逢 甲 大 學

通 訊 工 程 學 系

專 題 研 究 報 告 書

適用於任意架構的
網路電話監控與管理平台

VoIP Internet Phone
Monitoring and Management Platform

指導教授：趙啓時

學　　生：陳文遠

　　莊亞晉

中 華 民 國 一 零 六 年 十 二 月

摘要

網路的普及讓我們的生活獲得許多便利，而在目前人手一機的時代，大家經常會使用 Line、Skype 或是其他的手機 APP 來進行通話。由於 VoIP 網路電話對現代人來說十分的方便，又可以節省掉通話費用，因此將會越來越多人使用它。既然網路電話也是透過網路傳輸，那就表示我們隨時隨地都有被他人監控的可能性。倘若眞的有第三者想要從中監控我們的通話，那他們又該經過哪些機制或手段才能達成呢？又或者是因爲某些需求而需要控管每位使用者的通話內容時又該如何做到。所以，爲了因應現在網路電話當紅的趨勢，我選擇搭件一個 VoIP 網路電話控管平台來進行研究，欲達到可以有效率控管所有使用者通話內容、時間以及地點的功能，同時也能夠透過完整的監控管理過程來探討其中的資訊安全問題。

關鍵詞： VoIP、網路電話封包監控、語音辨識、資訊安全、資料庫管理。

Abstract

The popularity of Internet makes our lives more and more convenience. In this era, people usually take advantage of some tools such as Line, Skype or other applications to make a phone call. And the APP phone call is also transmitted by network. Not only VoIP Internet phone call is grows to convenient for modern people, but also can save a lot of phone bills. Consequently, more and more people will choose to use it. Since the VoIP Internet phone is transmitted by network, it means that we have the possibility of being monitored by third party. In this regard, what mechanisms does third party need to achieve to monitor what we transmit or receive. How should we do if we have some requirement to monitor each user's Internet phone call. In order to response to the current trend of Internet phone call, we would like to make the project about designing a platform to research. And we want to achieve the ability to efficiently manage the time and location of all users' call. At the same time, we can also discuss this complete monitoring process to explore the information security issue.

Keywords： VoIP, Internet phone package monitoring, voice recognition, information security, Database management

目錄

第一章　前言

1-1 研究動機

現在時代科技進步的非常迅速，相較於以前的按鍵式手機，到了現在幾乎是人人手上都會有一支智慧型手機。除了手機以外，無線網路在近幾年也迅速的發展起來，到了現在的 4G 以及 5G。網路深入世界上每一個人的生活，大多數的人也幾乎是每天都會使用到網路，所以透過網路來傳遞資料也已經是不可避免的狀況。

因爲智慧型手機與網路的普及，也讓低成本的網路電話逐漸地取代掉一般的傳統電話，而且網路電話對於現代人來說不但很方便，同時又可以節省掉大量的通話費用，倘若本身就有無線網路或是周圍有 WIFI 的話，那麼使用網路電話來聯絡彼此將會是一個很好的選擇。

網路電話已經成爲現在的趨勢，所以對於這些網路電話的監控與管理在某些必要的時刻當然也是不可或缺的，例如調查單位需要監控特定對象的通話亦或是因爲有某些需求而需要調出某時刻的通話內容時，那該要如何達成目的呢。所以我們的研究目的便是建立一個 VoIP 網路電話監控管理平台來解決上述的需求。

1-2 系統特色

在這次的研究中已經不再侷限於有限網路，在無線網路的環境下也照樣能進行網路電話的監控管理，剛好可以適應於目前無線網路當道的時代。而我們的特點又因爲使用的是 ARP Spoofing 的手法來取得通話者的音訊封包，所以就算處在有 Switch 的網路環境下也可以正常的進行監控。

若相比國內外的網路電話監控相關研究，多是達到即時監控以及資料庫管理的功能而已，但是在我們的研究中爲了讓使用者更方便於分析通話內容，所以除了網路電話監聽之外，又在系統中增加了語音辨識功能，讓使用者可以在平台上將監聽到的音訊轉成文字。此外，我們也建立了一個資料庫平台，它可以紀錄下所有的監控資料，並可以隨時隨地回來調閱監聽資訊。下**圖 1** 是網路電話監控管理系統的已達到的功能。

