銘　傳　大　學

資　訊　工　程　學　系

專 題 研 究 總 審 文 件

本校一零七學年度　資訊工程學系

組員：張力仁　、楊承榆

所提專題研究：ICMP 通訊協定模糊測試攻擊

指 導 教 授 ：徐武孝

中 華 民 國　一 零 七　年　十二　月　十二　日

摘要

隨著網路發展越來越蓬勃，網際網路資訊變得普及且透明化，個人資訊也充斥在網路上，網路安全問題就變得更為重要。雖然網路已發展多年，仍有許多我們尚未知曉網路漏洞，而有心人士會利用這些未知的漏洞去進行網路攻擊，盜取資料或是勒索金錢。網路安全是現今一個很重要的議題，也是每個使用人所應該要有的基本概念。

模糊測試的定義是在已知目標結果的情況下，透過送出非預期的資料，並進行觀察是否會有預期外的狀況發生。本研究將以模糊測試技術來探討ICMP協定中可能存在的漏洞，主要分為五個階段: (1)研讀ICMP協定的內容並理解協定內容的傳輸方式與運作原理。 (2)藉由程式撰寫去熟悉該如何使用Socket。 (3)利用所學的ICMP與Socket的知識去進行模糊測試中所需要的ICMP欄位模組撰寫。 (4)進行ICMP協定的模糊測試。 (5)使用Wireshark觀察模糊測試後的封包狀態。本次研究測試環境將於Ubuntu上執行，並以Python作為撰寫程式的語言，我們的核心目標是利用模糊測試確認在ICMP協定中是否存在著未發現的漏洞。

關鍵字: ICMP、模糊測試、網路安全

目錄

圖目錄

表目錄

第一章 緒論

1.1 研究背景與動機

在這個資訊發達的年代，網路的普及性愈來愈高，為生活帶來的便利也急遽提升，也因此網路的安全隱私也相對地受到考驗，上過電腦網路的課程後，對網路的知識與了解有了近一步的認知，了解到一些網路上的攻擊手段，例如常見的DDOS攻擊[1]，或透過不同網路間的協定，對封包內容進行修改，導致這個封包在接收端收到時，會導致系統當機等問題。在經歷三年來的網路學程以及一些啟發性的自學，使我們對於網路安全議題有了進一步的興趣，於是我們開始想要在資安議題上做更進一步探討與實測。

從網路概論中，我們所學習到的OSI五層架構[2]中，IP(Internet Protocol)是在課堂中最常聽見且重要的協定，IP協定屬於第三層(Network)架構中的主要協定，在探討這層的時候，我們對於IP所附屬的ICMP(Internet Control Message Protocol)感到了興趣，因為我們常用的ping、traceroute指令都是屬於ICMP協定，也曾在數年前看到國外的一篇文章，內文說明了一種網路攻擊Death of Ping [3]，透過改變封包內容，導致接收端將封包重組時，長度會超過最大的處理範圍，讓接收端發生錯誤，引起了我們對ICMP的興趣。

1.2 研究目的

ICMP協定是用於錯誤偵測及回報的機制，我們將對ICMP協定的內容進行理解，並且我們將針對ICMP協定使用模糊測試(fuzz testing)，尋找在ICMP協定中可能存在的漏洞。

第二章 文獻探討

2.1 模糊測試(Fuzz testing)

模糊測試[4]（fuzz testing）是軟體測試中的一個重要的技術，核心思想是藉由無效、非預期、隨機的資料輸入到測試目標中，並進行監視程式是否發生如錯誤或崩潰的異常，以此發現可能的程式錯誤。模糊測試常用於檢測軟體或電腦系統的安全漏洞。

「模糊測試」這個名詞最早由美國威斯康辛大學(University of Wisconsin)的巴頓·米勒(Barton Miller)於1988年提出。他們隨機自動生成檔案文件以及命令列，並透過這些隨機生成的物件連續性的快速輸入直到崩潰，藉此來測試Unix程式的可靠性。他們對測試發現的錯誤進行了系統的分析。此外，為了讓其他研究人員能夠對進行類似的研究實驗，他們還公開了原始碼，測試流程以及原始結果資料。

模糊測試工具主要分為兩類，突變測試（mutation-based）以及產生測試（generation-based），突變測試主要是透過修改已經存在的資料去當作測試資料，產生測試則是通過對程式進行建模來生成新的測試資料，而本次使用的模糊測試軟體為產生測試。測試對象並不限定網路協定，文件格式、滑鼠和鍵盤事件以及API呼叫序列等都可以進行測試。

2.2 Open Source Software - Fuzz

OSS-Fuzz [5]是由Google透過GitHub所釋出的模糊測試專案。開放源碼軟體(Open Source Software)是許多程式、網站或服務的骨幹，在最近的安全事件顯示一些就算是經驗老道的開發人員也很難直接在程式碼上找到的錯誤，於是Google為了提升穩定性，便產生了這套測試專案。

OSS-Fuzz結合了現代中的各種模糊測試技術及可延展的分散式執行來讓通用的軟體架構更安全也更穩定。Google在釋出OSS-Fuzz的重點之一也是希望推動模糊測試成為Open Source開發的標準程序之一，以確保重要Open Source專案的安全性。

