

第11章实验报告

1 实验一：使用私钥访问SSH服务器

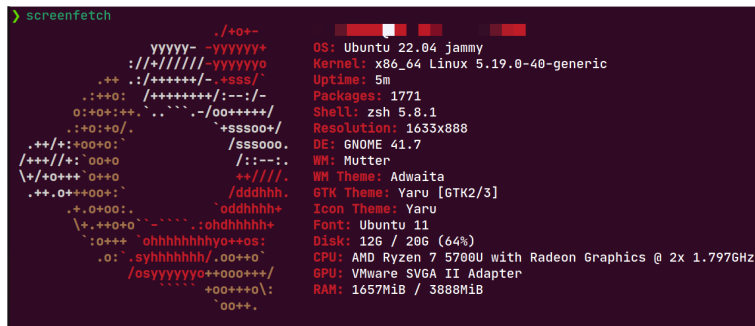
1.1 实验原理

利用非对称加密算法生成一对公私钥对。私钥由自己保管，公钥对外公开，当用公钥对数据加密时，只有对应的私钥才能解密；用私钥对数据签名时，也只有对应的公钥才能验证。利用这一特性可以实现远程服务器对用户身份的认证。

用户可提前将公钥上传至服务器，当用户发起登陆请求时，用户用私钥对服务器发来的随机串进行签名，将签名结果返回给服务器，服务器收到后用公钥进行验证，如果一致则用户身份验证通过允许登陆。

1.2 实验环境

- 使用VMware安装虚拟机Ubuntu22.04并将软件源换成清华源

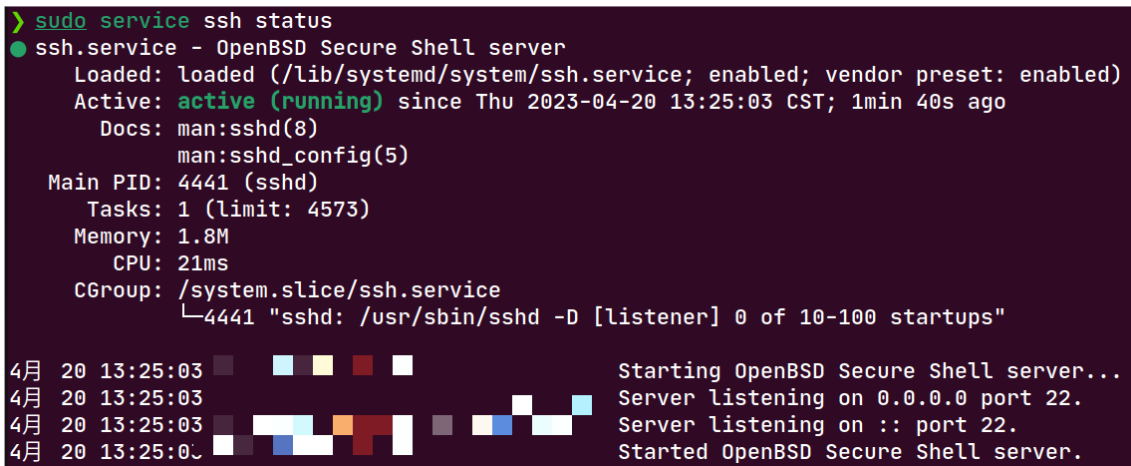


```
screenfetch
      ./+o+/-
      yyyyy~ -yyyyyy+
      ://+///// -yyyyyyo
      .++ :/++++++/- .+sss/^
      .!+o+ :/++++++/-!-:-!/-
      o!+o+!++ .+oooo+/-
      .!+o+!+o/. +sssoo+/
      .++/+!+oo+o!^ /sssooo.
      /+++//+! 0o+o /!:-:-!
      \+/+o+++ 0+o+ ++//!/.
      .++ .o+++oo+!^ /dddhhh.
      .+ .o+oo!^ .oddhhhh+
      \+.+o+o+!^ .:ohdhhhh+
      :o+++ `ohhhhhhhhyo++os:
      .o!`syhhhhhhh/.oo++o^
      /osyyyyyyo++ooo+++/
      :++++ +oo+++o\!
      ^oo++.
```

