一套設備從設計、開發、部署、維護至報廢，通常歷經十幾年，由於不同的建置方式、運轉條件和環境因素，有時服務時間可能比預期還長，例如，發電廠的連網設備可能使用 20 多年都未更新，但駭客攻擊烏克蘭電廠的工業控制設備，只消幾秒鐘就能切斷電力供應。

## 入侵 IoT 的獨特之處

由於 IoT 的安全能力與傳統 IT 有很大差異，入侵 IoT 系統也會用到不同技術。IoT 生態系通常由嵌入式設備和感測器、行動 APP、雲端設施和網路通訊協定組成，常見的通訊協定包括使用 TCP/IP 的協定（如 mDNS、DNS-SD、UPnP、WS-Discovery 和 DICOM）、短距離無線電協定（如 NFC、RFID、藍牙和 BLE）、中距離無線電協定（如 Wi-Fi、Wi-Fi Direct 和 Zigbee）和長距離無線電協定（如 LoRa、LoRaWAN 和 Sigfox）。

與傳統的安全測試不同，IoT 安全測試經常需要拆解及檢查設備硬體、處理其他環境不常遇到的網路協定、分析控制設備的行動 APP，以及檢查設備與雲端 Web 服務通訊使用之*應用程式介面*（API），以下各章節將詳細說明如何執行這些任務。

來看看智慧門鎖的例子，圖 1-1 是智慧門鎖系統的通用架構，門鎖透過*低功耗藍牙*（BLE）和使用者手機上的行動 APP 通訊，行動 APP 透過 API，經由 HTTPS 與雲端的智慧門鎖伺服器通訊，這種網路設計方式，智慧門鎖依靠使用者的行動設備連接到網際網路，並從雲端上的伺服器接收訊息。

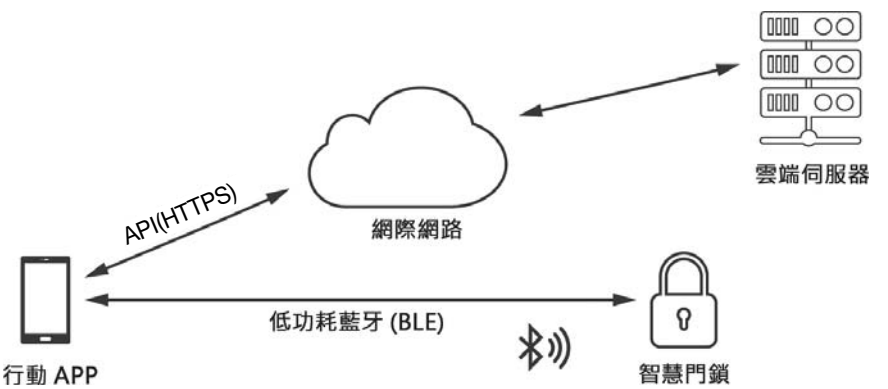


**圖 1-1：智慧門鎖系統的網路架構圖**

三個組件（智慧門鎖、行動 APP 和雲端服務）彼此信任地互動，使得 IoT 系統暴露出極大的攻擊表面，想像一下，你給訪客 Airbnb 一把數位鑰匙去開啟智慧門鎖，當你收回這把數位鑰匙後，會產生什麼後果？身為公寓和智慧門鎖設備的擁有者，你有權透過行動 APP 向雲端服務發送訊息，請求回收訪客

出技術中的弱點，而非找出有漏洞的資產或潛在的攻擊者，是目前最流行的威脅分類方案之一。STRIDE 的每個字母代表下列威脅的縮寫：

**Spoofing**（偽冒或欺騙）：行為者偽裝成系統組件的角色。

**Tampering**（竄改）：行為者危害資料或系統的完整性。

**Repudiation**（否認）：行為者否認在系統上所採取的某些行動。

**Information Disclosure**（資訊洩漏）：行為者危害到系統的資料機密性。

**Denial of Service**（阻斷服務）：行為者破壞系統組件或危害整個系統的可用性。

**Elevation of Privilege**（權限提升）：使用者或系統組件能夠將自己的存取權限提升到原本不該具有的層級。

STRIDE 的操作過程分為三個步驟：瞭解架構、分解為組件、找出每個組件的威脅。實際以輸液幫浦的威脅塑模來看看此框架運作方式。假設幫浦是使用 Wi-Fi 連接到醫院裡的控制伺服器，而醫院網路不安全且缺乏分段，也就是說醫院的訪客可以連接到 Wi-Fi 並被動監控幫浦的流量。就以這個情境來演練如何使用框架進行威脅塑模。

