

Informationen zu Secure Shell Packet Exchange

Inhalt

[Einleitung](#)

[Voraussetzungen](#)

[Anforderungen](#)

[Verwendete Komponenten](#)

[SSH-Protokoll](#)

[SSH-Austausch](#)

[Zugehörige Informationen](#)

Einleitung

Dieses Dokument beschreibt den Austausch auf Paketebene während der Secure Shell (SSH)-Aushandlung.

Voraussetzungen

Anforderungen

Cisco empfiehlt, dass Sie über Kenntnisse folgender grundlegender Sicherheitskonzepte verfügen:

- Authentifizierung
- Vertraulichkeit
- Integrität
- Wichtige Methoden für den Austausch

Verwendete Komponenten

Dieses Dokument ist nicht auf bestimmte Hardwareversionen beschränkt.

Die Informationen in diesem Dokument beziehen sich auf Geräte in einer speziell eingerichteten Testumgebung. Alle Geräte, die in diesem Dokument benutzt wurden, begannen mit einer gelöschten (Nichterfüllungs) Konfiguration.

SSH-Protokoll

Das SSH-Protokoll ist eine Methode zur sicheren Remote-Anmeldung von einem Computer zum anderen. SSH-Anwendungen basieren auf einer Client-Server-Architektur, die eine SSH-Client-Instanz mit einem SSH-Server verbindet.

SSH-Austausch

1. Der erste Schritt von SSH wird als Identification String Exchange.

a. Der Client erstellt ein Paket und sendet es an den Server, der Folgendes enthält:

- Version des SSH-Protokolls
- Software-Version

```
323 5.946818 10.65.54.8 10.106.51.72 SSHv2 82 Client: Protocol (SSH-2.0-PuTTY_Release_0.76)
> Frame 323: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface 0
> Ethernet II, Src: Cisco_3c:7a:00 (00:05:9a:3c:7a:00), Dst: Cimsys_33:44:55 (00:11:22:33:44:55)
> Internet Protocol Version 4, Src: 10.65.54.8, Dst: 10.106.51.72
> Transmission Control Protocol, Src Port: 56127, Dst Port: 22, Seq: 1, Ack: 1, Len: 28
v SSH Protocol
  Protocol: SSH-2.0-PuTTY_Release_0.76
```

Die Clientprotokollversion ist SSH2.0, und die Softwareversion ist Putty_0.76.

b. Der Server reagiert mit seinem eigenen Identification String Exchange, einschließlich der Version des SSH-Protokolls und der Softwareversion.

```
326 6.016955 10.106.51.72 10.65.54.8 SSHv2 73 Server: Protocol (SSH-2.0-Cisco-1.25)
> Frame 326: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface 0
> Ethernet II, Src: Cimsys_33:44:55 (00:11:22:33:44:55), Dst: Cisco_3c:7a:00 (00:05:9a:3c:7a:00)
> Internet Protocol Version 4, Src: 10.106.51.72, Dst: 10.65.54.8
> Transmission Control Protocol, Src Port: 22, Dst Port: 56127, Seq: 1, Ack: 29, Len: 19
v SSH Protocol
  Protocol: SSH-2.0-Cisco-1.25
```

Die Protokollversion des Servers ist SSH2.0, die Softwareversion ist Cisco1.25.

2. Der nächste Schritt ist Algorithm Negotiation. In diesem Schritt handeln sowohl Client als auch Server diese Algorithmen aus:

- Schlüsselaustausch
- Verschlüsselung
- HMAC (Hash-based Message Authentication Code)
- Komprimierung