2.3 Peach Fuzzer

Peach [6]是由Michael Eddington等人所開發，是一個遵守MIT開源許可證的模糊測試框架，Peach需要創建一個叫Peach Pits檔，該檔案定義了被模糊測試資料中的結構、類型資訊和相互關係。最初的Peach採用Python語言編寫，初版於2004年完成開發，並公佈於ph-neutral 0x7d4網站上。第二版發佈於2007年夏天，是第一款綜合的開源Fuzzing工具。第三版發佈於2013年初，這個版本放棄了之前的架構，從Python語言更改為使用C#語言，是一個以Microsoft.NET為框架來實現的跨平台Fuzzing工具，為當前較為流行的模糊測試工具之一。

第三章 研究方法

3.1 ICMP 協定說明

3.1.1 ICMP 內容與功能

Internet Control Message Protocol (ICMP) [7]其核心目的就是用來檢測網路的連線狀態。圖3-1中為常見的ICMP類型，主要可分為錯誤回報訊息與傳輸控制訊息，以錯誤回報訊息來說，當路由器傳輸資料後找不到下一個應當傳輸的路由器，或資料超過存活時間後被丟棄時會需要一個訊息來告知這些錯誤訊息；以傳輸控制訊息而言，藉由發送ICMP封包用以查詢目前的網路狀態，ICMP協定的功能就是用來告知這些訊息。當發生錯誤時，ICMP協定會如圖3-2將所發生的錯誤類型封裝在ICMP Data中，並藉由封裝在IP中回傳給原本的發送端，這時發送端才會知道發生了什麼種類的錯誤。在電腦中有些工具是利用ICMP協定所實行的，像是Ping [8] 和 Traceroute [9]都是利用傳送封包來確認網路間的連線狀態。

Category	type	Mean
錯誤回報訊息 (error-reporting message)	3	無法到達目的地
	4	Router負載過量，來源端卻繼續發送訊息
	5	重新導向Router路徑
	11	封包逾時
	12	參數錯誤
詢問訊息 (query message)	0	回應訊息封包
	8	請求訊息封包
	13	要求時間戳記封包，以計算Router時間差異
	14	回應要求的時間戳記封包

圖3-1 ICMP 類別與常見Type

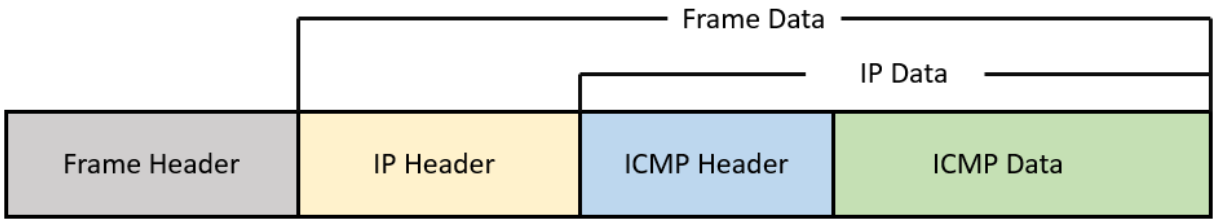


圖3-2　ICMP封裝於IP中

3.1.2 ICMP 封包格式與代表意義

ICMP協定的目的之一是用於檢測網路的連線狀況，在識別不同狀況時會以類別(Type)進行區分再以代碼(Code)定義特定訊息產生的原因。ICMP協定的類別非常的多，不同的類別有不同的代碼，也因此ICMP協定並不是所有的封包格式都長得一樣，以常見的Type 0 (echo-reply)和Type 8 (echo-request)為例，封包內格式如圖3-3:

Type: 定義ICMP訊息的種類。

Code: 對Type欄位進行詳細說明。

Checksum:檢測封包是否錯誤，欄位起始值填為0後計算。

Identifier、Sequence Number:兩個欄位都是做辨識用，由來源端程式所產生，請求封包送出後，目的端會將回應封包的Identifier和Sequence Number欄位填入來源端的Identifier和Sequence Number欄位內容，以此識別請求封包和回應封包是否為配對的。

	8bits	8bits	16bits
ICMP Header	Type	Code	Checksum
ICMP Data	Identifier		Sequence Number
	Data		

圖3-3　ICMP常見封包格式

3.1.3 ICMP 類別與功能

ICMP協定中類別可分為錯誤回報訊息與傳輸控制訊息，這邊將以ICMP中最常見的三個Type進行說明其運行方式。

錯誤回報訊息:ICMP其中一個重要的功能就是當發生錯誤的時候會進行回報的動作，我們以Type 3，無法到達目的地(Destination unreachable)作為例子，圖3-4為ICMP錯誤訊息的封包內容，當封包從來源端傳送到目的端時無法被傳送時，這個封包就會被丟棄，此時主機或路由器會利用來源端的IP位置回傳一個無法到達目的的錯誤訊息給原本的發送端。