## 第一步：瞭解架構

藉由檢查設備的架構開始威脅塑模之旅，此系統由藥物輸液幫浦和一個控制伺服器組成，控制伺服器可以向幾十台幫浦發送命令（圖 2-1），由護理師操作伺服器，但在某些情況下，也會授權 IT 管理員存取伺服器。


圖 2-1：輸液幫浦的簡要架構圖

控制伺服器有時需要更新軟體，包括更新其藥品庫資料和病患紀錄，亦即，它有時會連線到電子健康紀錄（EHR）和更新伺服器，EHR 資料庫保有病患的健康紀錄。儘管 EHR 和更新伺服器可能已超出安全評估範圍，還是要將它們放入威脅模型裡（圖 2-2）。


圖 2-2：輸液幫浦及其控制伺服器的擴展架構圖，伺服器還連接到 EHR 和更新伺服器

## 第二步：將架構分解為組件

現在要更仔細檢查架構內容，輸液幫浦和控制伺服器由若干組件組成，要進行模型分解，以便準確地找出威脅。圖 2-3 顯示此架構的細部組件。

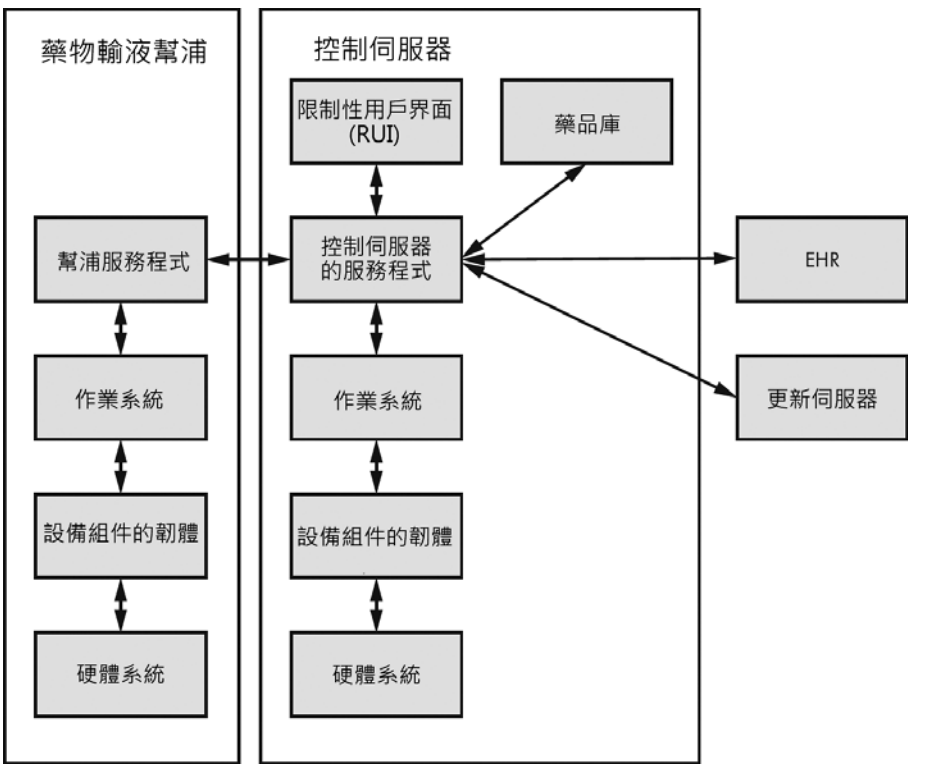


圖 2-3：進一步分解的威脅模型

幫浦系統是由硬體（真正的幫浦）、作業系統及內部運作的軟體和微控制器組成。當然也不要漏掉控制伺服器的作業系統、控制伺服器服務程式（管理控制伺服器的程式）以及管制使者與服務程式互動的限制性用戶界面（RUI）。

現在對系統有更深入瞭解，接著要確立資料在這些組件之間流動的方向，為達此目標，要找出機敏資料的位置，以及駭客可能攻擊的組件，可能還要想辦法挖出目前所不知的隱藏資料流路徑。在進一步檢查生態系後，得知資料會在所有組件之間雙向流動。圖 2-3 已使用雙向箭頭標註出此特性，請記住這個細節。

