding, but the ID of the message is not recognized, so an Unimplemented response is sent. Using the same notation as in [FB24], we will call this type of corruption *Evasively Corrupt*.

The likelihood of each result very much depends on the implementation.

If the attacker wants to delete N packets, they can choose to delete N - 1 packets using the technique of offsetting the sequence number already mentioned in the beginning of this section. There is a chance (depending, again, on the implementation) that the N-th packet will be *Evasively Corrupt*, and therefore ignored by the receiver. In this case, the attack is successful.

5 Extension downgrade attack on OpenSSH 9.4p1 and 9.5p1

The extension downgrade attack consists of applying a prefix truncation to remove the ExtInfo message that is sent in the secure channel. This will disable most extensions, including:

- **Keystroke timing countermeasures**: In SSH interactive mode, every keystroke that a user types is sent to the remote machine immediately after the key is pressed, which leaks the inter-keystroke timing information of user's typing. Without the keystroke timing countermeasures, this makes side-channel attacks possible. [DXS01]
- Disabling the **server-sig-algs** extension can lead to worse authentication requests. When this is extension enabled, the authentication is done through a public key algorithm. Otherwise, the client will try to authenticate through trial and error, which makes the communication more vulnerable to the attacker. [Bid18]

This first ExtInfo package contains information to initiate an extension negotiation between the peers. Other ExtInfo messages can be sent but, in these versions, those ones can not start the negotiation process again.

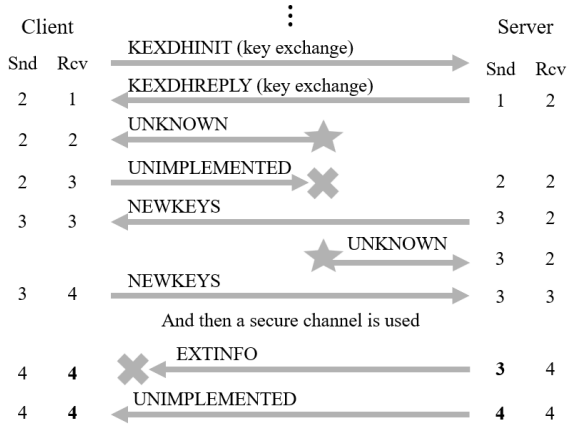


Figure 3: Extension downgrade attack on OpenSSH 9.4p1 or 9.5p1, with the encryption mode CBC-EtM

5.1 With ChaCha20-Poly1305

This attack works well with this encryption mode (for reasons explained in Subsection 4.1), so we simply have to perform the attack shown in Section 4 and Figure 2. In this attack, the first message of the secure channel is ExtInfo.

5.2 With CBC-EtM

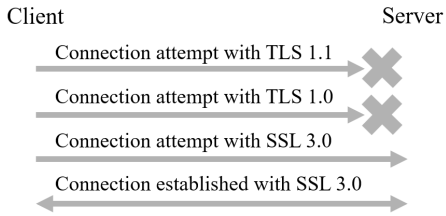
The attack with this encryption mode is presented in Figure 3. In that situation, the aim is to infect the ExtInfo so that it becomes *Evasively Corrupt*, as described in Subsection 4.3. To offset the count, during the handshake, an unknown message is sent to the client, and its Unimplemented response deleted by the attacker. This way, the MAC verification of the ExtInfo fails. Another unknown packet is sent to the server, but this time, the Unimplemented response is sent in the secure channel. The attacker hopes that this response is only *Evasively Corrupt*, so that the neither peer notices anything. The success rate is low, only around to 3%.

One way to improve the success rate is by replacing the unknown message sent to the server with a Ping message, which will have its answer Pong sent in the secure channel. Because this message is smaller, the likelihood that it's *Evasively Corrupt* increases to approximately 80%.

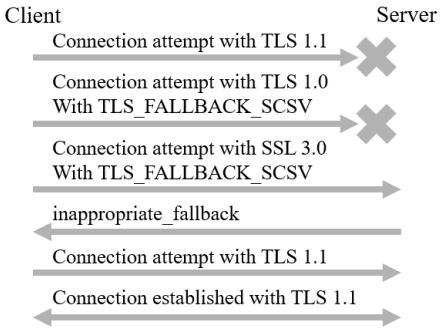
6 TLS downgrade attack

A basic TLS downgrade attack is shown in Figure 4a. The client attempts to connect to a server, but the connection fails because of the man-in-the-middle (in other cases, the connection can also fail because of a network error). In older procedures, the client would try to connect with older protocols, like older TLS versions and the older SSL versions. That means that the man-in-the-middle, just by intercepting certain packets, could trigger a version downgrade [KFT24].