VoIP 網路電話監管系統功能表	
有/無線網路環境	✓
Switch 網路環境	✓
網路電話監控	✓
語音辨識功能	✓
資料庫管理	✓

圖 1：系統功能表

第二章　VoIP 網路電話監控管理系統規劃

在本章節中，我們將會針對這個 VoIP 網路電話監控管理系統的架構以及製作系統時所使用的**軟體**以及**套件工具**來做詳細的分點說明。下**圖 2** 爲簡易的系統架構流程示意圖。

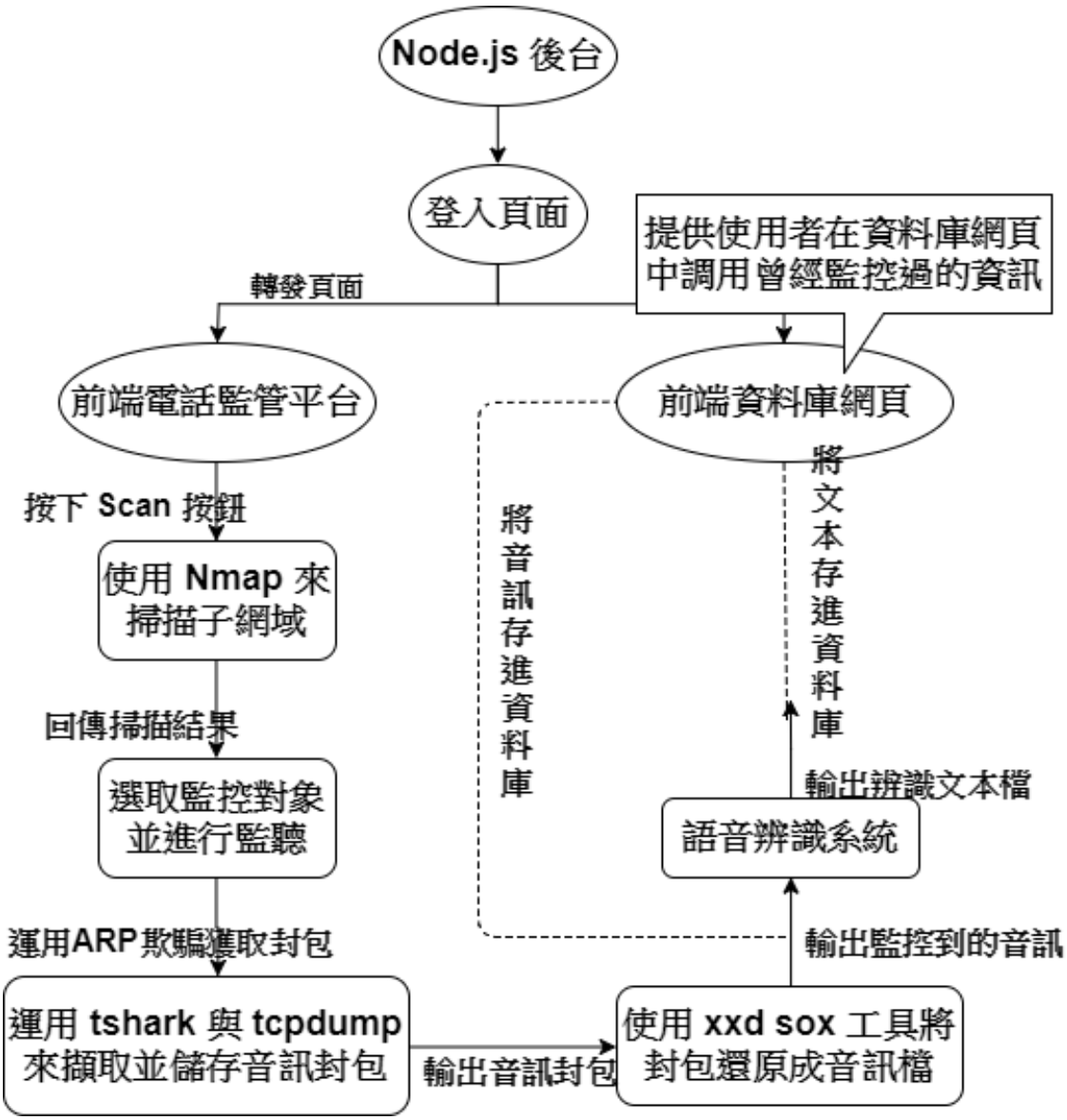

圖 2：簡易系統架構示意圖

2-1 建立網頁伺服器

在本次研究中的伺服器我們**使用 Node.js 來搭建**。Node.js (下**圖 3**) 是一個能在伺服器端運行 JavaScript 的開放原始碼，並同時擁有**跨平台的執行環境**。此外，Node.js 採用了 Google 開發的 V8 引擎，其速度比起傳統的 PHP 有很大的提升。Node.js 在**事件驅動設計**方面也非常強大，簡單而且擁有一致的介面，非常適合應用在我們的 Project。

圖 3：Node.js

2-2 掃描開啓的 Port

如果有軟體需要透過網路來進行對外通訊時，通常會有個特定開啓的 Port，所以若想要監控通訊軟體的封包就必須先掃描想監控的網路域底下有哪些 Host 開啓了哪個 Port 來進行連線通訊。例如常見的 HTTP 是 80 Port、SSH 是 22 Port，而我們要監控的網路電話軟體 Linphone 所使用的 **SIP 是預設開啓了 5060 Port**。我們將可以**利用軟體會預設開啓特定 Port 來通訊的特性來辨認是否有相關的服務正在運作**。

在掃描 IP Port 的方面我們借助了 **Nmap** 來達成目的，正好在 Node.js 中也有 Nmap 的函數庫可以直接拿存取 Nmap API。所以當使用者在前端網頁按下監聽按鈕之後便會於後端進行網域掃描，再將結果回傳到前端。

2-3 擷取並儲存封包