1. Der Client sendet eine Key Exchange Init-Nachricht an den Server, in der er die unterstützten Algorithmen angibt. Die Algorithmen werden nach Präferenz geordnet aufgelistet.

```
329 6.021990 10.65.54.8 10.106.51.72 SSHv2 238 Client: Key Exchange Init
> Frame 329: 238 bytes on wire (1904 bits), 238 bytes captured (1904 bits) on interface 0
> Ethernet II, Src: Cisco_3c:7a:00 (00:05:9a:3c:7a:00), Dst: Cimsys_33:44:55 (00:11:22:33:44:55)
> Internet Protocol Version 4, Src: 10.65.54.8, Dst: 10.106.51.72
> Transmission Control Protocol, Src Port: 56127, Dst Port: 22, Seq: 1101, Ack: 20, Len: 184
> [3 Reassembled TCP Segments (1256 bytes): #327(536), #328(536), #329(184)]
v SSH Protocol
  SSH Version 2 (encryption:aes256-ctr mac:hmac-sha2-256 compression:none)
    Packet Length: 1252
    Padding Length: 11
  v Key Exchange
    Message Code: Key Exchange Init (20)
    > Algorithms
```

Schlüsselaustausch-Initialisierung

```

Algorithms
Cookie: 47a96215afc92003180b60342970a105
kex_algorithms length: 315
kex_algorithms string [truncated]: curve448-sha512,curve25519-sha256,curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,dif
server_host_key_algorithms length: 123
server_host_key_algorithms string: rsa-sha2-512,rsa-sha2-256,ssh-rsa,ssh-ed448,ssh-ed25519,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,ssh-dss
encryption_algorithms_client_to_server length: 189
encryption_algorithms_client_to_server string: aes256-ctr,aes256-cbc,rijndael-cbc@lysator.liu.se,aes192-ctr,aes192-cbc,aes128-ctr,aes128-cbc,chacha20-poly1305
encryption_algorithms_server_to_client length: 189
encryption_algorithms_server_to_client string: aes256-ctr,aes256-cbc,rijndael-cbc@lysator.liu.se,aes192-ctr,aes192-cbc,aes128-ctr,aes128-cbc,chacha20-poly1305
mac_algorithms_client_to_server length: 155
mac_algorithms_client_to_server string: hmac-sha2-256,hmac-sha1,hmac-sha1-96,hmac-md5,hmac-sha2-256-etm@openssh.com,hmac-sha1-etm@openssh.com,hmac-sha1-96-etm
mac_algorithms_server_to_client length: 155
mac_algorithms_server_to_client string: hmac-sha2-256,hmac-sha1,hmac-sha1-96,hmac-md5,hmac-sha2-256-etm@openssh.com,hmac-sha1-etm@openssh.com,hmac-sha1-96-etm
compression_algorithms_client_to_server length: 26
compression_algorithms_client_to_server string: none,zlib,zlib@openssh.com
compression_algorithms_server_to_client length: 26
compression_algorithms_server_to_client string: none,zlib,zlib@openssh.com

```

Vom Client unterstützte Algorithmen

b. Der Server antwortet mit einer eigenen Nachricht zur Initialisierung von Schlüsselaustausch, in der die unterstützten Algorithmen aufgeführt sind.

c. Da diese Nachrichten gleichzeitig ausgetauscht werden, vergleichen beide Parteien ihre Algorithmus-Listen. Wenn die von beiden Seiten unterstützten Algorithmen übereinstimmen, fahren sie mit dem nächsten Schritt fort. Wenn keine exakte Übereinstimmung vorliegt, wählt der Server den ersten Algorithmus aus der Liste des Clients aus, den er ebenfalls unterstützt.

d. Wenn sich Client und Server nicht auf einen gemeinsamen Algorithmus einigen können, schlägt der Schlüsselaustausch fehl.

```

334 6.093250 10.106.51.72 10.65.54.8 SSHv2 366 Server: Key Exchange Init
> Frame 334: 366 bytes on wire (2928 bits), 366 bytes captured (2928 bits) on interface 0
> Ethernet II, Src: Cimsys_33:44:55 (00:11:22:33:44:55), Dst: Cisco_3c:7a:00 (00:05:9a:3c:7a:00)
> Internet Protocol Version 4, Src: 10.106.51.72, Dst: 10.65.54.8
> Transmission Control Protocol, Src Port: 22, Dst Port: 56127, Seq: 20, Ack: 1285, Len: 312
SSH Protocol
  SSH Version 2 (encryption:aes256-ctr mac:hmac-sha2-256 compression:none)
    Packet Length: 308
    Padding Length: 4
    Key Exchange
      Message Code: Key Exchange Init (20)
      Algorithms

