了解Secure Shell数据包交换

目录

简介

本文档介绍在安全外壳(SSH)协商期间的数据包级别交换。

先决条件

要求

Cisco建议您了解基本的安全概念：

- 身份验证
- 机密性
- 完整性
- 密钥交换方法

使用的组件

本文档不限于特定硬件版本。

本文档中的信息都是基于特定实验室环境中的设备编写的。本文档中使用的所有设备最初均采用原始（默认）配置。

SSH 协议

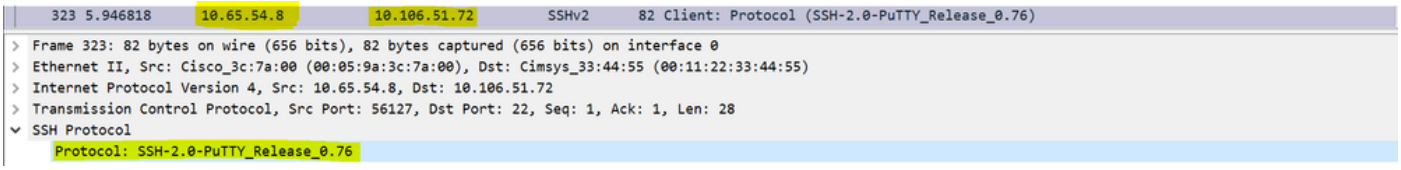
SSH协议是一种保护计算机之间远程登录的方法。SSH应用基于客户端-服务器架构，连接SSH客户端实例与SSH服务器。

SSH交换

1. SSH的第一步称为 Identification String Exchange.

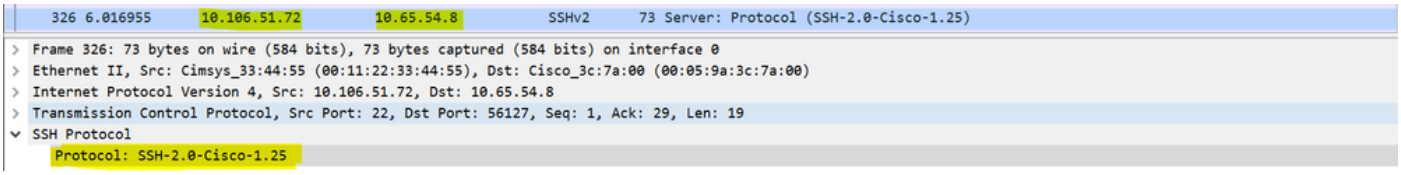
a.客户端构建数据包并将其发送到服务器，该服务器包含：

- SSH协议版本
- 软件版本



客户端协议版本为SSH2.0，软件版本为Putty_0.76。

b.服务器使用其自己的标识字符串交换做出响应，包括其SSH协议版本和软件版本。

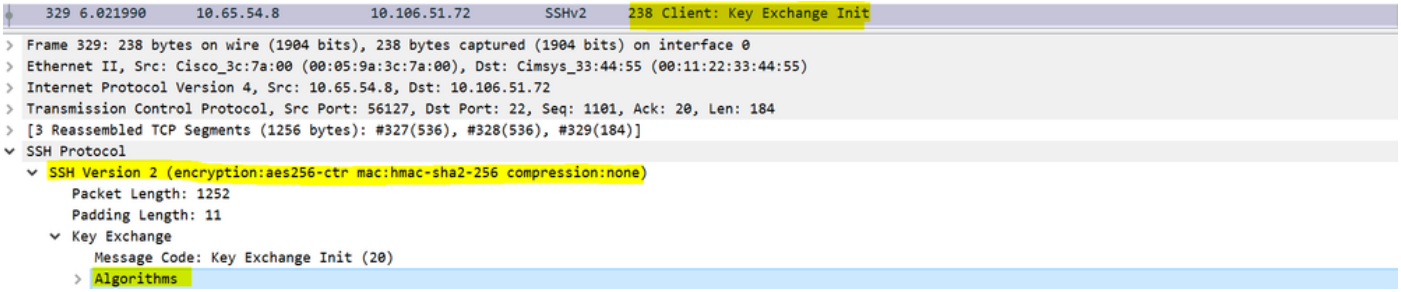


服务器的协议版本为SSH2.0，软件版本为Cisco1.25

2. 下一步是**Algorithm Negotiation.**。在此步骤中，客户端和服务器协商这些算法：

- 密钥交换
- 加密
- HMAC（基于哈希的消息验证码）
- 压缩

1. 客户端向服务器发送Key Exchange Init消息，指定其支持的算法。算法按优先顺序列出。



密钥交换初始化

```
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
```

客户端支持的算法

b.服务器使用其自己的密钥交换初始化消息进行响应，列出其支持的算法。

c.由于这些消息同时交换，双方会比较其算法列表。如果双方支持的算法匹配，则继续下一步。如果没有完全匹配项，服务器将从客户端的列表中选择其也支持的第一个算法。

d.如果客户端和服务器无法就通用算法达成一致，则密钥交换会失败。

```
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
```

