Homework #6 Solution

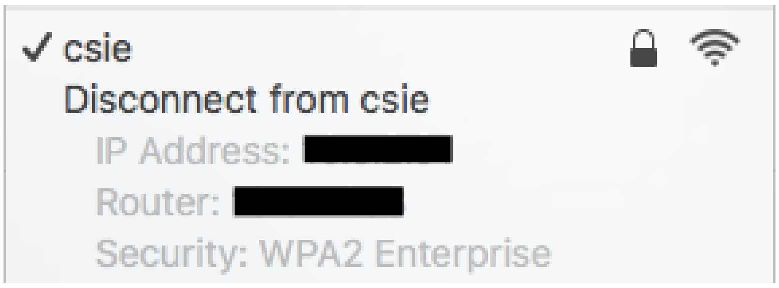
Contact TAs: `vegetable@csie.ntu.edu.tw`

Network Administration

Wi-Fi Authentication (15%)

1. WPA-Personal, also referred to as WPA-PSK (pre-shared key) mode, is designed for home and small office networks and **doesn't require an authentication server**. Each wireless network device encrypts the network traffic by deriving its 128-bit encryption key from a 256 bit shared key. The password for connecting to the Wi-Fi network is shared among all users.
   WPA-Enterprise, also referred to as WPA-802.1X mode, is designed for enterprise networks and **requires a RADIUS authentication server**. The RADIUS server is responsible for authenticating the users. Various kinds of the Extensible Authentication Protocol (EAP) are used for authentication.

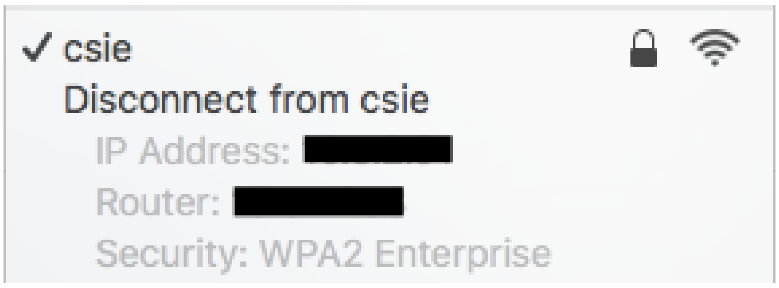
2. WPA2-Enterprise (It's fine if your answer is WPA-Enterprise)


✓ csie
Disconnect from csie
IP Address: [redacted]
Router: [redacted]
Security: WPA2 Enterprise

Wi-Fi Encryption (15%)

1.
   - WEP: RC4 steam cipher
   - WPA: RC4 steam cipher
   - WPA2: AES

2. WPA2


✓ csie
Disconnect from csie
IP Address: [redacted]
Router: [redacted]
Security: WPA2 Enterprise

WPA3 (10%)

1. Opportunistic Wireless Encryption (OWE): Many locations, e.g. restaurants, coffee shops, hotels, provide open (unencrypted) wireless networks. This makes the traffic vulnerable to sniffing

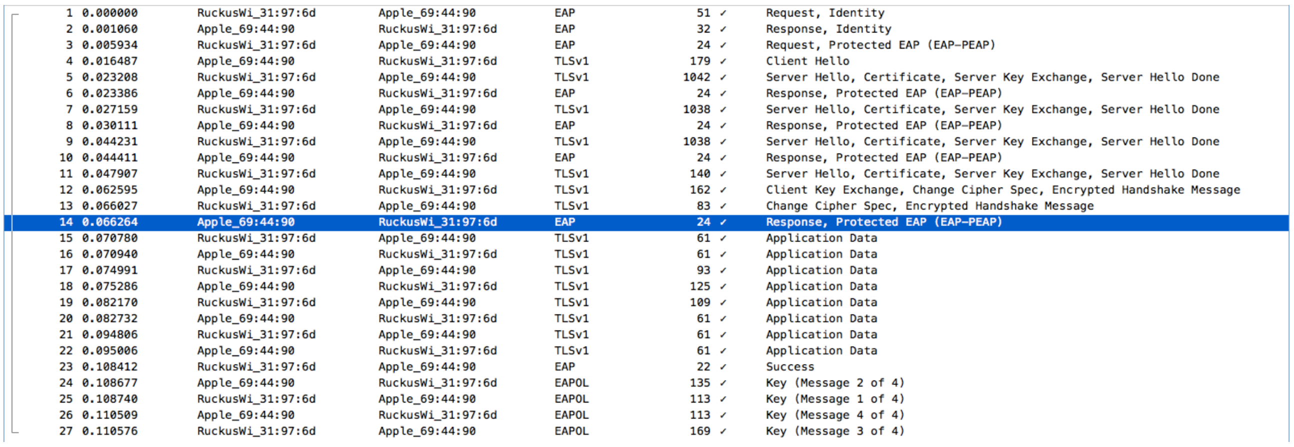
attacks. Even if the network is protected by a Pre-Shared Key (PSK), an attacker can still observe the 4-way handshake and compute the traffic encryption key. OWE, on the other hand, provides individualized data encryption. The client and the AP first perform a Diffie-Hellman key exchange and use the resulting pairwise secret, that can't be derived by sniffing the traffic, with the 4-way handshake.