This downgrade attack is made possible only if TLS_FALLBACK_SCSV is not sent by the client [AL15]. More information about this message can be found in section Section 7.



(a) Basic TLS downgrade attack, where the connection attempts are deleted by a man-in-the-middle.



(b) Schema of how TLS_FALLBACK_SCSV helps prevent a downgrade attack. The server here recognizes the FALLBACK SCSV message. Here, the server supports a better version, so it returns the inappropriate_fallback message.

Figure 4: Schemas about TLS downgrade attacks.

6.1 Downgraded to SSL 3.0

When the communication is downgraded to SSL 3.0, and if the CBC mode is enabled, the communication becomes vulnerable to a POODLE attack (Padding Oracle On Downgraded Legacy Encryption). This is made possible because the way the encryption functions makes it vulnerable.

Similarly to the way CBC-EtM or CBC-EaM works for SSH (explained in Subsection 4.2 and Subsection 4.3), a MAC is generated during encryption, and the padding is checked during the decryption. But in the algorithms of these subsections, the minimum size of the padding is 4 [FB24], while in SSL 3.0, the padding can be between the cipher's block length and zero [AOF11].

In this context, the oracle can be the server or the client. The way the implementation of SSL reacts to a given block gives the attacker information about if the padding is valid or not. To start the padding oracle attack, the attacker asks the oracle to find a ciphertext with the smallest. Because the padding is put at the end of the block, the information that one ciphertext has valid padding helps us determine the last byte. With this information, we can try to evaluate the second to last byte, and repeat. This attack doesn't need the initialization vector to work.

With a minimum padding of 4, the likelihood that the oracle marks the cipher as valid is close to one out of 255^4. In our case, the padding can be one or zero, so the attack is much more likely to succeed.

7 TLS downgrade protection

This specific downgrade protection is displayed in Figure 4b. To avoid downgrade attacks, and on the version TLS 1.0, 1.1 and 1.2, the client sends the TLS_FALLBACK_SCSV message each time it attempts a connection with a downgraded version [AL15]. A man-in-the-middle can not only remove this message from the packet because the handshake is cryptographically protected. But it still mostly does not break backward compatibility, because older TLS and SSL versions just ignore the message.

When a server receives a connection attempt with the TLS_FALLBACK_SCSV message, it checks its supported versions. If a better connection is possible, the server returns the inappropriate_fallback message. The client then restarts connections with better versions. [AL15]

TLS 1.3 uses a similar mechanism, where the value DOWNGRD01 or DOWNGRD00 must be sent to older TLS or SSL versions. [HL21]

References

[AL15] Bodo Moeller Adam Langley. Tls fallback signaling cipher suite value (scsv) for preventing protocol downgrade attacks. *RFC editor*, 2015.

[AOF11] Paul C. Kocher Alan O. Freier, Philip Karlton. The secure sockets layer (ssl) protocol version 3.0. *RFC editor*, 2011.

[Bid18] Denis Bider. Extension negotiation in the secure shell (ssh) protocol. *RFC editor*, 2018.

[DXS01] Xuqing Tian Dawn Xiaodong Song, David Wagner. Timing analysis of keystrokes and timing attacks on ssh. *University of California, Berkeley*, 2001.

[FB24] Jörg Schwenk Fabian Bäumer, Marcus Brinkmann. Terrapin attack: Breaking ssh channel integrity by sequence number manipulation. *terrapin-attack.com*, 2024.

[Gib05] Steve Gibson. Arp cache poisoning. *Gibson Research Corporation*, 2005.

[Gri25] Shiran Grinberg. Man-in-the-browser attacks. *Cynet*, 2020 (updated in 2025).

[HL21] Yonghwi Kwon Hyunwoo Lee, Doowon Kim. Tls 1.3 in practice: How tls 1.3 contributes to the internet. *ACM Digital Library*, 2021.

[KFT24] Sze Yiu Chau Ka Fun Tang, Ka Lok Wu. Investigating tls version downgrade in enterprise software. *CODASPY*, 2024.

[Kop23] Jan Kopriva. After 28 years, sslv2 is still not gone from the internet... but we're getting there. *Internet Storm Center*, 2023.

[Moh24] Remya Mohanan. What is an evil twin attack? definition, detection, and prevention best practices. *spiceworks*, 2024.

[Nid19] Tomasz Andrzej Nidecki. All about man-in-the-middle attacks. *Acunetix*, 2019.

[YN18] Adam Langley Yoav Nir. Chacha20 and poly1305 for ietf protocols. *RFC editor*, 2018.