SSHkex Leveraging virtual machine introspection for extracting SSH keys and decrypting SSH network traffic

Lemtaffah Khalil Abdellah
[Redacted]

Outline

- Introduction
- How SSH works?
- SSH Key Exchange
- OpenSSH
- Situation Setup
- Workflow
- Key Extraction
- Decryption

Introduction

Computer communication was and still a crucial process in the world of IT

Introduction

Researchers started developing protocols to ease this scheme

- FTP
- Telnet
- HTTP

Introduction

Remote communication is cool and stuff, but is it really ~~secure?~~

quick answer, no.

Introduction

Remote communication protocols are vulnerable to many dangers, naming:

- Man In The Middle - Sniffing cleartext information
- Weak credentials - Bruteforce (No rate limits enforced)
- Binary exploits - Relevant for privilege escalation
- …

Introduction

SSH (sort of) fixes the previous issues, by encrypting the communication tunnel using exchanged keys.

How SSH works?

Secure Shell (SSH) is a network protocol that enables secure remote access to computers. Using SSH, we can:

- Send remote commands
- Transfer files securely
- Arbitrarily tunnel network traffic (port forwarding)
- …

SSH packet layout

During communication, this is what a packet looks like on SSH transmission:

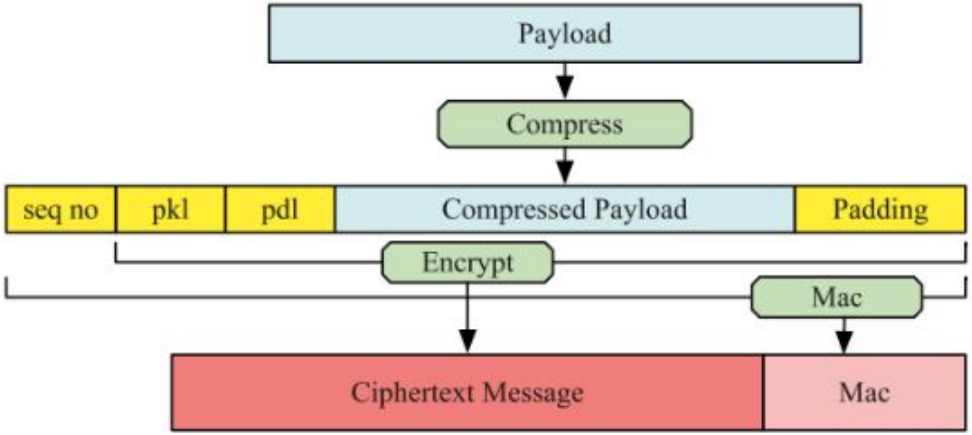


Fig. 1. SSH transport protocol packet layout.

Respects privacy + integrity

SSH Key Exchange (KEX)

SSH handshake is a process in the SSH protocol responsible for negotiating initial trust factors for establishing a secure channel

SSH Key Exchange (KEX) - Version exchange

Exchanging a string between the client and the server to know which SSH version they’re using.

SSH Key Exchange (KEX) - Key exchange (RFC4253)

Sharing a secret in a public space between the 2 entities following this structure:

1. Key A (Initialization vector, client to server)
   $IV_{client2server} = Hash(K, h, \text{“}A\text{”}, session\_id)$
2. Key B (Initialization vector, server to client)
   $IV_{server2client} = Hash(K, h, \text{“}B\text{”}, session\_id)$
3. Key C (Encryption key, client to server)
   $EK_{client2server} = Hash(K, h, \text{“}C\text{”}, session\_id)$
4. Key D (Encryption key, server to client)
   $EK_{server2client} = Hash(K, h, \text{“}D\text{”}, session\_id)$
5. Key E (Integrity key, client to server)
   $IK_{client2server} = Hash(K, h, \text{“}E\text{”}, session\_id)$
6. Key F (Integrity key, server to client)
   $IK_{server2client} = Hash(K, h, \text{“}F\text{”}, session\_id)$

For decryption:

- Extract knowledge about (K, h and sid)
- All the decryption keys

SSH Key Exchange (KEX) - Key exchange (RFC4253)

SSHKex deals with this step of the communication, by using VMI.