2. Simultaneous Authentication of Equals (SAE): When a weak password is chosen for WPA-Personal network, it is vulnerable to offline dictionary attacks. SAE, a variant of Dragonfly Protocol, will be used to improve the security of the 4-way handshake process to protect against brute-force attacks.

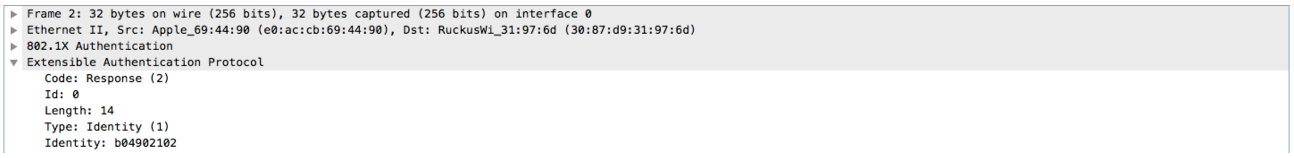
3. 192-Bit Secure Suite: This is a cryptographic strength enhancement that is aligned with the Commercial National Security Algorithm (CNSA) Suite from the Committee on National Security Systems. It's intended for government, defense, and industrial applications.

4. Device Provisioning Protocol: This protocol will provide a simple and secure way to add devices with limited or no display interface.

Seeing is Believing (10%)

3. 

```
 1 0.000000   RuckusWi_31:97:6d   Apple_69:44:90      EAP      51 ✓  Request, Identity
 2 0.001060   Apple_69:44:90      RuckusWi_31:97:6d   EAP      32 ✓  Response, Identity
 3 0.005934   RuckusWi_31:97:6d   Apple_69:44:90      EAP      24 ✓  Request, Protected EAP (EAP-PEAP)
 4 0.016487   Apple_69:44:90      RuckusWi_31:97:6d   TLSv1   179 ✓  Client Hello
 5 0.023208   RuckusWi_31:97:6d   Apple_69:44:90      TLSv1  1042 ✓  Server Hello, Certificate, Server Key Exchange, Server Hello Done
 6 0.023386   Apple_69:44:90      RuckusWi_31:97:6d   EAP      24 ✓  Response, Protected EAP (EAP-PEAP)
 7 0.027159   RuckusWi_31:97:6d   Apple_69:44:90      TLSv1  1038 ✓  Server Hello, Certificate, Server Key Exchange, Server Hello Done
 8 0.030111   Apple_69:44:90      RuckusWi_31:97:6d   EAP      24 ✓  Response, Protected EAP (EAP-PEAP)
 9 0.044231   RuckusWi_31:97:6d   Apple_69:44:90      TLSv1  1038 ✓  Server Hello, Certificate, Server Key Exchange, Server Hello Done
10 0.044411   Apple_69:44:90      RuckusWi_31:97:6d   EAP      24 ✓  Response, Protected EAP (EAP-PEAP)
11 0.047907   RuckusWi_31:97:6d   Apple_69:44:90      TLSv1   140 ✓  Server Hello, Certificate, Server Key Exchange, Server Hello Done
12 0.062595   Apple_69:44:90      RuckusWi_31:97:6d   TLSv1   162 ✓  Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
13 0.066027   RuckusWi_31:97:6d   Apple_69:44:90      TLSv1    83 ✓  Change Cipher Spec, Encrypted Handshake Message
14 0.066264   Apple_69:44:90      RuckusWi_31:97:6d   EAP      24 ✓  Response, Protected EAP (EAP-PEAP)
15 0.070780   RuckusWi_31:97:6d   Apple_69:44:90      TLSv1    61 ✓  Application Data
16 0.070940   Apple_69:44:90      RuckusWi_31:97:6d   TLSv1    61 ✓  Application Data
17 0.074991   RuckusWi_31:97:6d   Apple_69:44:90      TLSv1    93 ✓  Application Data
18 0.075286   Apple_69:44:90      RuckusWi_31:97:6d   TLSv1   125 ✓  Application Data
19 0.082170   RuckusWi_31:97:6d   Apple_69:44:90      TLSv1   109 ✓  Application Data
20 0.082732   Apple_69:44:90      RuckusWi_31:97:6d   TLSv1    61 ✓  Application Data
21 0.094806   RuckusWi_31:97:6d   Apple_69:44:90      TLSv1    61 ✓  Application Data
22 0.095006   Apple_69:44:90      RuckusWi_31:97:6d   TLSv1    61 ✓  Application Data
23 0.108412   RuckusWi_31:97:6d   Apple_69:44:90      EAP      22 ✓  Success
24 0.108677   Apple_69:44:90      RuckusWi_31:97:6d   EAPOL   135 ✓  Key (Message 2 of 4)
25 0.108740   RuckusWi_31:97:6d   Apple_69:44:90      EAPOL   113 ✓  Key (Message 1 of 4)
26 0.110509   Apple_69:44:90      RuckusWi_31:97:6d   EAPOL   113 ✓  Key (Message 4 of 4)
27 0.110576   RuckusWi_31:97:6d   Apple_69:44:90      EAPOL   169 ✓  Key (Message 3 of 4)
```