OS: Ubuntu 22.04 jammy
Kernel: x86_64 Linux 5.19.0-40-generic
Uptime: 5m
Packages: 1771
Shell: zsh 5.8.1
Resolution: 1633x888
DE: GNOME 41.7
WM: Mutter
WM Theme: Adwaita
GTK Theme: Yaru [GTK2/3]
Icon Theme: Yaru
Font: Ubuntu 11
Disk: 12G / 20G (64%)
CPU: AMD Ryzen 7 5700U with Radeon Graphics @ 2x 1.7976GHz
GPU: VMware SVGA II Adapter
RAM: 1657MiB / 3888MiB

- 开启SSH服务

- 1 `sudo apt update`
- 2 `sudo apt install openssh-server -y`
- 3 `sudo ps -e | grep ssh`
- 4 `sudo service ssh start`
- 5 `sudo service ssh status` # 查看SSH服务状态



```
> sudo service ssh status
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2023-04-20 13:25:03 CST; 1min 40s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Main PID: 4441 (sshd)
    Tasks: 1 (limit: 4573)
   Memory: 1.8M
      CPU: 21ms
   CGroup: /system.slice/ssh.service
           └─4441 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

4月 20 13:25:03 Starting OpenBSD Secure Shell server...
4月 20 13:25:03 Server listening on 0.0.0.0 port 22.
4月 20 13:25:03 Server listening on :: port 22.
4月 20 13:25:03 Started OpenBSD Secure Shell server.
```