基本上在學校上網路概論的課程時就已經教過一個封包擷取的方法，就是透過免費且開源的網路封包分析軟體 Wireshark 來進行封包的擷取與分析。但在本研究中爲了方便做系統整合，我們選擇使用的是 Wireshark 的控制台版本，名叫 tshark。而 **tshark 的主要優點便是可以直接運用於 Shell Script 中**。

除了使用 tshark 來搜尋封包以外，我們也同時使用了 **tcpdump 來負責將搜尋到的網路通話封包儲存下來**。Tcpdump 是一個開源的封包分析與擷取軟體，不但可以分析封包流向，連封包的內容也可以監聽，最重要的是他同樣可以運行於 Shell Script，方便在研究中做系統整合。

2-4 ARP Spoofing 監聽封包

在章節 2-3 中我們提及了封包的搜尋與擷取，但是抓取封包內的前提是我們需要先想辦法取得兩個通話者之間的通訊封包，而倘若我們想要抓取到這些音訊封包的話就需要透過 **ARP 欺騙 (ARP Spoofing)** 的手段來達成。

其運作原理爲當前區域網路中的設備對其區域網路發送 ARP Request 的廣播封包來要求其 IP 與 MAC 的對應並非是設備自身的對應資訊時，會將其封包直接丟棄不予理會，但是攻擊者可以收取封包並且依照此封包來產生對應的假造封包，接著丟回區域網路中。當受害端設備收到攻擊者所僞造的 ARP Reply 封包時，將會修改自己的 ARP Table，結果將會造成自己的 ARP Table 資訊錯誤。當受害端設備要傳送封包到目的端時，由於 ARP Table 資訊已經被竄改，這**將會導致原本要傳遞的資料會被誤送到監控者手上**。

通常這種攻擊狀況比較常出現在 Default Gateway 的位址上，同樣地當監控者成功欺騙被監控者的設備後，使其受害主機端的封包要傳遞到 Default Gateway 時，封包會直接送到監控者的設備，倘若監控者又透過路由轉發將封包送回 Default Gateway 上，那麼此時的**被監控端就完全不會察覺其實自己已經遭受監控**。所以在我們的研究中就是使用了上述的手法來取得通話者的封包，下**圖 4** 爲網路電話的監控示意圖。