再來是於方塊圖中加入信任邊界（圖 2-4），信任邊界會將具有相同安全屬性的組件類別圈起來，有助於我們找出較容易受到威脅的資料流進入點。

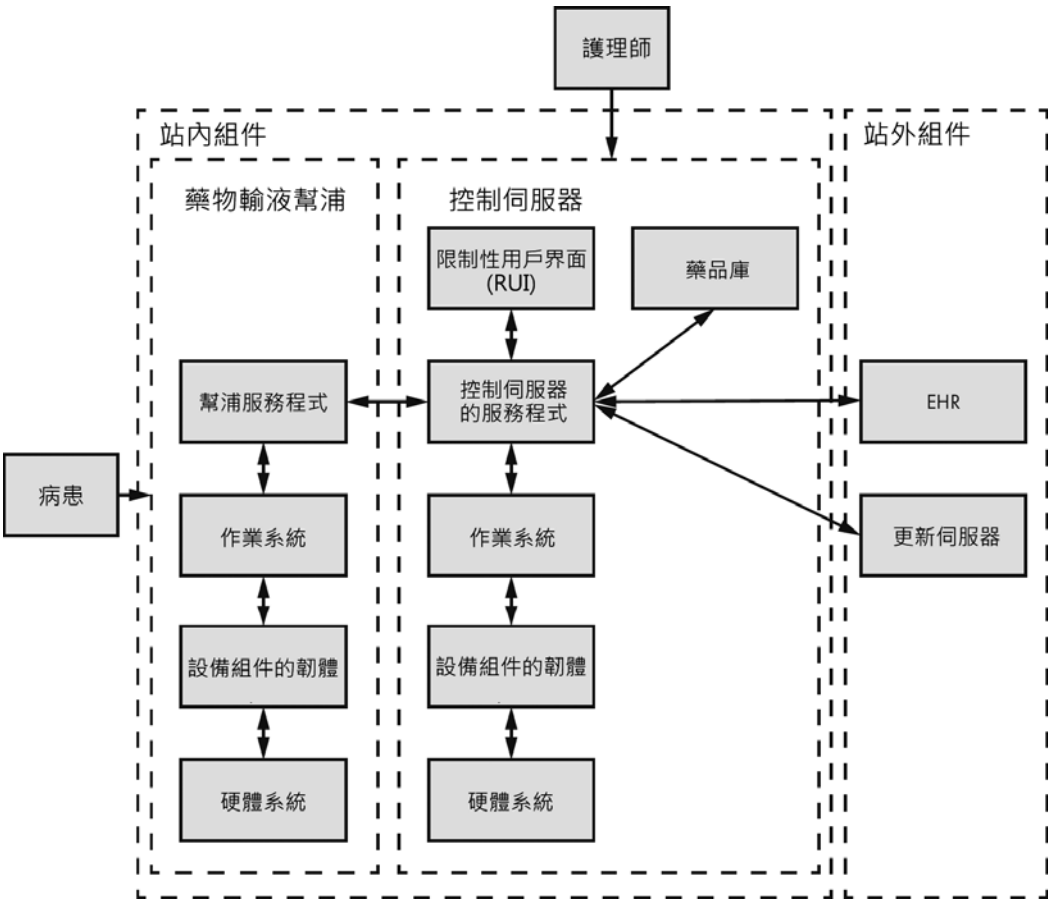


圖 2-4：具有信任邊界的架構圖

為幫浦、控制伺服器、評估範圍內的組件和範圍外的組件分別建立信任邊界，配合實際狀況，還增加兩個外部使用者：使用幫浦的病患和操作控制伺服器的護理師。

## 網路拓撲映射

拓撲映射可以描繪出網路上不同系統的連線關係，在測試整個設備和系統的生態系時，就可以執行此步驟，有時會發現某些設備和系統是經由路由器和防火牆相連接的，並不一定都位於同一 L3 網段上。L3 是指網路 OSI 模型第 3 層的網路層，負責封包轉發和繞送，當資料須透過路由器傳輸時，第 3 層就會發揮作用。描繪待測資產的網路地圖，有利於威脅塑模，可以幫助我們瞭解駭客是如何串連不同主機的漏洞而成功達成攻擊目的。圖 3-4 是網路拓撲高階示意圖。