- 安装 `net-tools`，使用 `ifconfig` 查看服务器IP地址

```

> ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.10 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::7fe:b430:802d:98b6 prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:54:68:ac txqueuelen 1000 (Ethernet)
    RX packets 70953 bytes 106673365 (106.6 MB)
    RX errors 753 dropped 790 overruns 0 frame 0
    TX packets 39889 bytes 2200368 (2.2 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 19 base 0x2000

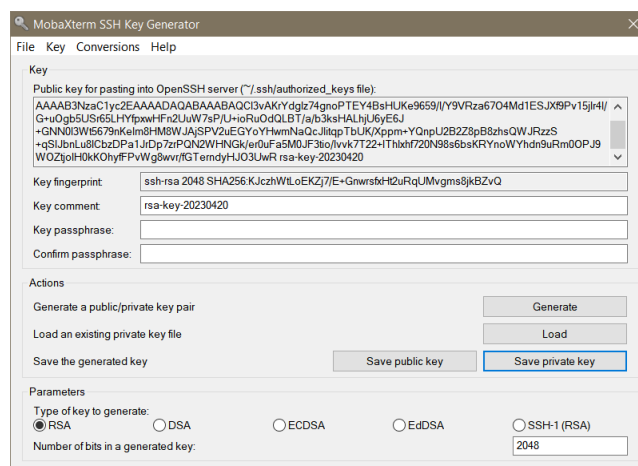
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 163 bytes 15110 (15.1 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 163 bytes 15110 (15.1 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

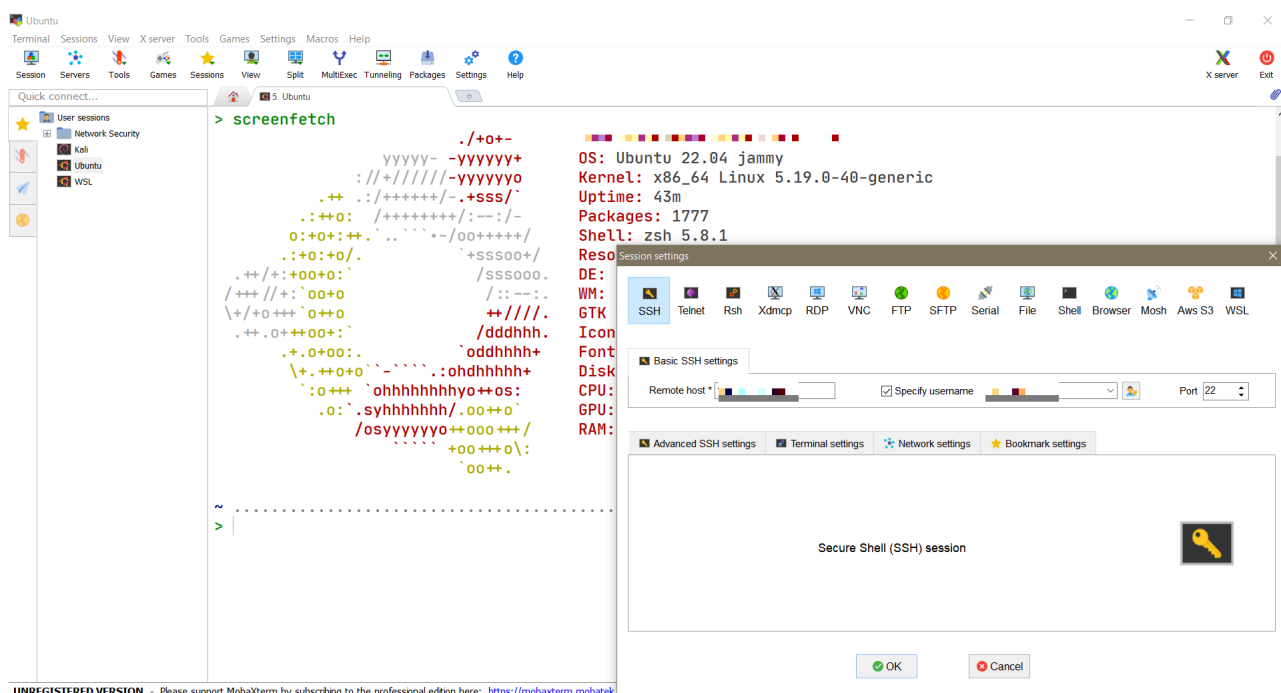
1.3 实验步骤

至此实验环境配置完成，下面开始实验“使用私钥访问SSH服务器”

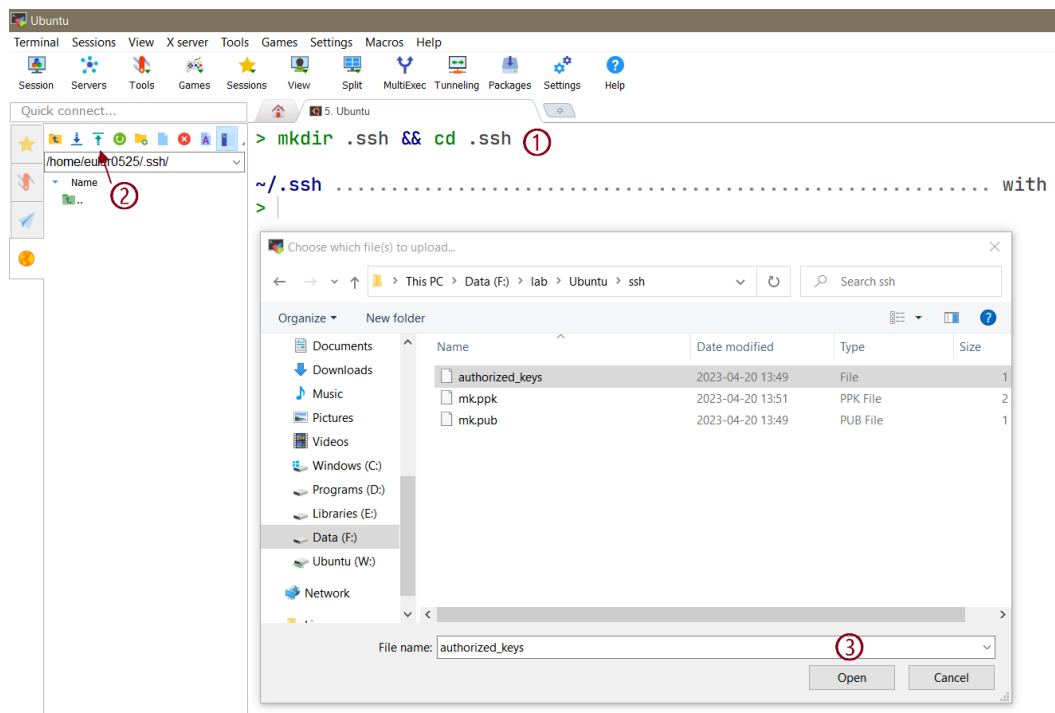
- 使用MobaXterm-Tools-MobaXterm SSH Key Generator生成基于RSA的公私钥对