4. 

```
▶ Frame 2: 32 bytes on wire (256 bits), 32 bytes captured (256 bits) on interface 0
▶ Ethernet II, Src: Apple_69:44:90 (e0:ac:cb:69:44:90), Dst: RuckusWi_31:97:6d (30:87:d9:31:97:6d)
▶ 802.1X Authentication
▼ Extensible Authentication Protocol
    Code: Response (2)
    Id: 0
    Length: 14
    Type: Identity (1)
    Identity: b04902102
```

5.
   - Stage 1: 1-3
   - Stage 2: 4-14
   - Stage 3: 15-22
   - Stage 4: 23
   - Stage 5: 24-27

System Administration

NOTE: You don't need to mention all the points in this answer to get full credit; also, it's ok for you to write down any reasonable answer or solution, TA will read through your report to decide the last score.

Short Answer Questions (16%)

Please write down your answers as pdf format, and separate them well.

1. **What happens when we boot a Linux computer?**

   Here we only briefly describe the whole process in Linux computer using traditional BIOS (you can also find respective introduction for UEFI easily).

   - First, the computer have to load BIOS to start the operating system; it'll get the necessary information, like hardware settings, disk information, system time, etc. Also, it'll do "Power-on Self Test", POST for short. After the test, BIOS will searching for the boot loader program, which is usually installed in the first sector of boot device, that is, MBR. After loading the boot loader, BIOS just gives back the control to boot loader, and GRUB will be loaded using the information in MBR.
   - In the case of computer with several OS, GRUB will display a splash screen for user to choose. GRUB has the filesystem information, and it also contains kernel and initrd image / initramfs. It will load them to the memory. Note: stackoverflow ref.
   - After loaded the kernel and initrd image / initramfs, another hardware check will be executed, and briefly, the root file system will be mounted, and init process will start.

2. **What is the difference between traditional BIOS and UEFI?**

   - BIOS uses MBR, which has only 32-bit entries in its table; but UEFI uses GPT (GUID partition table), which has 64-bit entries.
   - BIOS can only boot from drives of 2.1 TB (or you might usually hear, 2 TB); UEFI has no such limitation.
   - BIOS must run in 16-bit processor mode, and only has 1 MB of space to execute in; UEFI can run in 32-bit or 64-bit mode, which makes the boot process faster.
   - UEFI may be stored in flash memory on motherboard, or a hard drive, even shared in network. BIOS is always put in firmware.

3. **What is the difference between NFSv2, NFSv3 and NFSv4?**

   NFSv2 v.s. NFSv3

   - The file offsets limit in NFSv2 is 32-bit; it is extended to 64-bit in NFSv3. Also, the data transfer size limit is released, which is negotiated by client and server in NFSv3.
   - NFSv3 can use TCP as transport layer protocol, but NFSv2 can't.