OpenSSH

OpenSSH is a software that implements the SSH protocol and includes the server component *sshd* and other tools like *ssh* and *scp*

OpenSSH

It also performs privilege separation, which is relevant for the scope of this research

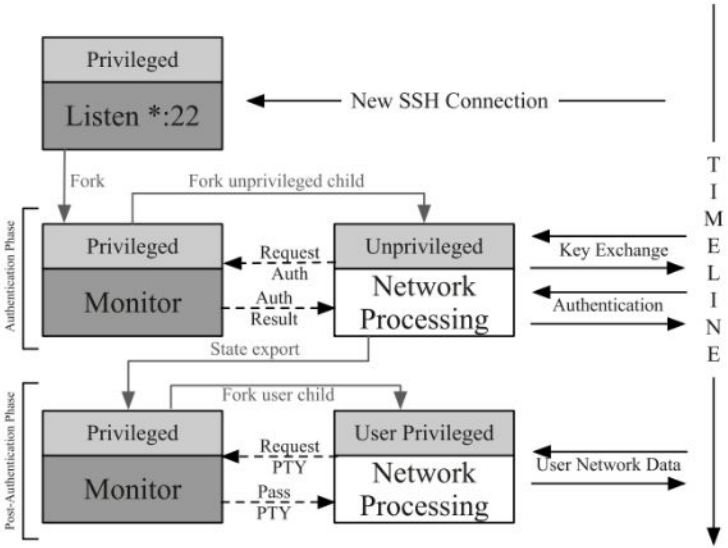


Fig. 2. OpenSSH privilege separation flow.

Situation Setup

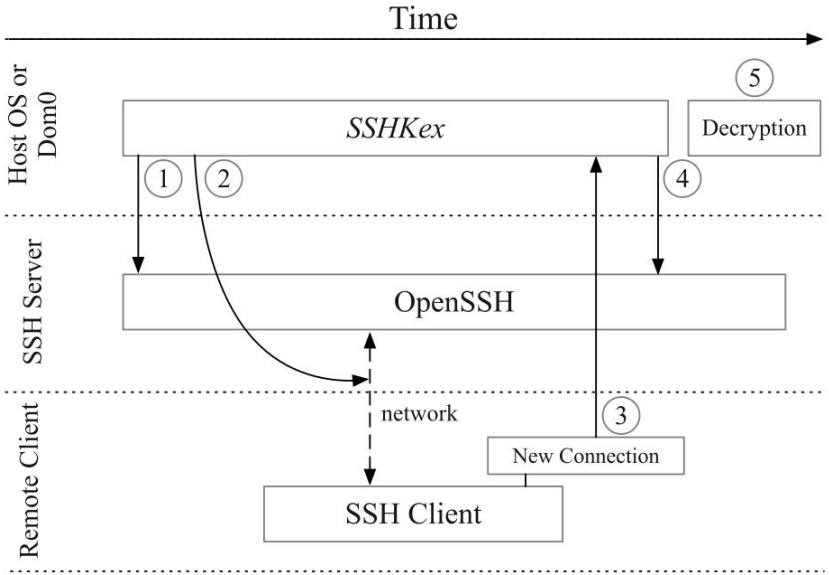
Suppose I have a VM honeypot. An attacker connects to my honeypot. I can capture the SSH traffic between the honeypot and the attacker. I want to decrypt and investigate these traffic.