而在 Linux Debian 的系統的套件庫中就收錄了一個 Dsniff 套件，此套件中有個 arpspoof 軟體就可以用來幫助我們完成 ARP Spoofing 的動作。當使用 arpspoof 指令來實行 ARP Spoofing 之後可以將被監控者的封包導到監聽者(使用者) 的手上，但是在一般狀況下電腦收到這些封包後是無法進行轉發的，所以還需要**在作業系統上開啓封包轉發(forward)的功能**，我們可以直接透過使用以下的指令(**式 1**)

```
echo “1” > /proc/sys/net/ipv4/ip_forward
```

式 1：開啓 Linux 系統的封包轉發

來開啓封包轉發功能才能讓封包順利轉發出去，這樣可以避免造成被監控者的網路斷線斷線而導致我們無法順利完成監控。所以獲得通話者的封包之後，我們就可以**透過章節 2-3 的方法來擷取出我們需要的部份**即可。

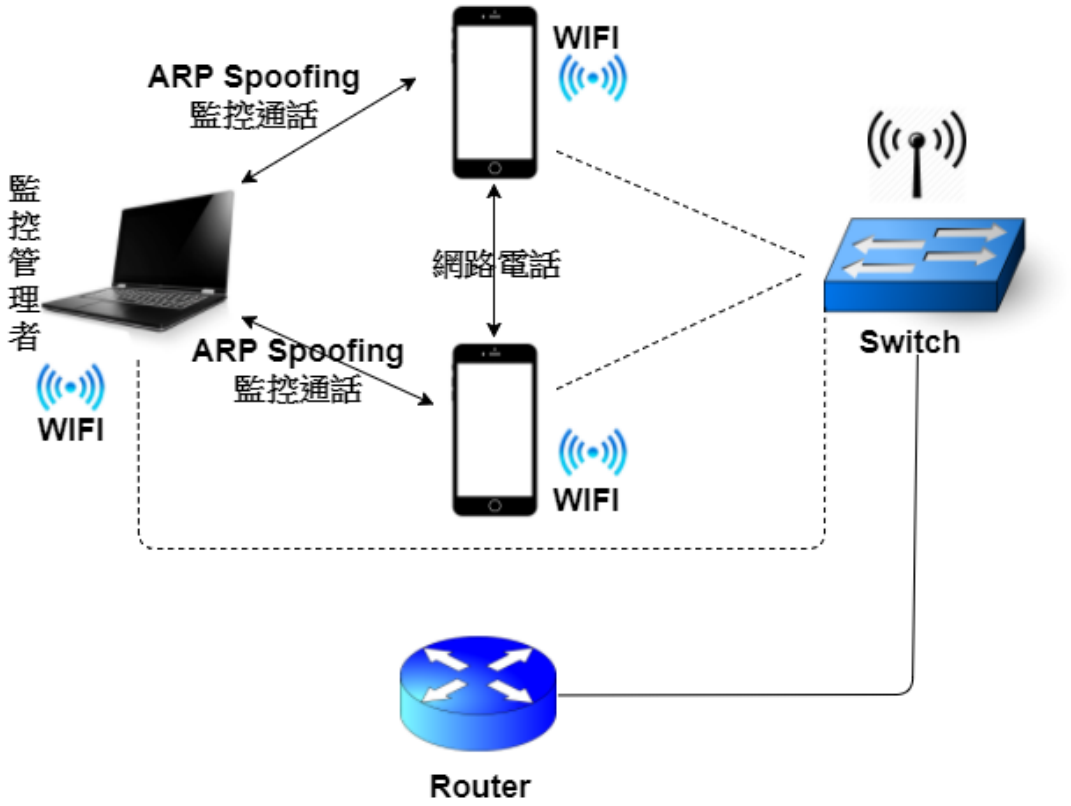

圖 4：網路電話監控監控示意圖

2-5 封包還原回聲音

使用章節 2.3 和 2.4 的手法順利取得通話者的封包後，接著便是需要將監聽到的音訊封包還原回聲音。而**聲音是透過取樣、量化與編碼之後所變成的數位訊息**，經由封包透過網路媒介來傳輸，所以當我們收到封包的時候，第一步就是要取出我們需要的音訊部分，而排除掉其他跟聲音無關的封包標頭。而此研究中爲了方便達到解析的目的，我們預先將兩隻被監控用的手機都到 Linphone APP 的設定中，統一把其音訊編碼方式都改成了 **PCMA(G.711A) 編碼**。