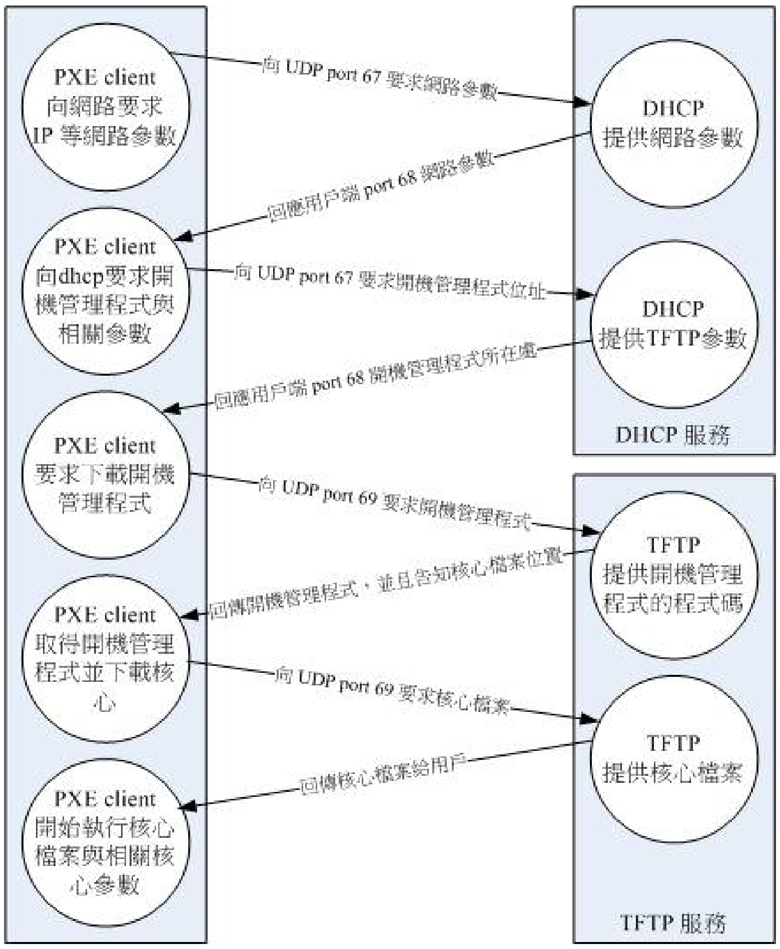
   NFSv3 v.s. NFSv4

   - NFSv3 (also, NFSv2) is stateless protocol, when NFSv4 is a stateful protocol.
   - NFSv4 uses Kerberos as authentication protocol, which is stronger than the AUTH_SYS in NFSv3 and NFSv2.

- NFSv4 supports delegation. Note: Oracle ref.

4. **In tradition, we build Preboot Execution Environment (PXE) with DHCP and TFTP services. Please explain how it works in detail.**

   Here is the illustration from vbird.


PXE client 向網路要求 IP 等網路參數
向 UDP port 67 要求網路參數
DHCP 提供網路參數
回應用戶端 port 68 網路參數
PXE client 向dhcp要求開機管理程式與相關參數
向 UDP port 67 要求開機管理程式位址
DHCP 提供TFTP參數
回應用戶端 port 68 開機管理程式所在處
DHCP 服務
PXE client 要求下載開機管理程式
向 UDP port 69 要求開機管理程式
TFTP 提供開機管理程式的程式碼
回傳開機管理程式，並且告知核心檔案位置
PXE client 取得開機管理程式並下載核心
向 UDP port 69 要求核心檔案
TFTP 提供核心檔案
回傳核心檔案給用戶
PXE client 開始執行核心檔案與相關核心參數
TFTP 服務

Briefly, the process contains several steps:

- PXE client broadcasts the packet to ask for network parameters, e.g. IP.
- After receive the packet from client, PXE server will return the necessary information to client.
- PXE client then ask for the boot image from server (pxelinux.0), and the client will execute it after receiving it; by default, the boot image searches the pxelinux.cfg directory on PXE server.
- Finally, the client will download all the files it needs (e.g. kernel and root filesystem), and then loads them.

Network Ninja (9%)

How to install an ArchLinux VM?

Note: here is just an easy installation tutorial of Arch-Linux. We'll skip the part of creating an empty VM.