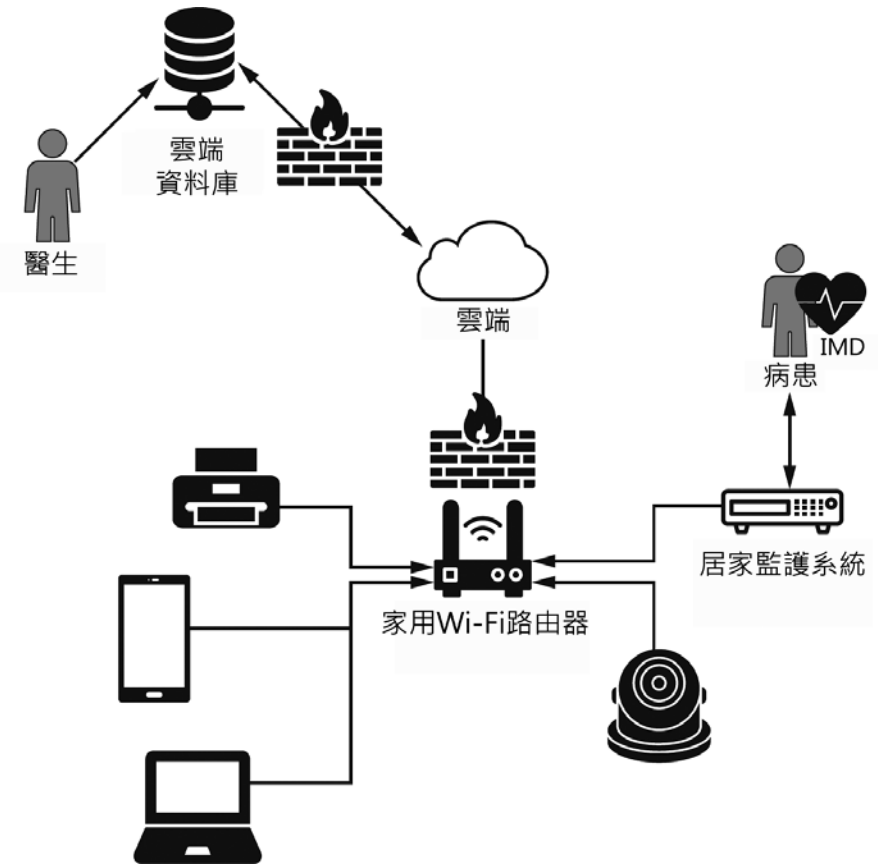


圖 3-4：家用網路的簡單拓撲圖，其中包括使用 IMD 病患的居家監護設備

這張簡要的網路地圖顯示病患的 IMD 會和居家監護設備通訊，另一方面，居家監護設備經由本地 Wi-Fi 基地台將診斷資料傳送到雲端，以供醫生定期監控病患生理是否出現異常。

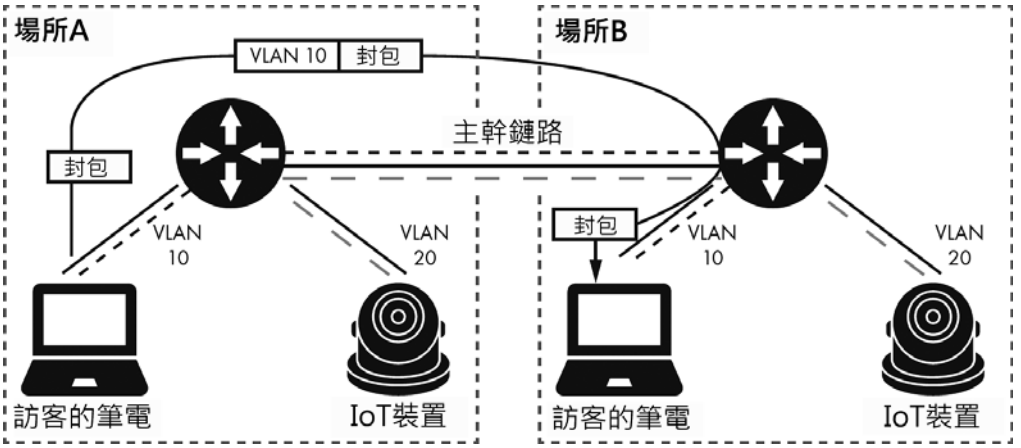


圖 4-1：以 VLAN 分隔訪客網段和 IoT 設備的常見網路架構

當 IP 攝影機之類的設備連接到存取埠時，網路認為它傳輸的封包是屬於某個 VLAN。另一方面，若某設備連接到主幹埠，便會建立 VLAN 主幹鏈路，這種連接方式允許任何 VLAN 封包通過，主幹鏈路主要用來相連不同的網路交換機和路由器。