PCMA 是一種由國際電信聯盟(ITU-T)訂定的音頻編碼方式，其**取樣頻率爲 8000 Hz**，主要應用於 VoIP 中，其優點是語音質量可以到達 CD 等級的音質，但是相對地它的耗費頻寬也最多，需要消耗到 64Kbps ，如果網路帶寬比較低的話也可以選用耗費較少頻寬的編碼方法如 G.723 或 G.729。

當我們使用了 ARP Spoofing 來獲取監控對象的音訊封包，並且使用 tshark 抓出這些封包之後，接著需要利用程式來取出聲音在封包內的編碼，而這些封包裡就是**一大串由十六進位法所表示的字串**數值，但因爲數位音訊是由二進位法所表示，所以我們還無法將其轉成音訊檔，必須先使用 Linux 的 xxd 將這些**十六進位字串反轉回原先的二進位編碼形式**並加以儲存。

我們將這些經由 xxd 轉換過後的原始二進位字串存成一個檔案，並使用 Sox 來將這一大串由二進位法所表示的數值**重新取樣回一個 wav 音訊檔**，因爲一開始我們預設兩支手機的音訊都使用 PCMA(G.711A) 編碼，所以在這裡使用 Sox 時我們就必須**設定取樣頻率參數爲每秒 8000 個字節**，並將最後結果存成一個 wav 檔，如此一來便可以完成音訊封包還原回聲音的動作，如下**圖 5**。而我們使用的 SoX 是一款跨平台而且開源的音訊轉檔編輯程式，利用他幫我們的 wav 檔加上音訊標頭以後就可以像一般正常的音訊執行檔一樣播放。