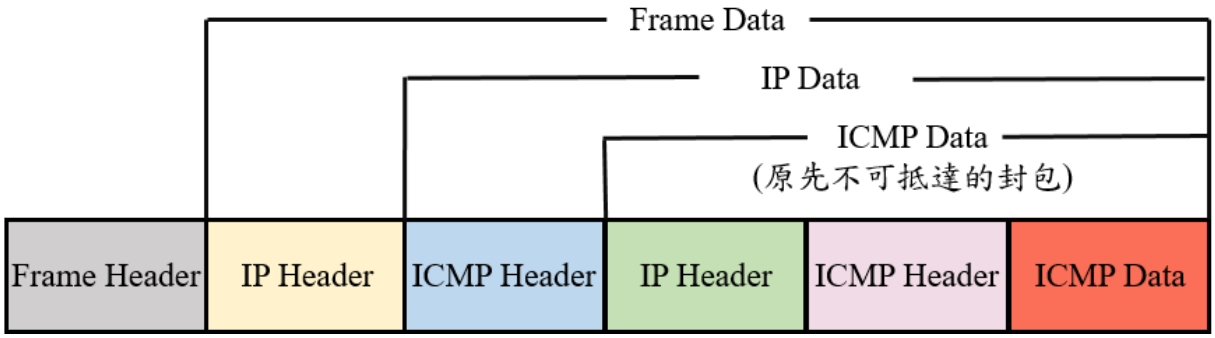


圖3-4　錯誤訊息的封裝方式

傳輸控制訊息:藉由詢問訊息，ICMP可以進行檢測某些網路問題，主要的詢問訊息為要求與回應(echo request and reply)，以及時間戳記要求與答覆(Timestamp request and reply)。以Type 8與Type 0，請求與回覆作為例子，目的端的路由器或主機送出回應要求的訊息給目的端，當目的端收到時，應該給予一個答覆訊息，由此可以確立來源端與目的端之間是否可以建立連線。

3.1.4 ICMP 常用工具

Ping：Ping指令為ICMP協定中的常用工具，用於測試IP協定是否能正確地到達指定主機(目的端)。來源端主機傳送ICMP的請求訊息(Type:8 Code0)給目的主機，圖3-5可見，若目的地主機運作中，就回應一個ICMP的回應訊息，圖3-6中為Ping在Wireshark上的擷取狀態，從來源端到目的端共進行了三次的請求與回應，與CMD上的狀態一致。Ping有計算往返時間(Round Trip Time)的功能，進行請求的時候，會將所需傳送時間填寫到Data中，當封包返回時，Ping會將抵達時間減去出發時間，並獲得往返時間。

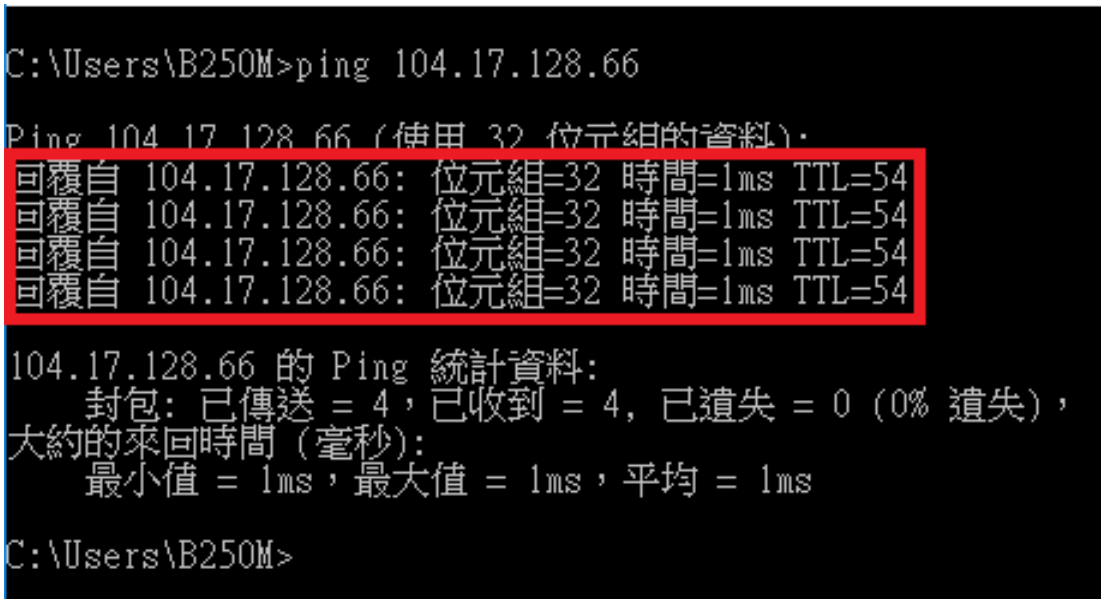


圖3-5　Ping執行結果

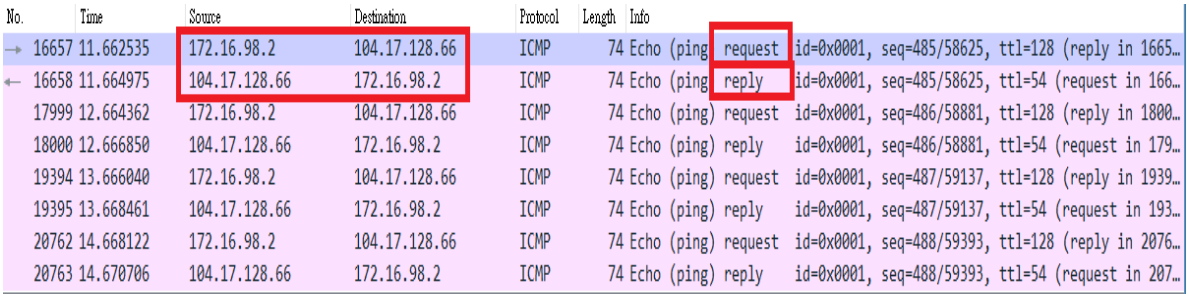
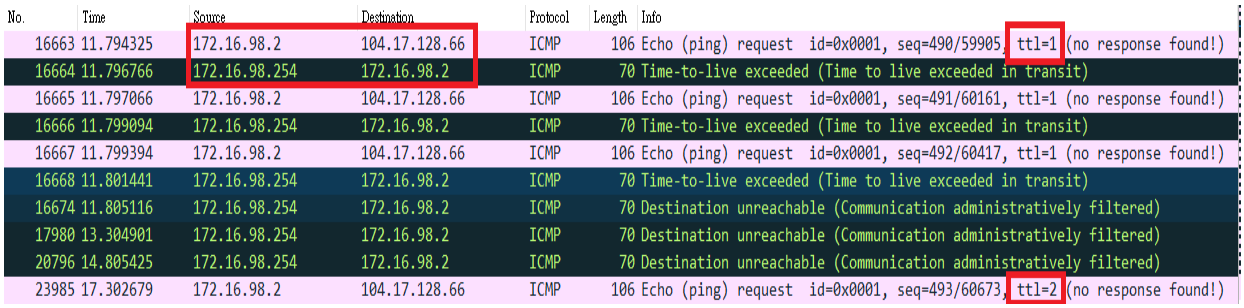