- Partition the disk(s)

  Assume that there is only one disk in VM, of which the name is sda.

  ```
  $ fdisk /dev/sda
  ```

  Enter the interaction interface, create the partitions needed. (`/`, `/nfs`, swap partitions, etc.) Also, we have to format your partitions, it will be something like:

  ```
  $ mkfs.ext4 /dev/sda1

  $ mkswap /dev/sda2
  $ swapon /dev/sda2
  ```

- Installation

  After creating the partitions, we have to mount them to install something in them:

  ```
  $ mount /dev/sda1 /mnt
  ```

  And we have to install the base package first:

  ```
  $ pacstrap /mnt base base-devel
  ```

- Configuration

  First, we have to generate a fstab file to tell operating system disk partition information:

  ```
  $ genfstab -U /mnt >> /mnt/etc/fstab
  ```

  Then we can change root (`chroot`) into the new system to make life easier :)

  ```
  $ arch-chroot /mnt
  ```

  Now the root directory is `/mnt`, and we still have several things to configure.

  ```

  # Time setting
  $ ln -sf /usr/share/zoneinfo/Asia/Taipei /etc/localtime
  $ hwclock --systohc --utc

  # Locale setting
  $ locale-gen
  $ echo LANG=en_US.UTF-8 > /etc/locale.conf
  $ export LANG=en_US.UTF-8

  # Hostname setting
  $ echo MyArchLinux > /etc/hostname
  $ echo MyArchLinux > /etc/hosts

  # The most important thing: install bootloader
  $ pacman -S grub
  $ grub-install /dev/sda
  $ grub-mkconfig -o /boot/grub/grub.cfg

  ```

```
# Change the root password
$ passwd

# Update system
$ pacman -Syu
```

And then, we can exit the environment and reboot the VM. Don't forget to change the boot device.

Packet logging

nftables ver. First, install the essential packages:

```
$ pacman -S nftables
```

Then we start `nftables` service:

```
$ systemctl start nftables
```

Configure the firewall using `nft` command:

```
$ nfs add table inet filter
$ nft add chain inet filter output
$ nft add rule inet filter output oif '!=' log group 7122
```

iptables ver.
Add the rules:

```
$ ipabels -t filter -A OUTPUT '!' -o lo -j NFLOG --nflog-group 7122
```

Use ulogd to capture packets
Of course, install the package first:

```
$ pacman -S ulogd
```

Write configuration like this in `/etc/ulogd.conf`:

```
[global]
logfile="/var/log/ulogd.log"
plugin="/path/to/ulogd_inppkt_NFLOG.so"
plugin="/path/to/ulogd_raw2packet_BASE.so"
plugin="/path/to/ulogd_filter_IFINDEX.so"
plugin="/path/to/ulogd_filter_IP2STR.so"
plugin="/path/to/ulogd_filter_PRINTPKT.so"
plugin="/path/to/ulogd_output_LOGEMU.so"
stack=sal:NFLOG,sab:BASE,saif:IFINDEX,saip:IP2STR,sap:PRINTPKT,sao:LOGEMU
[sal]
group=7122
[sao]
file="/var/log/ulogd/workstation_packets.log"
sync=1
```

Then start the service:

```
$ systemctl start ulogd
```

PXE & NFS (25%)

PXE

Official guide have told you everything.

1. Mount the image

   ```
   # Get the latest image
   $ cd /mnt
   $ wget http://archlinux.cs.nctu.edu.tw/.../archlinux-2018.06.01-x86_64.iso
   $ mkdir /mnt/archiso
   $ mount -o loop,ro /mnt/archlinux-2018.06.01-x86_64.iso /mnt/archiso
   ```

2. Assume that your network is already set up, now configure the DHCP / TFTP server.

   ```
   # Install pacakge
   $ pacman -S dnsmasq
   # Write necessary configuration; change the IP to your own IP
   $ cat /etc/dnsmasq.conf
   port=0
   interface=eth0
   bind-interfaces
   dhcp-range=192.168.0.50,192.168.0.150,12h
   dhcp-boot=/arch/boot/syslinux/lpxelinux.0
   dhcp-option-force=209,boot/syslinux/archiso.cfg
   dhcp-option-force=210,/arch/
   dhcp-option-force=66,192.168.0.1
   enable-tftp
   tftp-root=/mnt/archiso
   ```

3. Setup NFS server

   ```
   # Install package
   $ pacman -S nfs-utils
   # Set the export FS
   $ cat /etc/exports
   /mnt/archiso 192.168.0.0/24(ro,no_subtree_check)
   # In case the server was already running
   $ exportfs -r -a -v
   ```

4. Edit the option `archiso_nfs_srv` in PXE boot menu:

   ```
   archiso_nfs_srv=${pxeserver}:/mnt/archiso
   ```

NFS

For the NFS part, what you need to do is just to export another filesystem (or the same one is actually ok ···), and mount it on clients as you've done in in-class lab instructed by Hsin-Mu.

One thing to notice is that ArchLinux doesn't have a default firewall, so you should be able to directly mount your FS :).