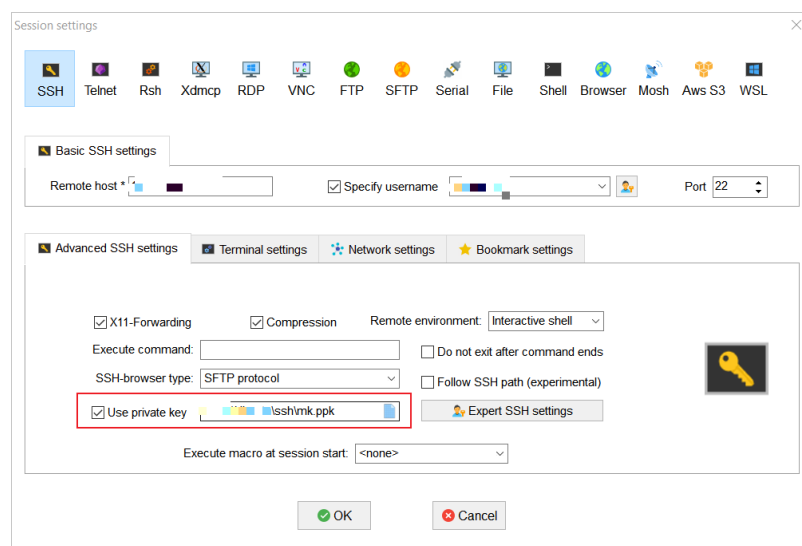
- 将公私钥分别保存为 `mk.pub` 和 `mk.ppk`，创建公钥的副本 `authorized_keys`
- 使用MobaXterm连接Ubuntu虚拟机（此时使用的是密码方式登陆）



- 在用户目录下创建 `.ssh` 目录并上传公钥副本

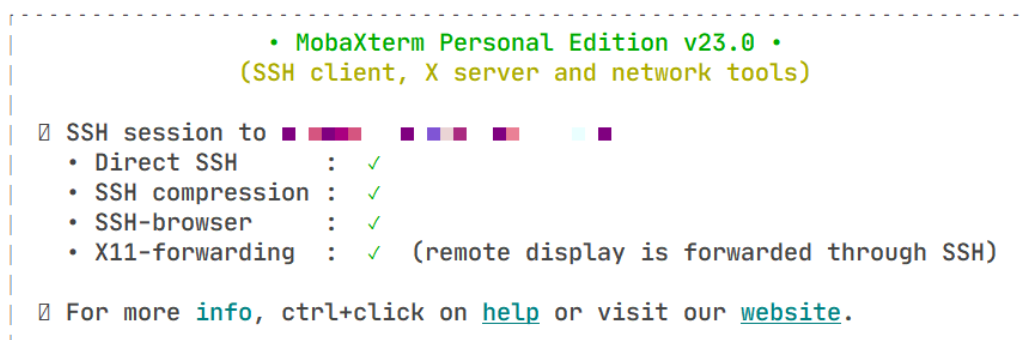


- 添加私钥



- 关闭MobaXterm与Ubuntu虚拟机的连接，重新打开，自动连接则说明使用私钥连接SSH服务器成功，并且不能使用密码进行SSH连接。

Authenticating with public key "rsa-key-20230420"



Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-40-generic x86_64)

2 实验二：为网站添加HTTPS

2.1 实验原理

HTTPS是由SSL+HTTP构建的网络协议，可进行加密传输和身份认证。其中数字证书是HTTPS实现安全传输的基础，它由权威的CA机构颁发，HTTPS通信流程如下

1. 服务器从可信CA申请证书；
2. 客户端请求与服务器建立连接；
3. 服务器发送网站证书（包含公钥）给客户端；
4. 客户端验证服务器数字证书，验证通过则建立通信；

HTTP协议传输的数据都是明文的，且不校验通信的双方的身份，所以为了安全起见可以采用HTTPS协议进行通信，它是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议。数字证书是HTTPS实现安全传输的基础，它由权威的CA机构颁发。HTTPS通信流程大致如下：

1. 服务器可信CA机构申请证书（本实验采用自签名证书）；
2. 客户端请求服务器建立连接
3. 服务器发送网站证书（证书中包含公钥）给客户端
4. 客户端验证服务器数字证书，验证通过则协商建立通信

2.2 实验环境

- 安装 [Nginx](#)