My investigation should:

1. Stealthy: not inform the attacker
2. Preservation: not modify the system
3. Evidence: obtain verifiable evidence

Workflow

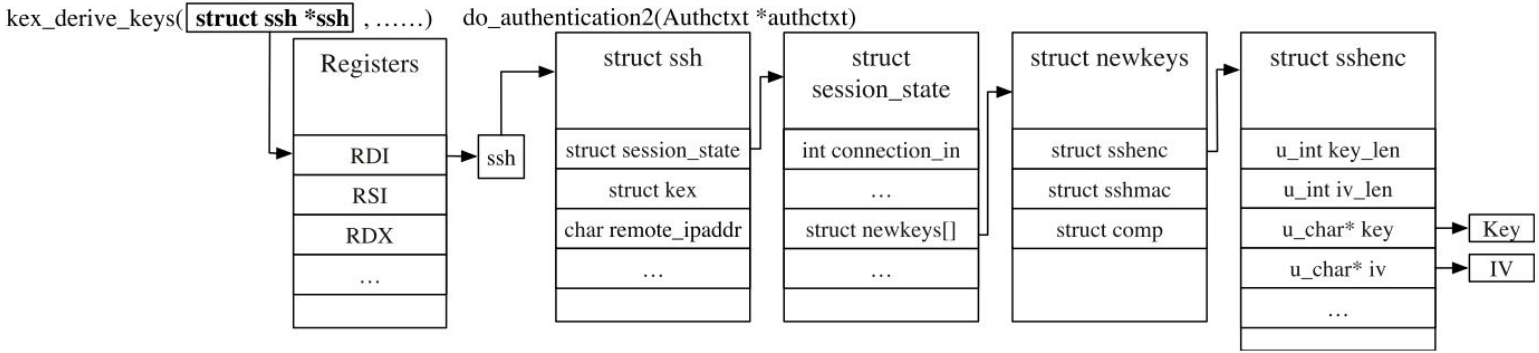
Since my honeypot is running as a VM client, I can use VMI (virtual machine introspection) to dump the memory of the unprivileged SSH subprocess during key exchange.



1. Setup
2. Network traffic capturing
3. Key capture trigger
4. Key extraction
5. Decryption

Key Extraction Procedure

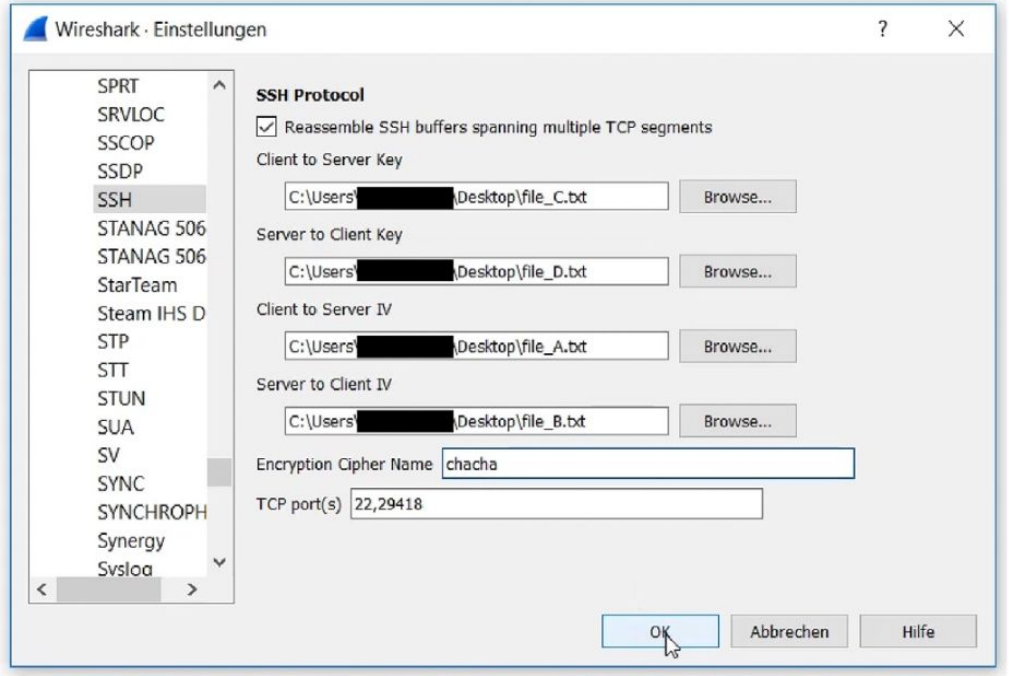
1. Run an unstripped binary of *sshd* on honeypot
2. Set a breakpoint (opcode CC, instruction INT3 “trap to debugger”) at the calling of functions *kex_derive_keys* and *do_authentication2.*
3. When the first breakpoint is hit, extract the memory address of the *ssh* struct, and continue execution.
4. When the second breakpoint is hit, extract two encryption keys from the *ssh* struct