交換器使用 VLAN 標籤來區分主幹鏈路裡的每個 VLAN 流量，要流經主幹鏈路的封包都會被貼上標籤，標籤內容與存取埠的 VLAN ID 相關聯，當封包到達目的交換器時，交換器會移除標籤並將封包傳送到正確的存取埠。有許多種標籤協定，例如交換器間鏈路（ISL）協定、區域網路仿真（LANE）協定，以及 IEEE 802.1Q 和 802.10（FDDI），網路可以使用其中一種來標記 VLAN。

## 欺騙網路交換器

許多交換器使用思科專有的動態主幹協定（DTP）動態建立 VLAN 主幹鏈路，DTP 能夠讓兩個相連的交換器建立主幹鏈路，並協商 VLAN 的標籤方式。

駭客便利用這個協定將設備偽裝成交換器，藉此實施網路交換器欺騙（switch spoofing）攻擊，讓合法交換器與他的設備建立主幹鏈路（圖 4-2），駭客便能存取來自受害交換器上的任何 VLAN 封包。

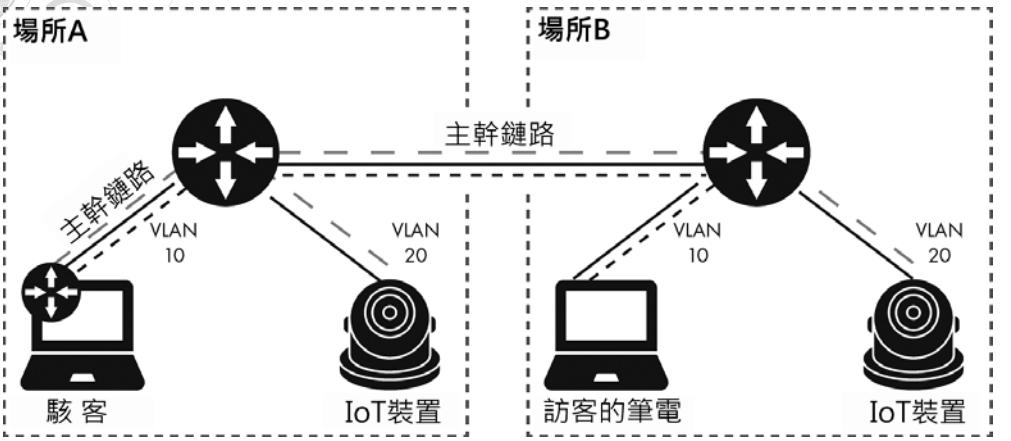


圖 4-2：網路交換器欺騙攻擊

接下來使用開源工具 Yersinia（*https://github.com/tomac/yersinia/*）發送相似於網路上真正交換器使用的 DTP 封包，嘗試執行此一攻擊，Kali 已事先安裝 Yersinia，如果讀者使用最新版 Kali，可能需要安裝 kali-linux-large 套件包才會包含 Yersinia 工具，安裝命令如下：

```
$ sudo apt install kali-linux-large
```

建議使用上面介紹的安裝方法，不要自己手動編譯工具，因為筆者發現某些工具在新版 Kali 會有一些編譯問題。

或者，試著以下列命令編譯 Yersinia：

```
# apt-get install libnet1-dev libgtk2.0-dev libpcap-dev
# tar xvfz yersinia-0.8.2.tar.gz && cd yersinia-0.8.2 && ./autogen.sh
# ./configure
# make && make install
```