圖3-6　Ping在Wireshark上狀態

Traceroute：Traceroute是一種路由器的追蹤工具，運作原理是利用TTL(Time To Live)每經過一個路由器時都會遞減的特性。正常情況下，TTL預設值是設為來源端與目的端間路由器數量的兩倍，以此確保TTL不會有數值不足或是過多的情況，當TTL的數值為一時，下一個路由器收到後，TTL變為零，並回傳連線逾時。Traceroute利用TTL為零時會有回傳連線逾時的特性，使用UDP(User Datagram Protocol)發送一個TTL為一的封包給下一個路由器(第一個路由器為電腦的default gateway)，當路由器收到後，會回傳連線逾時，並且記錄下RTT，圖3-7可看到經過每一個路由，Traceroute會重複三次此步驟，藉此可以獲得一個更好的RTT平均值，當需要到經過更遠處的路由時，TTL會多加上一，讓前一個路由器能夠轉送此訊息，而讓下一個路由能夠丟棄封包並傳回逾時訊息，這個步驟會重複至抵達目的地。圖3-8為Traceroute在Wireshark上的狀態，可以看到TTL在一開始直接設為一，經過三次請求回應後，下一個TTL就會變成二，以此進行下一個路由器的確認。

```
C:\Users\B250M>tracert 104.17.128.66

在上限 30 個躍點上追蹤 104.17.128.66 的路由

  1     2 ms     2 ms     2 ms  172.16.98.254
  2    <1 ms    <1 ms    <1 ms  172.17.254.253
  3    <1 ms    <1 ms    <1 ms  h254.s98.ts.hinet.net [168.95.98.254]
  4     1 ms     1 ms    <1 ms  tyfo-3302.hinet.net [168.95.23.170]
  5     1 ms     7 ms     6 ms  tyfo-3015.hinet.net [220.128.8.198]
  6     3 ms     2 ms     3 ms  tpdt-3011.hinet.net [220.128.8.2]
  7     1 ms     1 ms     1 ms  r4209-s2.hinet.net [220.128.7.205]
  8     1 ms     1 ms     1 ms  203-75-228-5.HINET-IP.hinet.net [203.75.228.5]
  9     3 ms     2 ms     2 ms  LYR.66.chief.net.tw [223.26.66.66]
 10     2 ms     2 ms     2 ms  78-85-21-113-static.chief.net.tw [113.21.85.78]
 11     1 ms     1 ms     1 ms  104.17.128.66

追蹤完成。

C:\Users\B250M>
```

圖 3-7　Traceroute 執行結果



No.	Time	Source	Destination	Protocol	Length	Info
16663	11.794325	172.16.98.2	104.17.128.66	ICMP	106	Echo (ping) request id=0x0001, seq=490/59905, ttl=1 (no response found!)
16664	11.796766	172.16.98.254	172.16.98.2	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
16665	11.797066	172.16.98.2	104.17.128.66	ICMP	106	Echo (ping) request id=0x0001, seq=491/60161, ttl=1 (no response found!)
16666	11.799094	172.16.98.254	172.16.98.2	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
16667	11.799394	172.16.98.2	104.17.128.66	ICMP	106	Echo (ping) request id=0x0001, seq=492/60417, ttl=1 (no response found!)
16668	11.801441	172.16.98.254	172.16.98.2	ICMP	70	Time-to-live exceeded (Time to live exceeded in transit)
16674	11.805116	172.16.98.254	172.16.98.2	ICMP	70	Destination unreachable (Communication administratively filtered)
17980	13.304901	172.16.98.254	172.16.98.2	ICMP	70	Destination unreachable (Communication administratively filtered)
20796	14.805425	172.16.98.254	172.16.98.2	ICMP	70	Destination unreachable (Communication administratively filtered)
23985	17.302679	172.16.98.2	104.17.128.66	ICMP	106	Echo (ping) request id=0x0001, seq=493/60673, ttl=2 (no response found!)