```
1 | sudo apt install nginx -y      # 安装nginx 目录/etc/nginx
2 | sudo systemctl start nginx     # 开启nginx服务
3 | sudo systemctl status nginx    # 查看nginx状态
```

```
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2023-04-28 23:02:41 CST; 9min ago
     Docs: man:nginx(8)
    Main PID: 1043 (nginx)
      Tasks: 3 (limit: 4573)
    Memory: 7.0M
       CPU: 82ms
    CGroup: /system.slice/nginx.service
            └─1043 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
              └─2072 "nginx: worker process"
                └─2073 "nginx: worker process"

4月 28 23:02:41 Euler0525-UbuntuVM systemd[1]: Starting A high performance web server and a reverse proxy server...
4月 28 23:02:41 Euler0525-UbuntuVM systemd[1]: Started A high performance web server and a reverse proxy server.
```

- 开放所需端口

```
1 | sudo ufw enable # 打开防火墙
2 | sudo ufw allow 80/tcp
3 | sudo ufw allow 443/tcp
4 | sudo ufw status
```

```
> sudo ufw status
Status: active

To Action From
--
80/tcp ALLOW Anywhere
443/tcp ALLOW Anywhere
80/tcp (v6) ALLOW Anywhere (v6)
443/tcp (v6) ALLOW Anywhere (v6)
```

- 检测nginx服务，在浏览器地址栏输入IP地址，出现下图界面则配置成功

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

2.3 实验步骤

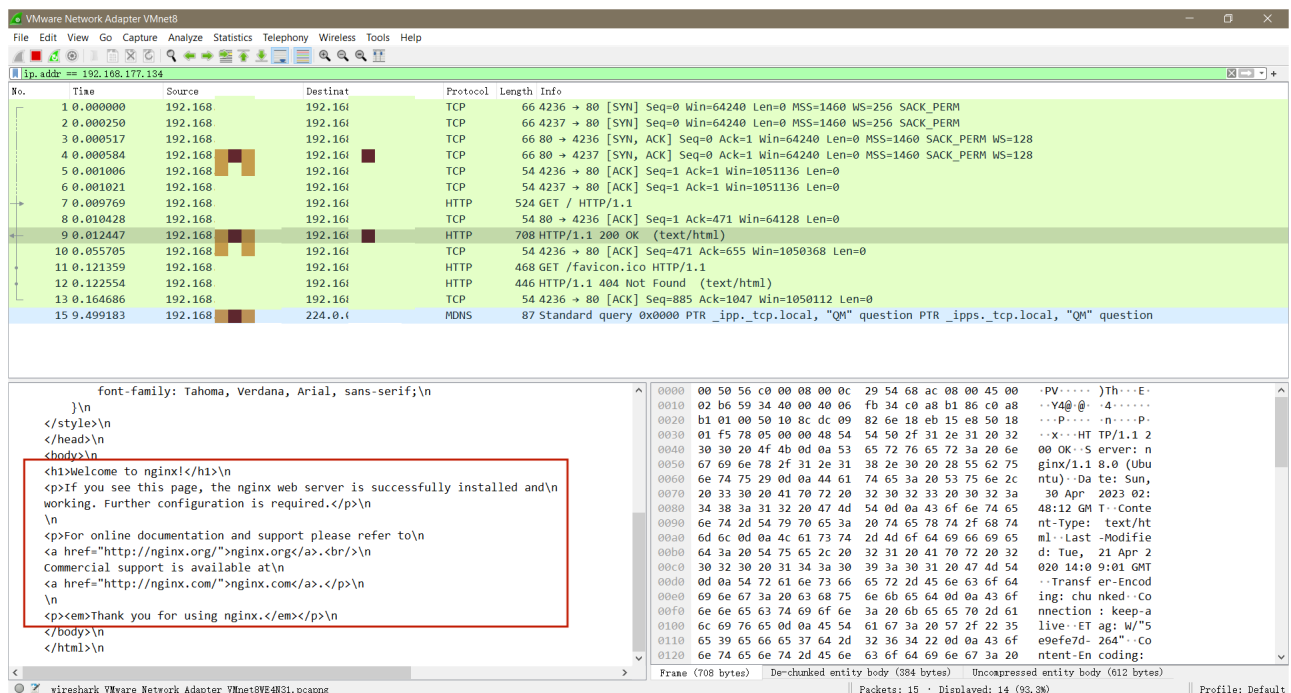
修改Nginx配置文件 `/etc/nginx/sites-available/default`，将 `server_name` 改为自己的IP地址（已做备份）