要在攻擊者的設備建立主幹鏈路，請開啟 Yersinia 的圖形界面：

```
# yersinia -G
```

在畫面上選擇「DTP」頁籤，點擊「Launch Attack」功能圖示開啟 DTP 協定功擊視窗，如圖 4-3 所示。

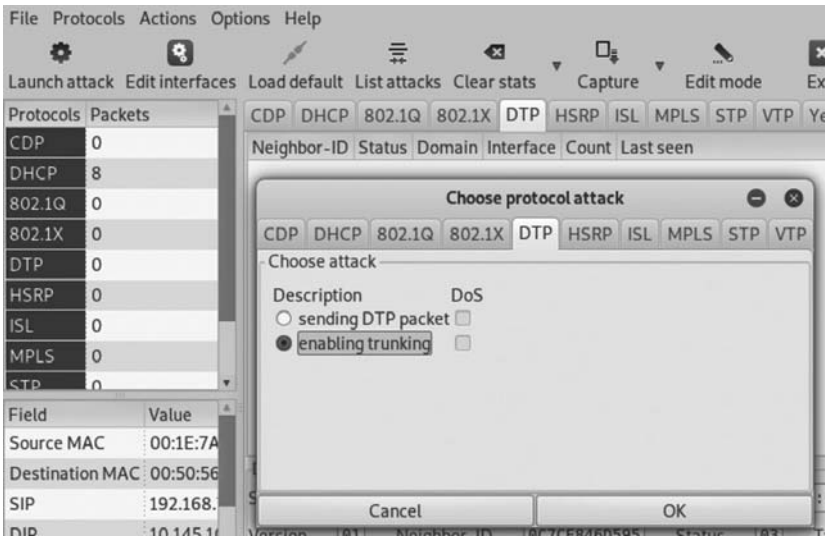


圖 4-3：Yersinia 協定攻擊的 DTP 頁籤

選擇此選項後，Yersinia 模仿支援 DTP 協定的交換器，並連接到受害交換器的連接埠，然後重複發送與受害交換器建立主幹鏈路所需的 DTP 封包。如果只想發送一個原始 DTP 封包，可以選擇第一項（即 sending DTP packet）。

一旦透過 DTP 頁籤建立主幹鏈路，應該會在 802.1Q 頁籤看到來自 VLAN 的資料，如圖 4-4 所示。

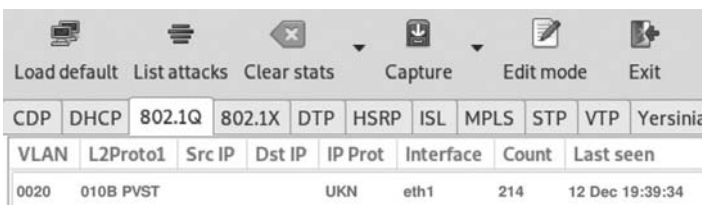


圖 4-4： Yersinia 的 802.1Q 頁籤

資料裡可看到 VLAN ID。想要存取 VLAN 封包，首先用 nmcli 命令確認網路介面：

```
# nmcli
eth1: connected to Wired connection 1
        "Realtek RTL8153"
        ethernet (r8152), 48:65:EE:16:74:F9, hw, mtu 1500
```

在這個例子中，攻擊者的電腦擁有 eth1 網路介面，請在 Linux 終端機輸入下列命令：

```
# modprobe 8021q
# vconfig add eth1 20
# ifconfig eth1.20 192.168.1.2 netmask 255.255.255.0 up
```

modprobe 命令會載入處理 VLAN 標籤的核心模組，Kali 已預裝該模組；使用 vconfig 命令建立具有所需 VLAN ID 的新介面，命令後面跟著參數 add、欲使用的網路介面名稱和 VLAN ID。Kali 已預裝 vconfig，其他版本的 Linux 可從 vlan 套件包取得。上面例子是指定 IoT 網路所使用的 VLAN 代號為 20，並以攻擊者電腦的網路介面作為通訊管道。最後以 ifconfig 命令在新建立的網路介面（eth1.20）上設定 IPv4 位址。

## 使用雙重標籤

之前提過，存取埠發送和接收未貼 VLAN 標籤的封包，因為這些封包已被視為屬於某特定 VLAN；另一方面，主幹埠發送和接收的封包則應該貼有 VLAN 標籤，這些封包可以來自任何存取埠，就算屬於不同 VLAN 的封包也可通行，當然也會有例外，具體結果取決於所使用的 VLAN 標籤協定。例如，在 IEEE 802.1Q 協定中，若封包到達主幹埠且沒有 VLAN 標籤，交換器會自動將此封包轉發至預先定義的 VLAN（稱為本徵 VLAN〔native VLAN〕），此封包的 VLAN ID 通常會被設為 1。