圖3-8　Traceroute在Wireshark上狀態

3.2 模糊測試流程

本節中將說明模糊測試的流程，並藉由Wireshark進行觀察與分析。

3.2.1 ICMP 的欄位模組與測試資料

當我們要對ICMP的欄位進行模糊測試時我們必須要理解ICMP的封包欄位，每一種不同的ICMP的Type可能會有不同的欄位，我們要撰寫出這些欄位的相應模組，並將我們要測試的數值寫入測試欄位中。

ICMP中不同的欄位所佔的長度都不同，在寫入要測試的資料時，我們會依照不同的測試欄位給予不同的測試資料(pattern)，舉例來說，當我們要對Type 8中的Code欄位進行測試時，考慮到Code欄位的長度為1 byte，我們所產生出來的測試值長度必須要為1 byte。

如圖3-9可看到1 byte的測試資料從\x00到\xff總共256筆，我們在需要用到1byte的測試資料時會選擇把全部的測試資料跑完，但是當用到2 bytes以上的欄位時，要將全部資料測完所需準備的資料是非常巨量的，以4 bytes的例子來說，就會有256的4次方(4294967296筆)種資料，龐大的資料量會導致在進行測試的時候非常沒有效率，而且Wireshark在進行這種巨量的資料截取時，往往會因為截取的封包過多而導致程式崩潰，所以我們在進行2 bytes以上的模糊測試時，會藉由隨機亂數產生一定數量的資料去測試。

圖3-9　1 byte全部測試資料

3.2.2 執行與觀測

當建立好要測試的ICMP模組與測試資料後，我們會利用虛擬機在Ubuntu的環境下執行模糊測試的程式，選擇在Ubuntu的環境下執行程式有兩個原因，首先是我們在進行OS課程時因為課程需求而接觸了此系統，對此系統的操作比較熟悉，其次是在此系統的環境下相比Windows系統較為乾淨，Windows上使用Wireshark可能會因為背景系統運作而產生許的封包，在進行觀測上會比較容易受到干擾。如圖3-10所見，執行程式時我們會用python進行Fuzz檔的編譯，接著我們會打上要測試的協定以及目標IP。

圖3-10　模糊測試程式執行方式

開始執行程式後，圖3-11中可以看到程式會列出所用的IP Version、發送端IP(src_ip)、目的地IP(dst_ip)、發送端實體位置(src_mac)以及網路遮罩(netmask)，往下為本次所要進行模糊測試的檔案，圖3-12是檔案所在的資料夾，可與執行狀態進行核對，在預設的資料夾中會找到我們指定測試的檔案。

圖3-11　程式執行狀態表

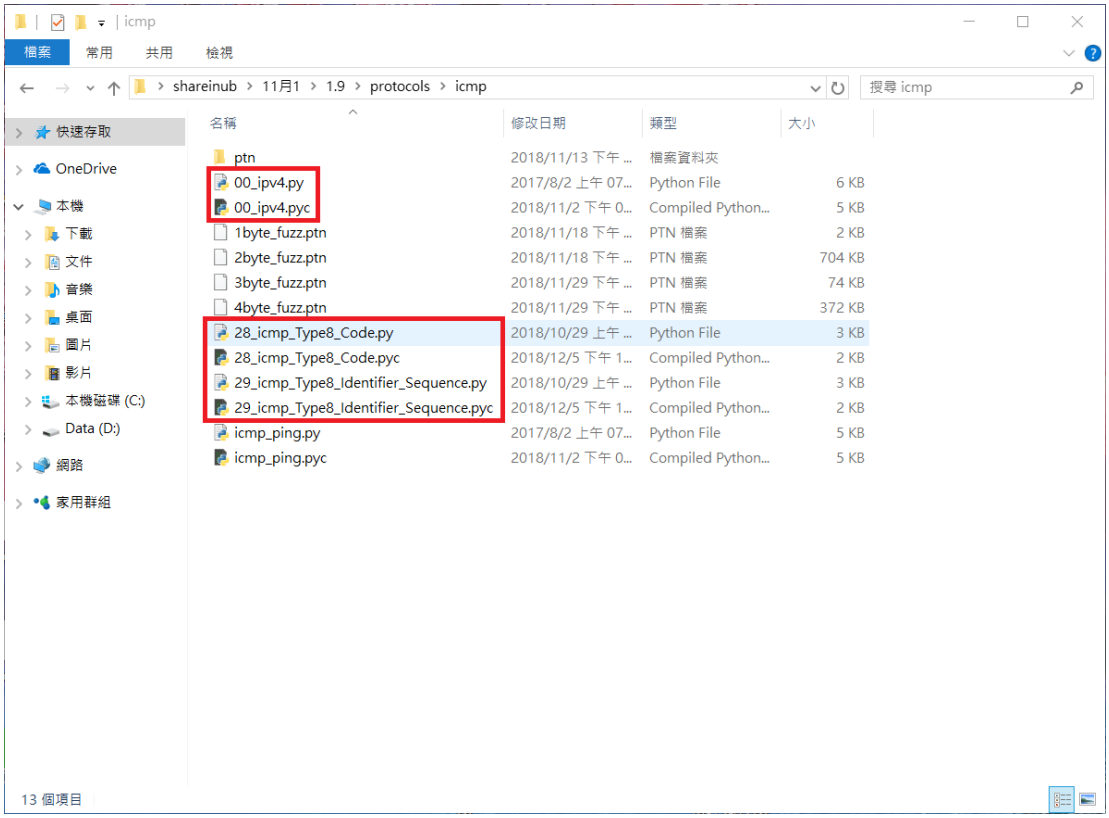


圖3-12　確認是否與指定測試資料吻合

程式開始執行後，從圖3-13中可以看到目前正在對哪一個欄位模組進行模糊測試，總共需要執行多少測試資料以及目前測了多少筆測試資料，同時，觀察已經開啟的Wireshark可看到圖3-14中的樣子，大量的ICMP封包正在被擷取，可以說明模糊測試程式正在進行ICMP封包的輸送。