```
1 server {
2     listen 80;
3     server_name 192.168.*.*;
4 }
5
```

- 重启Nginx服务

```
1 | sudo systemctl restart nginx
```

- 未配置SSL证书前，使用网络分析器（Wireshark）对HTTP协议会话进行解析，抓取数据包结果如下图



可以看到，再使用HTTP协议时，数据都是明文传输的。

- 使用OpenSSL工具生成证书，将生成的证书存储在 `/etc/nginx/ssl/` 目录下

```
1 | sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout nginx-selfsigned.key -  
out nginx-selfsigned.crt  
2 | # req -x509: 使用 X.509 证书签名请求  
3 | # -nodes: 跳过密码保护  
4 | # -days 365: 有效时间  
5 | # -newkey rsa:2048: 基于RSA生成  
6 | # 其中Common Name需要填服务器的相关的域名或IP地址
```

- 修改Nginx配置文件 `/etc/nginx/sites-available/default` (已做备份)

```
1 | server {  
2 |     listen 80 default_server;  
3 |     listen [::]:80 default_server;  
4 |  
5 |     server_name _;  
6 |     rewrite ^ https://$http_host$request_uri? permanent;  
7 |  
8 |     location / {  
9 |         try_files $uri $uri/ =404;  
10 |    }  
11 |  
12 | }  
13 |  
14 | server {  
15 |     listen 443 ssl;  
16 |     listen [::]:443 ssl;  
17 |  
18 |     ssl_certificate /etc/nginx/ssl/nginx-selfsigned.crt;  
19 |     ssl_certificate_key /etc/nginx/ssl/nginx-selfsigned.key;  
20 |  
21 |     root /var/www/html;  
22 |     index index.html index.htm index.nginx-debian.html;  
23 |  
24 |     server_name _;  
25 |  
26 |     location / {  
27 |         try_files $uri $uri/ =404;  
28 |     }  
29 | }  
30 |
```

- 重启Nginx服务