如果本徵 VLAN 的 ID 是屬於交換器其中某一存取埠，或者駭客執行交換器欺騙攻擊而取得該封包，就可能執行雙重標籤（double tagging）攻擊，如圖 4-5 所示。

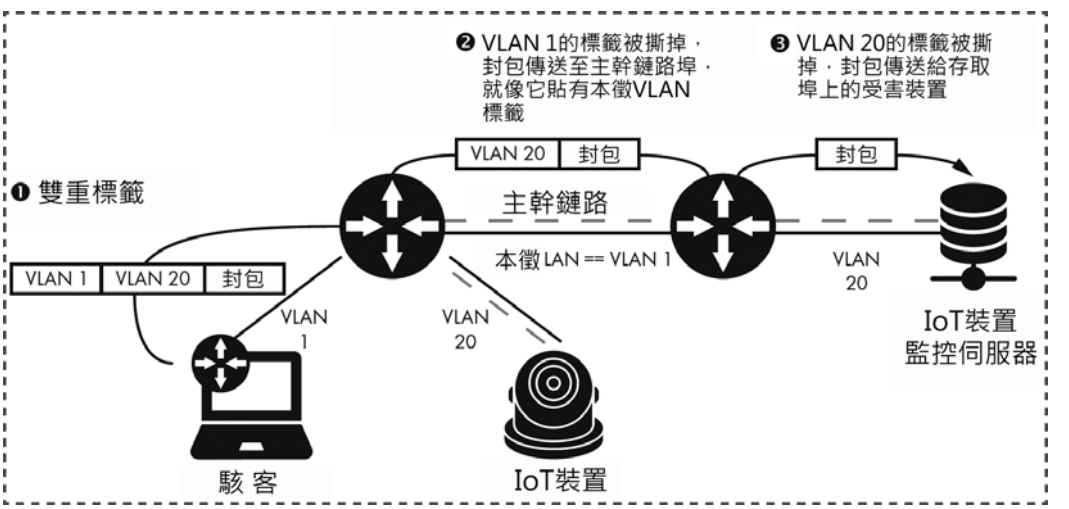


圖 4-5：雙重標籤攻擊

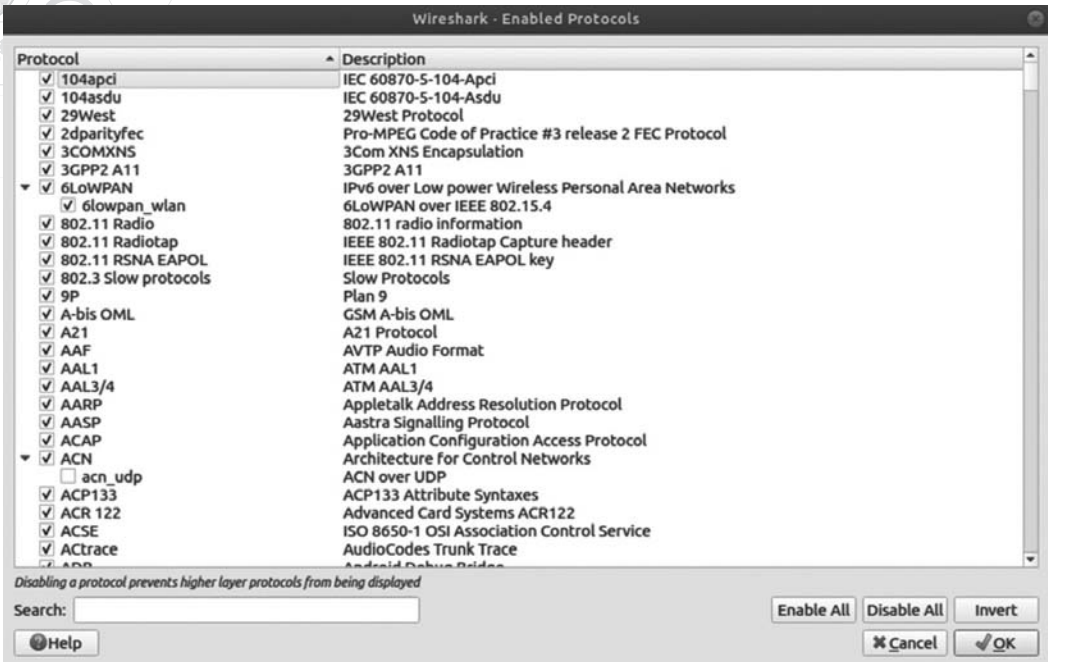


圖 5-1：檢視 Wireshark 擁有及已啟用的協定

如果協定規格是公開的，請檢查解剖器是否正確辨識所有欄位，對於複雜協定，解剖器經常會誤判，若發現任何疑點，要花點心思及精力去追查，想要有更深入的瞭解，可以查看常見漏洞和暴露（CVE）清單中關於 Wireshark 解剖器的弱點報告。