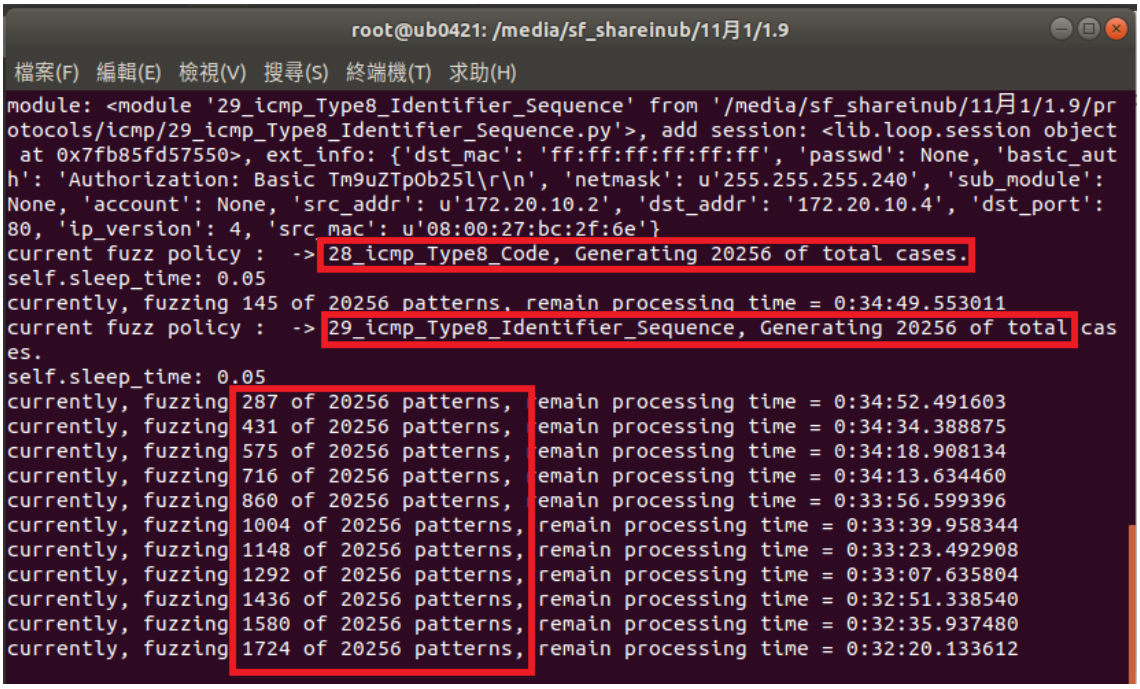


圖3-13　模糊測試發送狀態

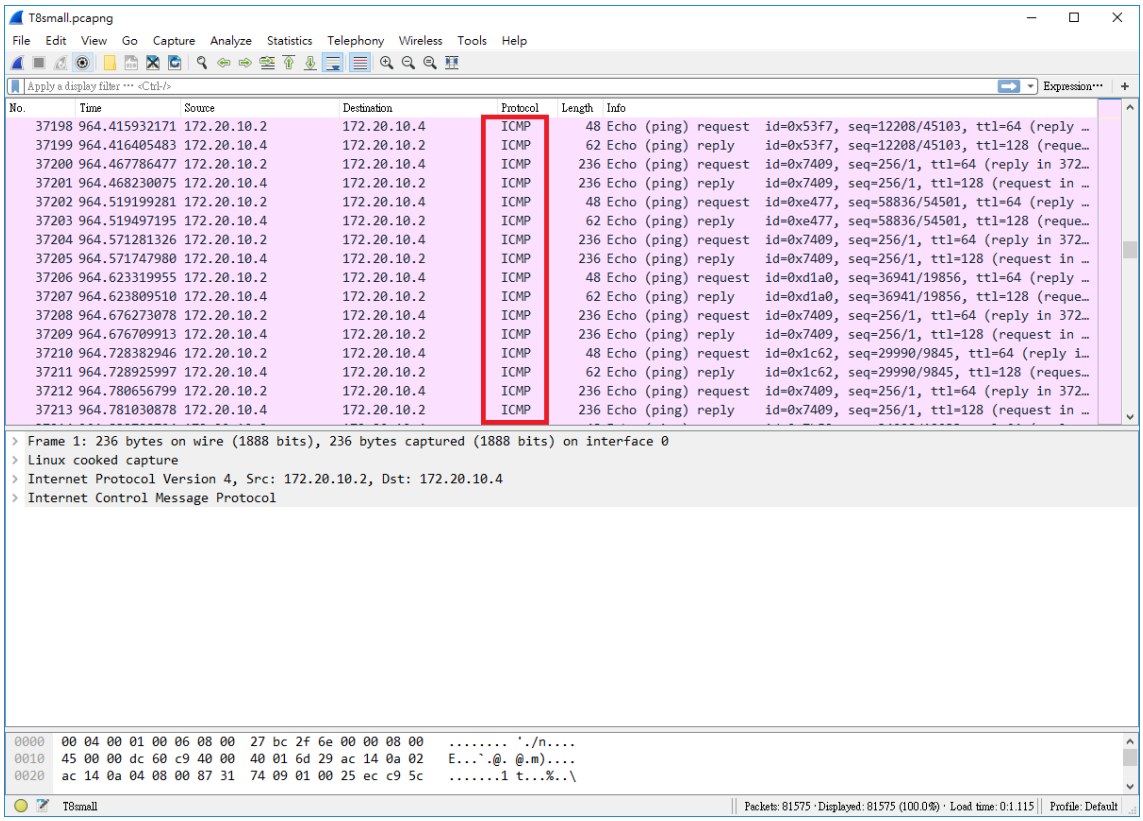


圖3-14　Wireshark抓取程式送出的ICMP封包

第四章 研究結果

4.1 檢驗內容

當模糊測試程式跑完所有的測試資料後，由於我們所輸入的測試資料是依順序進行模糊測試的，所以我們可以利用Wireshark上的封包內容進行檢驗我們所輸送的測試內容是否有達到我們原本的預期。我們以Type 8的Identifier、Sequence Number欄位為例，Identifier、Sequence Number為4bytes長度，我們將進行隨機亂數生成多筆測試資料，圖4-1為其中一組4byte的隨機亂數，頭尾的數值分別為"\x4f\xd5\x9c\x69"和"\xdd\x24\xf7\xb8"，圖4-2與圖4-3中分別為從Wireshark所觀測到的第一筆與最後一筆送出的測試資料，可以看到送出的數值與我們的測試資料頭尾是相同的，藉由這種檢驗方式我們可以確認這些模糊測試的傳輸資料是否跟我們所預期的相同。