```
1 | sudo systemctl restart nginx
```

- 浏览器导入自签名证书
- 使用网络分析器 (Wireshark) 对HTTPS协议会话进行解析

抓包结果如下, 可以发现数据被加密, :

Capturing from VMware Network Adapter VMnet8

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr == 192.168.177.134

No.	Time	Source	Destination	Protocol	Length	Info
4	0.001065	192.168.177.1	192.168.177.134	TLSv1.3	571	Client Hello
5	0.001285	192.168.177.134	192.168.177.1	TCP	54	443 → 2689 [ACK] Seq=1 Ack=518 Win=64128 Len=0
6	0.002734	192.168.177.134	192.168.177.1	TLSv1.3	1514	Server Hello, Change Cipher Spec, Application Data, Application Data
7	0.002767	192.168.177.134	192.168.177.1	TLSv1.3	182	Application Data, Application Data
8	0.003244	192.168.177.1	192.168.177.134	TCP	54	2689 → 443 [ACK] Seq=518 Ack=1589 Win=1051136 Len=0
9	0.003601	192.168.177.1	192.168.177.134	TLSv1.3	84	Change Cipher Spec, Application Data
10	0.003657	192.168.177.1	192.168.177.134	TCP	54	2689 → 443 [FIN, ACK] Seq=548 Ack=1589 Win=1051136 Len=0
11	0.003869	192.168.177.1	192.168.177.134	TLSv1.2	802	Application Data
12	0.004338	192.168.177.134	192.168.177.1	TCP	54	443 → 2689 [FIN, ACK] Seq=1589 Ack=549 Win=64128 Len=0
13	0.004716	192.168.177.134	192.168.177.1	TLSv1.2	728	Application Data
14	0.005218	192.168.177.1	192.168.177.134	TCP	54	2689 → 443 [ACK] Seq=549 Ack=1590 Win=1051136 Len=0
15	0.047533	192.168.177.1	192.168.177.134	TCP	54	2386 → 443 [ACK] Seq=749 Ack=675 Win=4100 Len=0
16	45.014938	192.168.177.1	192.168.177.134	TCP	55	[TCP Keep-Alive] 2386 → 443 [ACK] Seq=748 Ack=675 Win=4100 Len=1
17	45.016077	192.168.177.134	192.168.177.1	TCP	66	[TCP Keep-Alive ACK] 443 → 2386 [ACK] Seq=675 Ack=749 Win=501 Len=0 SLE=748 SRE=749
20	75.050473	192.168.177.134	192.168.177.1	TLSv1.2	78	Application Data
21	75.050843	192.168.177.134	192.168.177.1	TCP	54	443 → 2386 [FIN, ACK] Seq=699 Ack=749 Win=501 Len=0
22	75.051592	192.168.177.1	192.168.177.134	TCP	54	2386 → 2386 [ACK] Seq=749 Ack=700 Win=4100 Len=0

> Frame 20: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interface \Device\NPF...
> Ethernet II, Src: VMware_54:68:ac (00:0c:29:54:68:ac), Dst: VMware_c0:00:08 (00:50:56:c0:00:08)
> Internet Protocol Version 4, Src: 192.168.177.134, Dst: 192.168.177.1
> Transmission Control Protocol, Src Port: 443, Dst Port: 2386, Seq: 675, Ack: 749, Len: 24
▼ Transport Layer Security
 ▼ TLSv1.2 Record Layer: Application Data Protocol: Hypertext Transfer Protocol
 Content Type: Application Data (23)
 Version: TLS 1.2 (0x0303)
 Length: 19
 Encrypted Application Data: 91d690b3679b79cfd5d773c1a962ed68976dee
 [Application Data Protocol: Hypertext Transfer Protocol]

0000 00 50 56 c0 00 08 00 0c 29 54 68 ac 08 00 45 00 .PV....)Th...E.
0010 00 40 b8 c3 40 00 40 06 9e 1b c0 a8 b1 86 c0 a8 .@.@.....
0020 b1 01 01 bb 09 52 e9 fd 15 f0 d8 f0 53 ad 50 18R....S.P
0030 01 f5 9c e0 00 00 17 03 03 00 13 91 d6 90 b3 67b:h:m
0040 9b 79 cf d5 d7 73 c1 a9 62 ed 68 97 6d ac

Payload is encrypted application data (tls.app_data), 19 bytes Packets: 30 · Displayed: 22 (73.3%) Profile: Default