```

Serverschlüsselaustausch-Initialisierung

3. Danach gehen beide Seiten in die Key Exchange Phase, um mithilfe des DH-Schlüsselaustauschs einen gemeinsamen geheimen Schlüssel zu generieren und den Server zu authentifizieren:

a. Der Client generiert ein Tastenpaar und Public and Private sendet den öffentlichen DH-Schlüssel im DH Group Exchange Init-Paket. Dieses Schlüsselpaar wird für die Berechnung des geheimen Schlüssels verwendet.

```

337 6.201114 10.65.54.8 10.106.51.72 SSHv2 326 Client: Diffie-Hellman Group Exchange Init
> Frame 337: 326 bytes on wire (2608 bits), 326 bytes captured (2608 bits) on interface 0
> Ethernet II, Src: Cisco_3c:7a:00 (00:05:9a:3c:7a:00), Dst: Cimsys_33:44:55 (00:11:22:33:44:55)
> Internet Protocol Version 4, Src: 10.65.54.8, Dst: 10.106.51.72
> Transmission Control Protocol, Src Port: 56127, Dst Port: 22, Seq: 1309, Ack: 612, Len: 272
SSH Protocol
  SSH Version 2 (encryption:aes256-ctr mac:hmac-sha2-256 compression:none)
    Packet Length: 268
    Padding Length: 6
    Key Exchange
      Message Code: Diffie-Hellman Group Exchange Init (32)
      Multi Precision Integer Length: 256
      DH client e: 1405ab00ff368031363467ad6653967d5a64eac4734e5dc6...
      Padding String: 5c81f2cffc95

```

Client DH Public Key & Diffie-Hellman Group Exchange Init

b. Der Server generiert ein eigenes Public and Private Schlüsselpaar. Er verwendet den öffentlichen Schlüssel des Clients und sein eigenes Schlüsselpaar, um den gemeinsamen geheimen Schlüssel zu berechnen.

c. Der Server berechnet auch einen Exchange-Hash mit folgenden Eingaben:

- Client-Identifizierungszeichenfolge
- Server-Identifizierungszeichenfolge
- Nutzlast des Kunden KEXINIT
- Nutzlast des Servers KEXINIT
- Server Öffentlicher Schlüssel von Host-Schlüsseln (RSA-Schlüsselpaar)
- Öffentlicher Client-DH-Schlüssel
- Öffentlicher Serverschlüssel
- Gemeinsamer geheimer Schlüssel

d. Nach dem Computing des Hash signiert der Server ihn mit seinem privaten RSA-Schlüssel.

e. Der Server erstellt eine Nachricht mit dem Namen DH_Exchange_Reply, die Folgendes enthält:

- RSA - Öffentlicher Serverschlüssel (zur Unterstützung des Clients bei der Serverauthentifizierung)
- DH-Öffentlicher Schlüssel des Servers (zur Berechnung des gemeinsamen geheimen Schlüssels)
- HASH (zur Authentifizierung des Servers und zum Nachweis, dass der Server den gemeinsamen geheimen Schlüssel generiert hat, da der geheime Schlüssel Teil der Hash-Berechnung ist)