圖4-1　4 bytes的第一筆與最後一筆測試資料

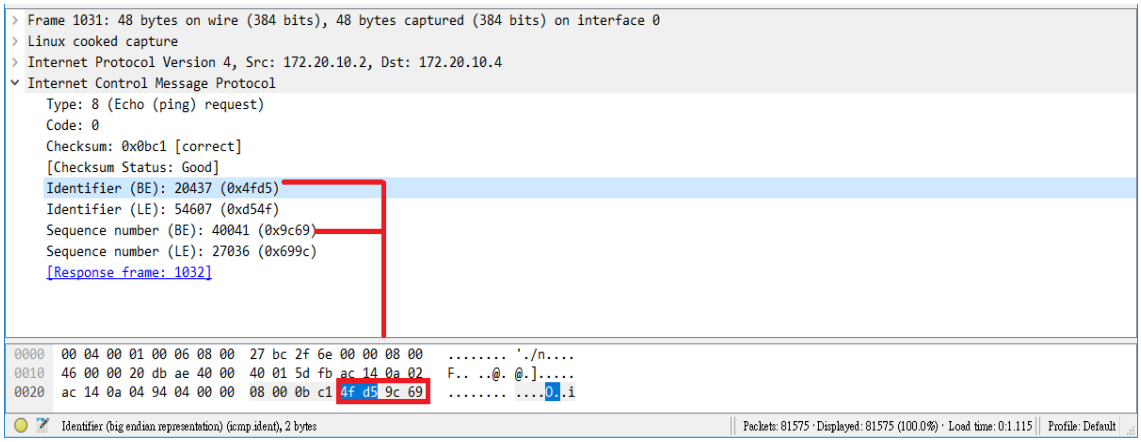


圖4-2　對照Wireshark的4 bytes第一筆資料封包是否正確

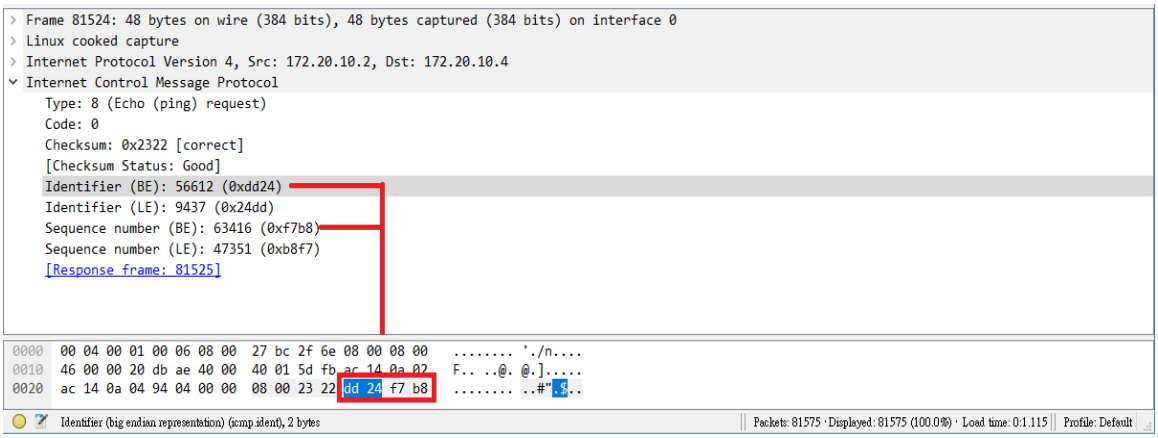


圖4-3　對照Wireshark的4 bytes最後一筆資料封包是否正確

4.2 模糊測試的運行狀態

圖4-4為對Type 8的Code欄位進行模糊測試後Wireshark的片段擷取，可以看到27號封包為我們所試圖送出的ICMP封包，由於Type 8的狀態為進行請求(Request)，因此在發送完後28號封包會進行回應(Reply)的動作，這邊為我們進行模糊測試所送出的請求回應封包，每當輸送完模糊測試的封包後，模糊測試程式會送出一個檢測封包，這些封包在ICMP協定上會判定為正確且無誤的請求回應封包，藉由這些檢測封包我們可以判定每次發送完模糊測試封包後，後續的伺服器有無異常，如果出現異常，後續的ICMP封包可能就無法得到回應，此時就可以知道是否出現漏洞。