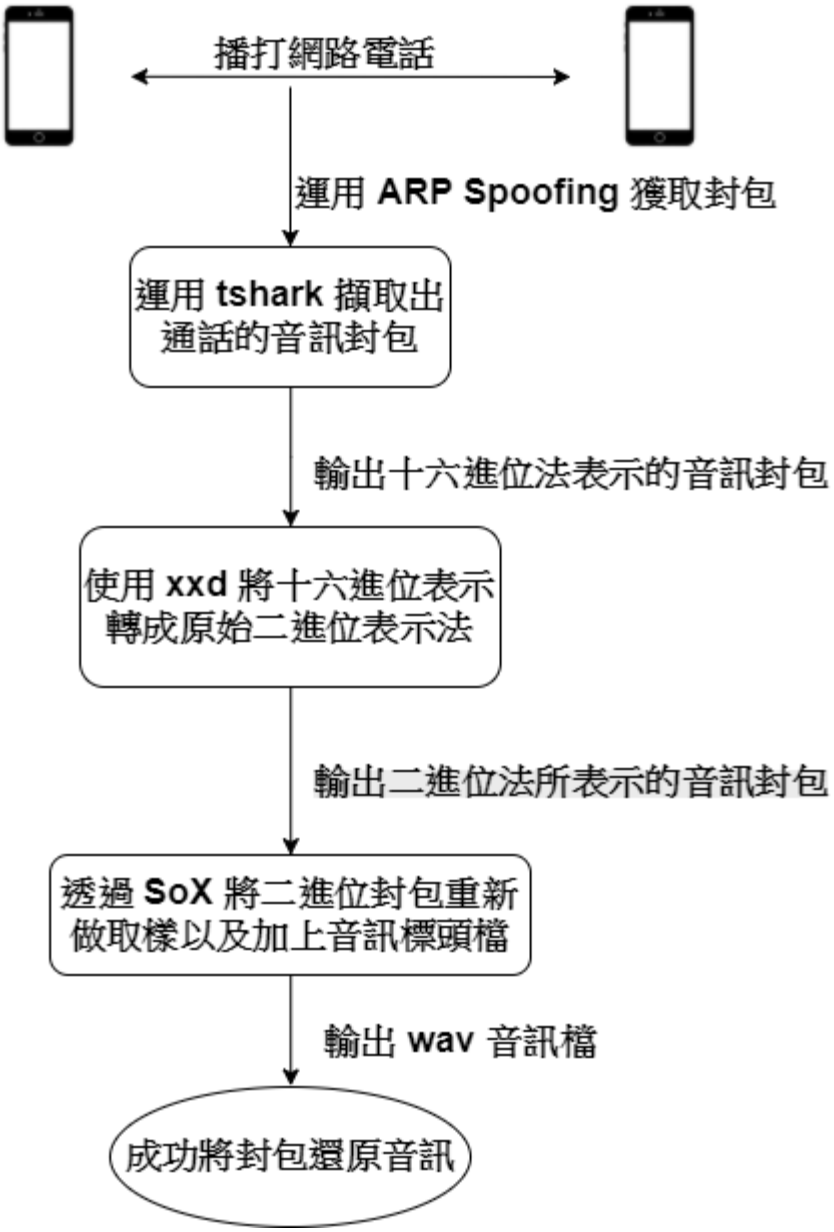

圖 5：封包還原回音訊

2-6 語音辨識系統

爲了讓使用者可以更方便分析音訊內容，所以在還原監聽到的音訊封包來順利取得音訊之後，下一步我們提供使用者進行語音辨識。在停止監聽之後，辨識音訊的按鈕將會彈起，當使用者按下此按鈕後將會開始進行語音辨識，將剛剛監聽到的通話內容辨識成文字並顯示在頁面上，如此一來可以更方便使用者來分析其通話內容。

若想要親自建立出一套完善的語音辨識系統是需要長久的時間來培養的，所以我們選擇直接使用 Google 的 Web Speech API 來達成研究目的即可，因爲 Google 的語音辨識系統早已經發展的十分完善並且**有很高的辨識度**。此 API 功能的主角是一個稱爲 **webkitSpeechRecognition** 的物件，所以我們只需要拿此物件來做程式設計即可，但是同時此物件也只支援於 Google Chrome 瀏覽器，所以使用者必須是使用 Google Chrome 瀏覽器來登入才能支援語音辨識功能。但是 Web Speech API 有個缺點，就是只能提供使用者從麥克風來進行收音並加以辨識，所以無法由自己傳入音訊封包或音訊檔來進行辨識。爲了解決此問題，我們採用了參考文獻[8]的方法來將**音頻輸出重新定向到麥克風**。

2-7 資料庫管理系統

一個網站管理服務絕對少不了處理與紀錄大量的資訊，舉凡使用者資料、服務內容和各種資訊都需要仰賴資料庫系統來儲存，所以在本研究中**我們選擇使用 MongoDB**。每當使用者登入監聽平台並進行網路電話監聽之後，後端伺服器便會把監聽到的資訊(包含監控時的年月日和時間)**自動存到資料庫**中，倘若使用者也按了語音辨識按鈕來進行音訊辨識的話，那麼辨識完成後產生的文本檔案也會自動幫使用者存進資料庫中，最後則會將資訊呈現在網頁上，使用者則**可以隨時進入資料庫頁面中來下載曾經監控過的音訊以及文本檔案**，而我們的資料庫架構大約如下**圖 6**。

