Secure Sockets

Part 5 – SSH Handshake

Barak Gonen

Contents – SSH

- Secure Shell goals
- SSH handshake
  - Explore using WSL + Wireshark

SSH

- Secure Shell
- Port 22
- Remote login protocol
- Understanding TLS makes SSH pretty straightforward
- How is SSH used differently from TLS?
  - TLS is used to secure HTTP
    - Generally, only the server needs to be authenticated
  - SSH is used to connect to remote computers (servers)
    - Client must be authenticated
    - Server must run SSH daemon

SSH Handshake Hands On

- Run Wireshark, filter “ssh”
- Use ssh (WSL or OpenSSH) or Putty (www.putty.org) to connect to level1@io.netgarage.org

SSH Handshake

- Client announces protocol version
- Server responds with protocol version

Source	Destination	Protocol	Length	info
192.168.1.221	138.201.80.190	SSHv2	107	Client: Protocol (SSH-2.0-OpenSSH_8.9p1 Ubuntu-3ubuntu0.1)
138.201.80.190	192.168.1.221	SSHv2	105	Server: Protocol (SSH-2.0-OpenSSH_7.4p1 Debian-10+deb9u7)

SSH Handshake – cont.

- Algorithm negotiation – similar to TLS cipher suites
- Both sides announce preferences
  - Key exchange algorithm (KEX)
  - Encryption algorithm
  - MAC algorithm (hash + signature)
  - Compression algorithm
- Server's selection considers the client's preference

```
Key Exchange (method:curve25519-sha256)
    Message Code: Key Exchange Init (20)
  Algorithms
      Cookie: eb017782b03190b71d0c7f7fe5a0ab2d
      kex_algorithms length: 276
      kex_algorithms string [truncated]: curve25519-sha256,curve25519-sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-sha2-nistp521,sntrup761x25519-
      server_host_key_algorithms length: 463
      server_host_key_algorithms string [truncated]: ssh-ed25519-cert-v01@openssh.com,ecdsa-sha2-nistp256-cert-v01@openssh.com,ecdsa-sha2-nistp384-cert-v01@opens
      encryption_algorithms_client_to_server length: 108
      encryption_algorithms_client_to_server string: chacha20-poly1305@openssh.com,aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-gcm@openssh.com
      encryption_algorithms_server_to_client length: 108
      encryption_algorithms_server_to_client string: chacha20-poly1305@openssh.com,aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-gcm@openssh.com
      mac_algorithms_client_to_server length: 213
      mac_algorithms_client_to_server string [truncated]: umac-64-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sha2-512-etm@openss
      mac_algorithms_server_to_client length: 213
      mac_algorithms_server_to_client string [truncated]: umac-64-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sha2-512-etm@openss
      compression_algorithms_client_to_server length: 26
      compression_algorithms_client_to_server string: none,zlib@openssh.com,zlib
      compression_algorithms_server_to_client length: 26
      compression_algorithms_server_to_client string: none,zlib@openssh.com,zlib
```

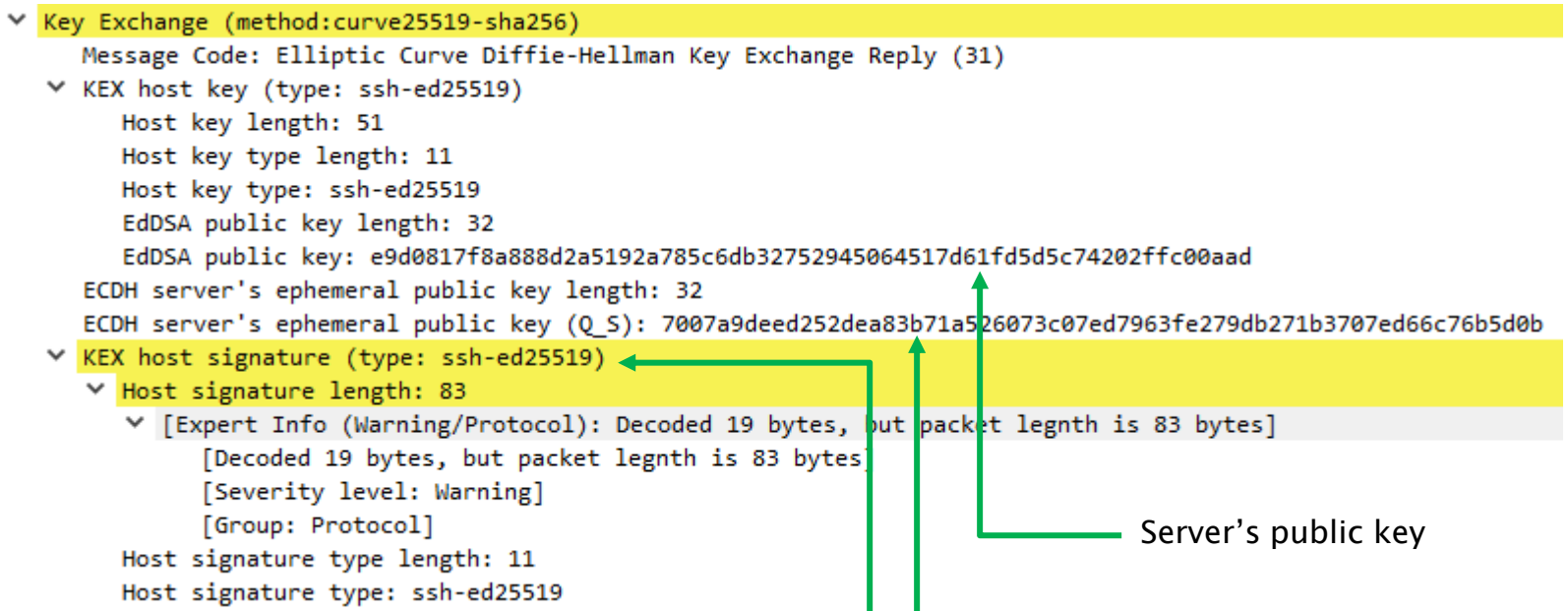
SSH Handshake – cont.

- Sides share key exchange public key
- Server authenticates by sending its public key
  - No PKI!
  - If it the first time the client contacts the server, it will be warned

```
barak@DESKTOP-91SIDPQ:~/.ssh$ ssh level1@io.netgarage.org
The authenticity of host 'io.netgarage.org (138.201.80.190)' can't be established.
ED25519 key fingerprint is SHA256:cLpGrbske5NkXN27QJJgGC4mKj5somd3CnPXxtfW39Y.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

  - The public key should be compared to the key stored in the server
- Server sends MAC of all previous communication
- Client verifies the MAC using the server's public key

Server's Key Exchange Reply



Server's public key

Server's public key exchange data

Signed hash (MAC) of all previous communication

Client Authentication

- Once keys are generated, communications is encrypted
- Clients authenticate using either:
  - Password
  - Client's public & private keys pre-saved on the server