服务器密钥交换初始化

3. 此后，双方进入**Key Exchange**阶段，使用DH密钥交换生成共享密钥并对服务器进行身份验证：

a.客户端生成密钥对，Public and Private并在DH组交换初始数据包中发送DH公钥。此密钥对用于计算密钥。

```
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
```

客户端DH公钥和Diffie-Hellman组交换初始化

b.服务器生成自己的Public and Private密钥对。 它使用客户端的公钥和自己的密钥对来计算共享密钥。

c.服务器还使用以下输入计算Exchange散列：

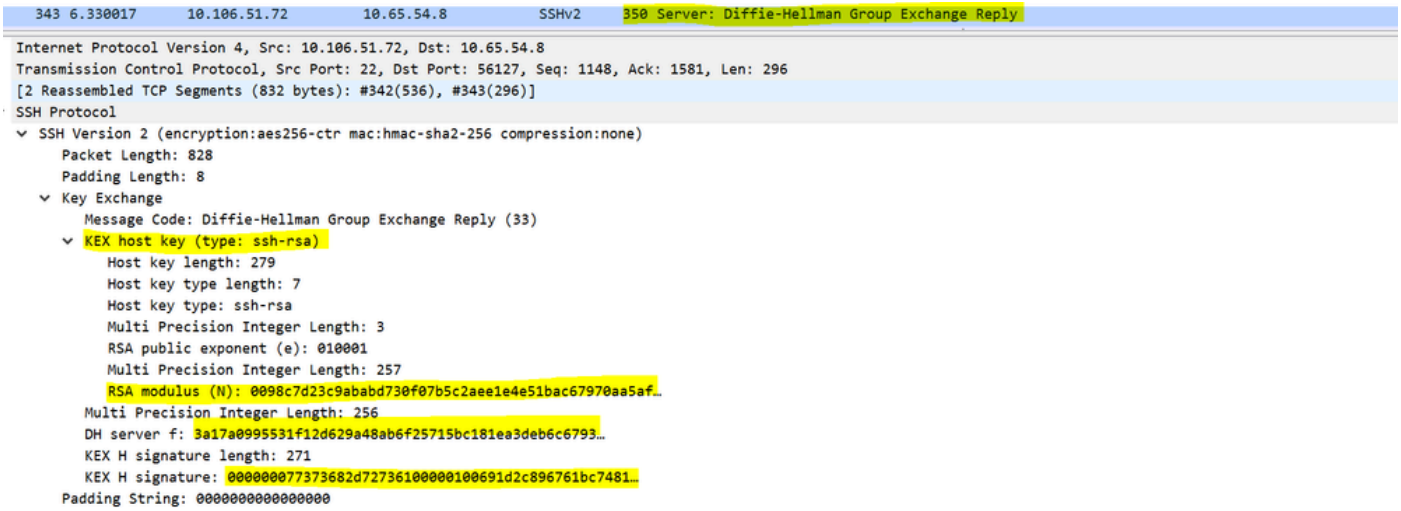
- 客户端标识字符串

- 服务器标识字符串
- 客户端KEXINIT的有效负载
- 服务器KEXINIT的有效负载
- 服务器来自主机密钥的公钥（RSA密钥对）
- 客户端DH公钥
- 服务器DH公钥
- 共享密钥