## 分析協定

在分析階段，可透過產生和重放封包來瞭解協定的工作原理，目的是為了清楚協定的整體結構，包括其傳輸層、訊息和可用的操作。

### 取得網路流量副本

依照設備類型，擷取分析所需的網路流量之方法亦不相同，如果支援代理（proxy）設定，可方便從中途攔截流量，有時需依情況，選擇主動或被動網路流量嗅探，嘗試為每個到手的案例盡量產生夠多的流量，若有不同的用戶端協助產生流量，將更能瞭解不同實作方式所隱藏的差異和怪癖。讀者若想要瞭解不同的流量擷取方法，可參閱 James Forshaw 撰寫的《Attacking Network Protocols》（No Starch Press 發行，中譯本《王牌駭客的網路攻防手法大公開》由碁峰資訊出版）。

分析階段的第一步可以考慮擷取流量，並仔細檢視發送出去和接收到的封包，有時會看到一些明顯的問題，所以，在繼續進行主動分析之前，執行此一操作會很有幫助。有一些被擷取的公開封包可在 *https://gitlab.com/wireshark/wireshark/-/wikis/SampleCaptures/* 找到，這些都是很好的研究資源。

## 用 Wireshark 分析網路流量

如果 Wireshark 具有可以解析你所產生的流量之協定解剖器，可以在 Enabled Protocols 視窗找到此協定解剖器，將它的名稱左方之查核框打勾，就可以啟用此解剖器了，如圖 5-2 所示。


圖 5-2：Wireshark 的 Enabled Protocols 視窗中被停用的協定解剖器

現在嘗試找出以下內容：

**訊息中的第一個 Byte：**有時出現在起始連線的交握（handshake）封包或訊息之第一個 Byte 是魔術位元組，是讓我們快速辨別服務的信號。

**起始連線的交握資訊：**交握是任何協定的重要功能，在此階段可瞭解協定的版本和支援的功能，包括加密方法等安全功能，重現交握過程，對開發網路掃描器很有幫助，如此便能輕易從網路找到這類設備和服務。

**協定裡使用的任何 TCP/UDP 串流和常見的資料結構：**有時會從封包中找到明文字串或常見的資料結構識別文字，例如在訊息的開頭帶有訊息長度之封包。

```
<timestamp> ServerContext.cpp:299] Disk compression is disabled
<timestamp> ServerIndex.cpp:1449] No limit on the number of stored patients
<timestamp> ServerIndex.cpp:1466] No limit on the size of the storage area
<timestamp> ServerContext.cpp:164] Reloading the jobs from the last execution of
Orthanc
<timestamp> JobsEngine.cpp:281] The jobs engine has started with 2 threads
<timestamp> main.cpp:848] DICOM server listening with AET ORTHANC on port: 4242
<timestamp> MongooseServer.cpp:1088] HTTP compression is enabled
<timestamp> MongooseServer.cpp:1002] HTTP server listening on port: 8042
(HTTPS encryption is disabled, remote access is not allowed)
<timestamp> main.cpp:667] Orthanc has started
```

如果不想安裝 Orthanc，也可從本書線上資源或 Wireshark Packet Sample Page（*https://wiki.wireshark.org/SampleCaptures*）找到別人擷取的 DICOM 封包樣本。

## 啟用 Wireshark 的 Lua 環境

在開始撰寫程式碼之前，請先安裝 Lua，並在 Wireshark 啟用它。如圖 5-4 所示，可以從 Wireshark 的「About Wireshark」視窗檢查 Lua 是否可用。[1]


圖 5-4：從「About Wireshark」視窗檢查是否已啟用 Lua

1. 譯注：若想深入瞭解如何用 Lua 開發 Wireshark 協定解剖器，可參考 Jessey Bullock 和 Jeff T. Parker 合著之《Wireshark for Security Professionals》，中譯本《資安專家談 Wireshark》由碁峰資訊出版。