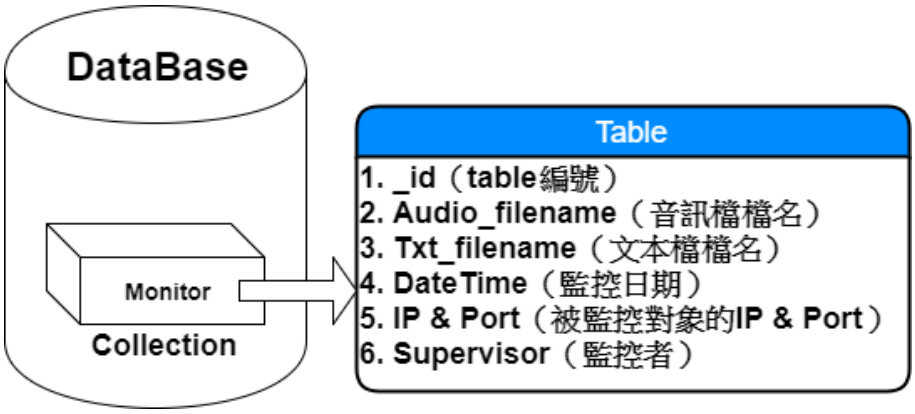

圖 6：資料庫結構

第三章 系統展示

3-1 準備需要的工具

我們需要一台電腦(監管用設備)，需求就如同**圖 7** 所示。首先在這台 Linux 系統的電腦上使用 Node.js 來架設一個網站伺服器，而因爲之後我們會使用到 Google 的語音辨識物件來做程式設計，所以客户端必須使用 Google Chrome 瀏覽器來登入伺服器才可以支援語音辨識功能。

另外我們還需要準備兩隻 Android 系統的手機(被監管之設備)，並在 Google Play 商店中搜尋到 **Linphone APP** 並下載，因爲我們在此研究中將會使用 Linphone 這款在 **SIP 協定下通訊的 VoIP 網路電話通訊軟體**來進行電話監控與管理的演示。

作業系統	Lubuntu 16.04 LTS
伺服器	Node.js v4.2.6
客戶端	Google Chrome 瀏覽器

圖 7：我們使用的監控設備

3-2 登入頁面以及監控頁面

由於電話的監控還會牽扯到資訊安全問題，所以在登入到監管頁面之前會先轉發到**登入驗證頁面**(如**圖 8**)，我們只讓特定人員可以登入進來，同時也使用了 **cookie-based Session 的技術來記錄每位登入進平台的使用者**，當使用者以正確的 Username 和 Password 成功登入之後，將可以被順利的轉發到網路電話監管頁面(如**圖 9**)。

圖 8：使用者驗證

圖 9：網路電話監控平台

3-3 監控網路電話

3-3-1 掃描網域下正在通話的裝置

等待順利轉發到監控頁面之後，使用者可以在**圖 9 區塊 1** 的地方輸入想要掃描的 IP 位址(或 IP 網域)和 Port，例如圖中的 192.168.43.1/24:5060。接著將可以按下**掃描網域按鈕**讓後端伺服器幫你進行網域掃描，由於我們用的這款 Linphone APP 所使用的 SIP 協定是預設開啓 5060 Port，因此在實際操作時我們可以直接針對 5060 Port 來進行掃描即可，但倘若使用者是**輸入一段 Port 範圍來進行掃描也是可行的**。

3-3-2 進行電話監聽

掃瞄出結果後將會回傳並顯示在**圖 9 區塊 2** 的網域掃描結果欄位裡面以提供使用者觀看當前有哪些通話對象，使用者可以**勾選出想要監控的對象並按下開始監聽按鈕來進行網路電話監聽**。當使用者結束監聽之後，將可以按下載音訊按鈕來下載剛剛所監控到的音訊內容(wav 檔)。

3-3-3 進行語音辨識

當停止監聽之後，使用者也可以選擇**按下語音辨識按鈕來進行音訊辨識**，後端將會幫使用者提取剛剛所監控到的音訊內容來進行語音辨識，將**通話內容辨識成文字檔**並且回傳到**圖 9 區塊 3** 的語音辨識結果欄位裡。在辨識完成之後，使用者將可以按下下載文本檔按鈕來下載辨識內容。此外，語音辨識方面必須先在**圖 9 區塊 4** 的地方先選定所要辨識的語言。

3-3-4 資料庫平台管理監控資訊

除了監控通話以外我們也希望可以達到管理通話的目的，所以最後還建立了一個資料庫平台(如**圖 10**)，我們將會在裡面記錄下過去監控過的每一筆資料並依照時間日期來做排序，可以提供使用者隨時回來資料庫平台調閱曾經監聽過的通話音訊檔以及語音辨識文本檔。所以在使用者**每一次監控網路電話之後，後端伺服器就會自動幫使用者把資訊都存到資料庫**並且呈現在網頁上。