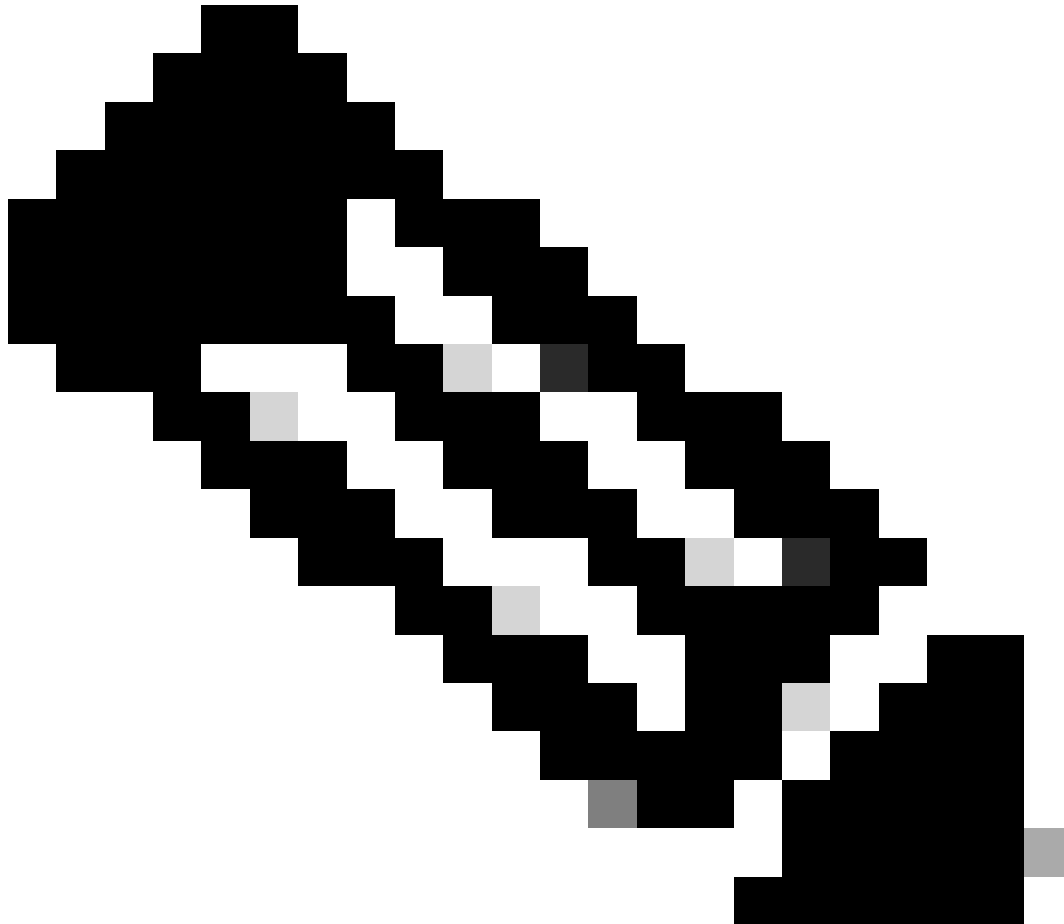
```
343 6.330017 10.106.51.72 10.65.54.8 SSHv2 350 Server: Diffie-Hellman Group Exchange Reply
Internet Protocol Version 4, Src: 10.106.51.72, Dst: 10.65.54.8
Transmission Control Protocol, Src Port: 22, Dst Port: 56127, Seq: 1148, Ack: 1581, Len: 296
[2 Reassembled TCP Segments (832 bytes): #342(536), #343(296)]
SSH Protocol
  SSH Version 2 (encryption:aes256-ctr mac:hmac-sha2-256 compression:none)
    Packet Length: 828
    Padding Length: 8
    Key Exchange
      Message Code: Diffie-Hellman Group Exchange Reply (33)
      KEX host key (type: ssh-rsa)
        Host key length: 279
        Host key type length: 7
        Host key type: ssh-rsa
        Multi Precision Integer Length: 3
        RSA public exponent (e): 010001
        Multi Precision Integer Length: 257
        RSA modulus (N): 0098c7d23c9ababd730f07b5c2aee1e4e51bac67970aa5af...
        Multi Precision Integer Length: 256
        DH server f: 3a17a09995531f12d629a48ab6f25715bc181ea3deb6c6793...
        KEX H signature length: 271
        KEX H signature: 00000007373682d72736100000100691d2c896761bc7481...
        Padding String: 0000000000000000
```

Server DH Public Key & Diffie-Hellman Group Exchange-Antwort

f. Nach dem Empfang von DH_Exchange_Reply berechnet der Client den Hash auf die gleiche Weise und vergleicht ihn mit dem empfangenen Hash, wobei er ihn mit dem öffentlichen RSA-Schlüssel des Servers entschlüsselt.


g. Vor der Entschlüsselung des empfangenen HASH muss der Client den öffentlichen Schlüssel

des Servers überprüfen. Diese Verifizierung erfolgt über ein digitales Zertifikat, das von einer Zertifizierungsstelle (Certificate Authority, CA) signiert wird. Wenn das Zertifikat nicht vorhanden ist, muss der Client entscheiden, ob er den öffentlichen Schlüssel des Servers akzeptiert.