d.计算散列值后，服务器使用其RSA私钥对其进行签名。

e.服务器构建消息DH_Exchange_Reply，包括：

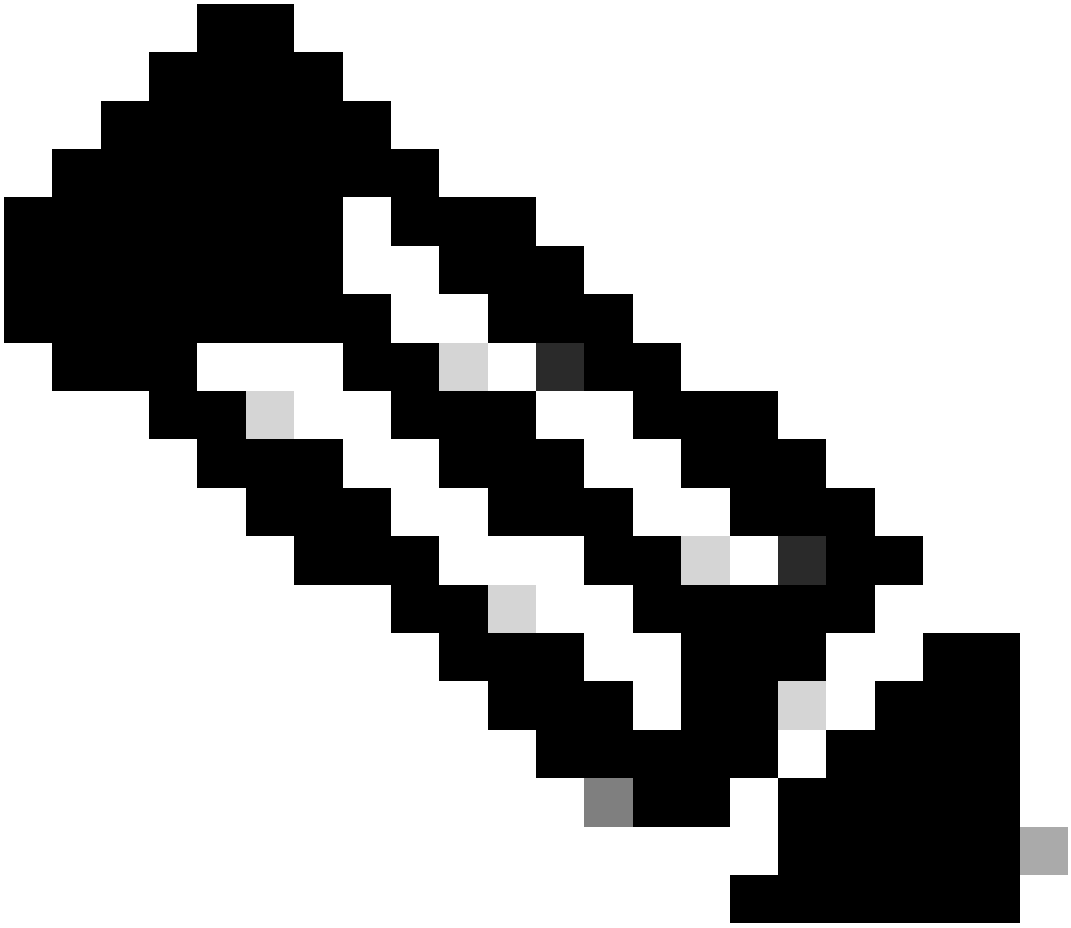
- 服务器的RSA-公钥（帮助客户端对服务器进行身份验证）
- 服务器的DH-公钥（用于计算共享密钥）
- 散列（验证服务器并证明服务器已生成共享密钥，因为密钥是散列计算的一部分）