圖 10：資料庫平台

第四章 結論及未來研究

4-1 結論

在本次研究中，我們所做到的事情就是建立一個適用於有/無線網路以及 Switch 網路環境或其他架構下的 VoIP 電話監控管理平台，使用者可以在上面進行子網路掃描並接著選擇想要管理的對象來進行監控，將可以取得其通話音訊檔，接著還可以把音訊拿來做語音辨識，並將辨識結果顯示在頁面上來方便使用者分析，最後再建立一個資料庫平台來負責儲存下每一筆曾經監控過的資料，其中包含了監控的日期時間以及文本檔案。

4-2 需改進之處

在章節 2.5 中有提到因爲我們爲了方便解析音訊，所以統一到 Linphone APP 的設定中將其編碼方式都改成了 PCMA(G.711A) 編碼，但是在現實情況下監控通話時不可能只僅限於 PCMA(G.711A) 編碼，所以未來會繼續改進此研究，令其也能處理其他編碼的音訊。但增加其他編碼的音訊處理方式也並非難事，因爲我們手上已經握有了 xxd 與 sox 兩個強大的工具，所以只需要去測試每個監控到的音訊的取樣頻率，並且逐一地嘗試重組資訊即可達到我們的目的。

4-3 未來研究

在語音辨識的方面，目前只能在停止監聽後才去做語音辨識，因爲我們還無法將收取到的音訊封包即時地播放出來，只能在完成整個監控過程後將其存成一個音訊檔才做播放，而未來期望能夠做到一邊監控時就能一邊辨識音訊的功能。若達到上述的功能之後，我們甚至還能讓使用者設定關鍵字，當通話者在多久時間內講到此關鍵字才繼續去做監控，否則的話則關閉等的延伸功能。

爲了加強此研究之管理方面的功能，我們在將來還會加入地圖系統，把監控對象的地點在地圖上標示出來並將其連線，同時也會接著把地點資訊也存到資料庫中來提高其資訊的完整度，以強化系統的管理功效。

因爲 IPv4 位址數量的枯竭，因此 IPv6 正在逐漸取代掉 IPv4。我們都知道 ARP 在 IPv6 中已經不復存在了，因爲在 IPv6 中使用了鄰居發現協定(NDP)來取代掉 ARP，但這並不代表此系統將會被中止，因爲我們在未來還會繼續研究 neighbor spoofing ，用以應對 IPv6 網路環境下的通話。

最後，這個研究除了探討 VoIP 網路電話的監控管理之外，我們也希望能透過其實作完整地監控管理過程來得知該如何做出應對的防禦，但是我們在此份研究中還尚未提出一套完善的防禦機制，這也是我們未來可以繼續努力的部分。

參考文獻

[1] Nmap：https://nmap.org/

[2] tshark：https://www.wireshark.org/docs/man-pages/tshark.html

[3] tcpdump:http://www.tcpdump.org/tcpdump_man.html

[4] arpspoof [man page]:https://linux.die.net/man/8/arpspoof

[5] xxd [man page]：https://www.systutorials.com/docs/linux/man/1-xxd/

[6] sox [man page]：https://linux.die.net/man/1/sox

[7] Voice Driven Web Apps:Introduction to the Web Speech API, By Glen Shires at 14 January, 2013.

[8] Redirect Audio Out to Mic In(Linux), By Noah Petherbridgeat 03 February, 2010.

[9] Rajat Aghi, Sumeet Mehta, Rahul Chauhan, Siddhant Chaudhary and Navdeep Bohra, “A comprehensive comparison of SQL and MongoDB databases”, International Journal of Scientific and Research Publications, Volume 5, Issue 2, February 2015, ISSN 2250-3153.

[10] Voice over IP：https://en.wikipedia.org/wiki/Voice_over_IP

[11] Yu-Lun Lin, “Application Raspberry Pi to Wireless ARP Spoofing Attack Defense Mechanism”, on July 2014.

[12] API to access nmap from node.js [NPM] : https://www.npmjs.com/package/libnmap

[13] 錢逢祥、蔡政崇、林政毅，不一樣的 Node.js (2014)，松崗出版