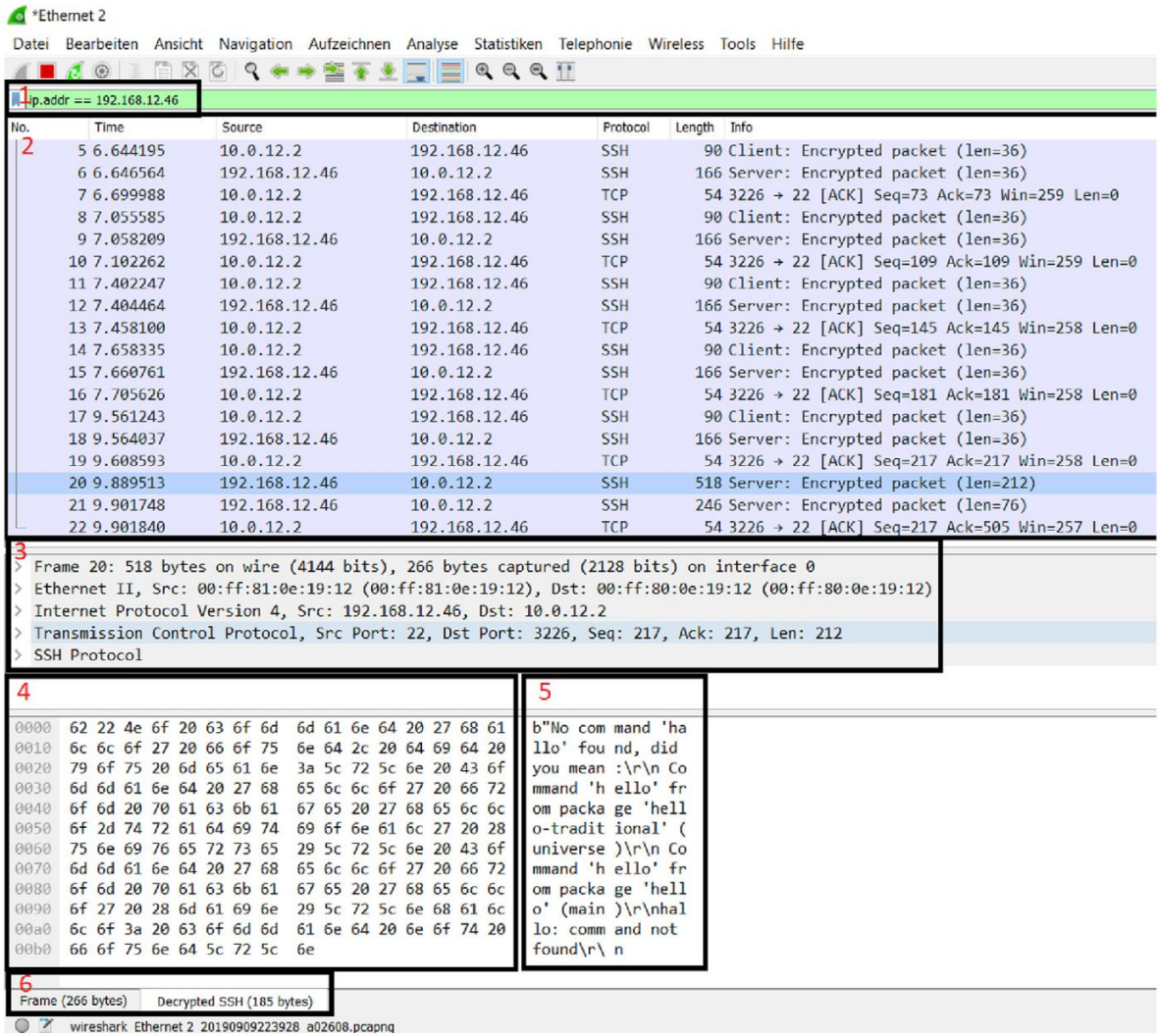
Decryption

```
Key 1 name : chacha20-poly1305@openssh.com
Key 2 name : chacha20-poly1305@openssh.com

key : A --- addr : 5588ac9fcde0 len : 64
93D20734C5F83B5BC81DD1816A042BB4624D1296BC1F8E
93FF02EA133CEC2F6B07922928ABEE64EC59DFC4BB11301
7284D6CAC9548BE9A8C6A9AC4EFB0D0035B
key : B --- addr : 5588ac9f8ac0 len : 64
A196F1B4A72308A8EC41AC72A84FA222CD699EF83F3CDE
7789AB37B47113CED57A67396EF07ACA52FE2B5D96F2E3
D5E3AD982B6F32850472AA4B29C1221CDCD4
key : C --- addr : 5588ac9fed90 len : 64
0B77E10E49CAB165B93CF0C861FEB38C097F3F8F8E6AA7
FFF4D979ABC5F3164A8136FE3C48DFE86CF59710BA7998
FA996B40AC21B50E5B2F6A799F08C7C3E183
key : D --- addr : 5588ac9fa810 len : 64
8F84E1EEFD3E09A4F8A338DDF7E47A59808B0C96020595
DA70F073A5830A7ABE666C4FB6BE2F60DD837C63E2DC8
766B6BF62DFBD77BDA98BD1FCEB791E9442B8
key : E --- addr : 5588ac9fce30 len : 64
D446F576112F2B09C6758D1A8ABD70ED4B945A8298CD6
7F8102ABDED23779132C4AC8CB1483C2D22533AC4BE1B
33F838B97FBBA624900A13D19063C99F8E8693
key : F --- addr : 5588ac9fbd00 len : 64
4FFE294F171DBAD11CA45B289BD2E27C3E48BEF73E1D1
C503CDF726BA0071998C71AD9A2066D2613B130CD3122
6F6FDF927BA38202BCCDFE6D25ED380644186C
```