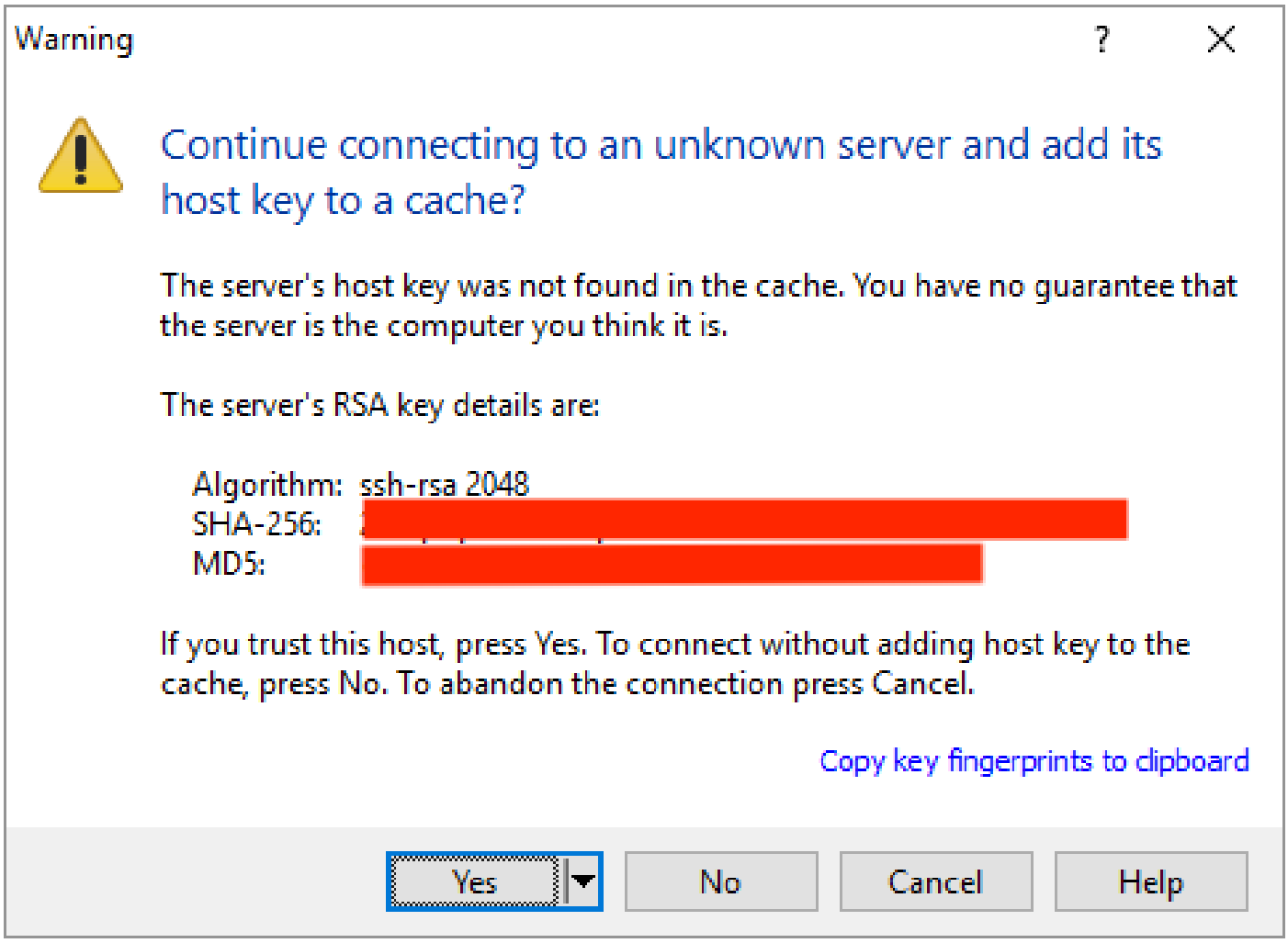
服务器DH公钥和Diffie-Hellman组交换应答

f.收到DH_Exchange_Reply后，客户端以相同方式计算哈希值，并将其与收到的哈希值进行比较，然后使用服务器的RSA公钥对其进行解密。

g.在解密收到的HASH之前，客户端必须验证服务器的公钥。此验证通过证书颁发机构(CA)签名的数字证书完成。如果证书不存在，则由客户端决定是否接受服务器的公钥。



注意：首次通过SSH登录不使用数字证书的设备时，可能会弹出一个窗口，要求您手动接受服务器的公钥。为避免每次连接时都出现此弹出窗口，您可以选择将服务器的主机密钥添加到您的缓存。

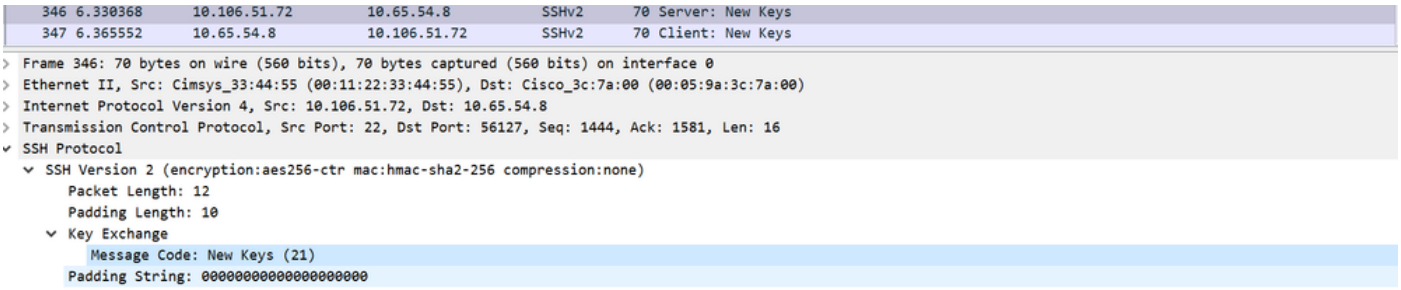


服务器的RSA密钥

4. 由于现在已生成共享密钥，因此两个终端都使用共享密钥来派生这些密钥：

- 加密密钥
- IV密钥-这些是用作对称算法输入的随机数，用于增强安全性
- 完整性密钥

密钥交换的结束由NEW KEYS' 消息的交换来发送，该消息通知每一方，将来的所有消息都将加密并使用这些新的密钥加以保护（例如，RADIUS或TACACS+）。



客户端和服务器新密钥

5. 最后一步是服务请求。客户端向服务器发送SSH服务请求数据包，以启动用户身份验证。服务器以SSH Service Accept（SSH服务接受）消息做出响应，提示客户端登录。此交换通过已建立的安全信道进行。

相关信息

- https://www.cisco.com/c/en/us/support/docs/security-vpn/secure-shell-ssh/4145-ssh.html
- https://datatracker.ietf.org/doc/html/rfc4253
- 思科技术支持和下载

关于此翻译
思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。