Hinweis: Wenn Sie SSH zum ersten Mal in ein Gerät einbinden, das kein digitales Zertifikat verwendet, werden Sie möglicherweise in einem Popup-Fenster aufgefordert, den öffentlichen Schlüssel des Servers manuell zu akzeptieren. Um zu vermeiden, dass dieses Popup bei jeder Verbindung angezeigt wird, können Sie den Host-Schlüssel des Servers Ihrem Cache hinzufügen.

Warning



Continue connecting to an unknown server and add its host key to a cache?

The server's host key was not found in the cache. You have no guarantee that the server is the computer you think it is.

The server's RSA key details are:

Algorithm: ssh-rsa 2048
 SHA-256: [REDACTED]
 MD5: [REDACTED]

If you trust this host, press Yes. To connect without adding host key to the cache, press No. To abandon the connection press Cancel.

[Copy key fingerprints to clipboard](#)

Yes No Cancel Help

RSA-Schlüssel des Servers

4. Da der gemeinsame geheime Schlüssel jetzt generiert wird, verwenden beide Endgeräte ihn, um folgende Schlüssel abzuleiten:

- Verschlüsselungsschlüssel
- IV-Schlüssel - Dies sind Zufallszahlen, die als Eingabe für symmetrische Algorithmen verwendet werden, um die Sicherheit zu erhöhen
- Integritätsschlüssel

Das Ende des Schlüsselaustauschs wird durch den Austausch der `NEW KEYS` Nachricht signalisiert, die jeden Teilnehmer darüber informiert, dass alle zukünftigen Nachrichten mit diesen neuen Schlüsseln verschlüsselt und geschützt werden.

346	6.330368	10.106.51.72	10.65.54.8	SSHv2	70	Server: New Keys
347	6.365552	10.65.54.8	10.106.51.72	SSHv2	70	Client: New Keys

```

> Frame 346: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
> Ethernet II, Src: Cimsys_33:44:55 (00:11:22:33:44:55), Dst: Cisco_3c:7a:00 (00:05:9a:3c:7a:00)
> Internet Protocol Version 4, Src: 10.106.51.72, Dst: 10.65.54.8
> Transmission Control Protocol, Src Port: 22, Dst Port: 56127, Seq: 1444, Ack: 1581, Len: 16
✓ SSH Protocol
  ✓ SSH Version 2 (encryption:aes256-ctr mac:hmac-sha2-256 compression:none)
    Packet Length: 12
    Padding Length: 10
  ✓ Key Exchange
    Message Code: New Keys (21)
    Padding String: 00000000000000000000
  
```

Neue Schlüssel für Client und Server

5. Der letzte Schritt ist die Serviceanfrage. Der Client sendet ein SSH-Dienstanforderungspaket an den Server, um die Benutzerauthentifizierung zu initiieren. Der Server antwortet mit einer SSH Service Accept-Nachricht, in der der Client zur Anmeldung aufgefordert wird. Dieser Austausch erfolgt über den eingerichteten sicheren Kanal.

Zugehörige Informationen

- <https://www.cisco.com/c/en/us/support/docs/security-vpn/secure-shell-ssh/4145-ssh.html>
- <https://datatracker.ietf.org/doc/html/rfc4253>
- [Technischer Support und Downloads von Cisco](#)

Informationen zu dieser Übersetzung

Cisco hat dieses Dokument maschinell übersetzen und von einem menschlichen Übersetzer editieren und korrigieren lassen, um unseren Benutzern auf der ganzen Welt Support-Inhalte in ihrer eigenen Sprache zu bieten. Bitte beachten Sie, dass selbst die beste maschinelle Übersetzung nicht so genau ist wie eine von einem professionellen Übersetzer angefertigte. Cisco Systems, Inc. übernimmt keine Haftung für die Richtigkeit dieser Übersetzungen und empfiehlt, immer das englische Originaldokument (siehe bereitgestellter Link) heranzuziehen.