Decryption


*Ethernet 2
Datei Bearbeiten Ansicht Navigation Aufzeichnen Analyse Statistiken Telephonie Wireless Tools Hilfe
1
ip.addr == 192.168.12.46
2
No. Time Source Destination Protocol Length Info
5 6.644195 10.0.12.2 192.168.12.46 SSH 90 Client: Encrypted packet (len=36)
6 6.646564 192.168.12.46 10.0.12.2 SSH 166 Server: Encrypted packet (len=36)
7 6.699988 10.0.12.2 192.168.12.46 TCP 54 3226 → 22 [ACK] Seq=73 Ack=73 Win=259 Len=0
8 7.055585 10.0.12.2 192.168.12.46 SSH 90 Client: Encrypted packet (len=36)
9 7.058209 192.168.12.46 10.0.12.2 SSH 166 Server: Encrypted packet (len=36)
10 7.102262 10.0.12.2 192.168.12.46 TCP 54 3226 → 22 [ACK] Seq=109 Ack=109 Win=259 Len=0
11 7.402247 10.0.12.2 192.168.12.46 SSH 90 Client: Encrypted packet (len=36)
12 7.404464 192.168.12.46 10.0.12.2 SSH 166 Server: Encrypted packet (len=36)
13 7.458100 10.0.12.2 192.168.12.46 TCP 54 3226 → 22 [ACK] Seq=145 Ack=145 Win=258 Len=0
14 7.658335 10.0.12.2 192.168.12.46 SSH 90 Client: Encrypted packet (len=36)
15 7.660761 192.168.12.46 10.0.12.2 SSH 166 Server: Encrypted packet (len=36)
16 7.705626 10.0.12.2 192.168.12.46 TCP 54 3226 → 22 [ACK] Seq=181 Ack=181 Win=258 Len=0
17 9.561243 10.0.12.2 192.168.12.46 SSH 90 Client: Encrypted packet (len=36)
18 9.564037 192.168.12.46 10.0.12.2 SSH 166 Server: Encrypted packet (len=36)
19 9.608593 10.0.12.2 192.168.12.46 TCP 54 3226 → 22 [ACK] Seq=217 Ack=217 Win=258 Len=0
20 9.889513 192.168.12.46 10.0.12.2 SSH 518 Server: Encrypted packet (len=212)
21 9.901748 192.168.12.46 10.0.12.2 SSH 246 Server: Encrypted packet (len=76)
22 9.901840 10.0.12.2 192.168.12.46 TCP 54 3226 → 22 [ACK] Seq=217 Ack=505 Win=257 Len=0
3
Frame 20: 518 bytes on wire (4144 bits), 266 bytes captured (2128 bits) on interface 0
Ethernet II, Src: 00:ff:81:0e:19:12 (00:ff:81:0e:19:12), Dst: 00:ff:80:0e:19:12 (00:ff:80:0e:19:12)
Internet Protocol Version 4, Src: 192.168.12.46, Dst: 10.0.12.2
Transmission Control Protocol, Src Port: 22, Dst Port: 3226, Seq: 217, Ack: 217, Len: 212
SSH Protocol
4
0000 62 22 4e 6f 20 63 6f 6d 6d 61 6e 64 20 27 68 61
0010 6c 6c 6f 27 20 66 6f 75 6e 64 2c 20 64 69 64 20
0020 79 6f 75 20 6d 65 61 6e 3a 5c 72 5c 6e 20 43 6f
0030 6d 6d 61 6e 64 20 27 68 65 6c 6c 6f 27 20 66 72
0040 6f 6d 20 70 61 63 6b 61 67 65 20 27 68 65 6c 6c
0050 6f 2d 74 72 61 64 69 74 69 6f 6e 61 6c 27 20 28
0060 75 6e 69 76 65 72 73 65 29 5c 72 5c 6e 20 43 6f
0070 6d 6d 61 6e 64 20 27 68 65 6c 6c 6f 27 20 66 72
0080 6f 6d 20 70 61 63 6b 61 67 65 20 27 68 65 6c 6c
0090 6f 27 20 28 6d 61 69 6e 29 5c 72 5c 6e 68 61 6c
00a0 6c 6f 3a 20 63 6f 6d 6d 61 6e 64 20 6e 6f 74 20
00b0 66 6f 75 6e 64 5c 72 5c 6e
5
b"No com mand 'ha
llo' fou nd, did
you mean :\r\n Co
mmand 'h ello' fr
om packa ge 'hell
o-tradit ional' (
universe)\r\n Co
mmand 'h ello' fr
om packa ge 'hell
o' (main)\r\nhal
lo: comm and not
found\r\ n
6
Frame (266 bytes)
Decrypted SSH (185 bytes)
wireshark_Ethernet 2_20190909223928_a02608.pcapng

Overhead

50-90 millisecond overhead

Not noticeable by the attacker, especially if the attacker is behind many proxies or using Tor.