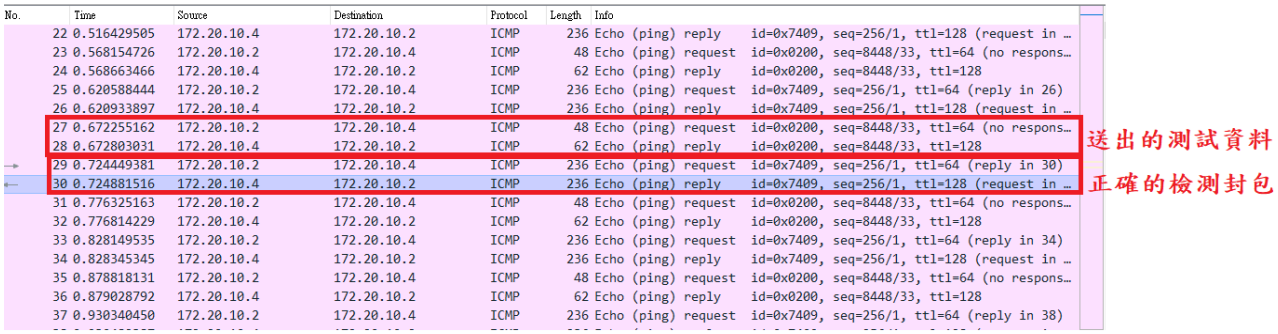


圖4-4　模糊測試的運行方式

4.3 結果與分析

我們將進行模糊測試過後結果分析，每個Type的功能與目的都不同，圖4-5為對Type欄位進行模糊測試後的Wireshark狀態，可看到每個Type欄位的目的性，但其中也有一些例外，如Type 1、Type 2及其他一些Type在協定中為被保留的，在Wireshark中的狀態顯示為Reserved，他們並沒有任何意義，而在Wireshark中Unknown ICMP(obsolete or malformed)被判定為未知的ICMP封包，有些可能在RFC文件中有紀錄或呈現保留狀態，但是Wireshark並沒有更新到最新的狀態，所以它會不知道這個Type是否存在。

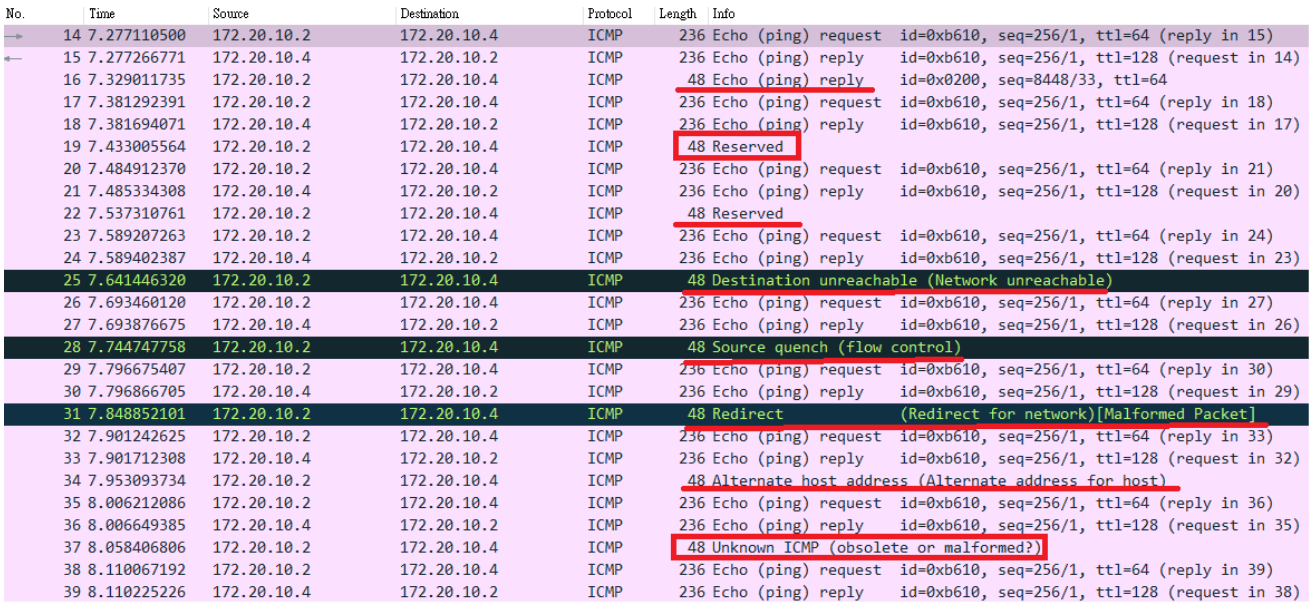


圖4-5　對Type欄位進行Fuzz

以下我們將用模糊測試完的Type來進行分析。

Type 8 (Echo Request)與Type 0 (Echo Reply)，是一種詢問訊息，Code只有0，表4-1為Type 8與Type 0的封包欄位，我們將對Code與Identifier、Sequence Number欄位進行模糊測試，Type 8將對目的端進行請求回應，已確定來源端與目的端間是否連通，若來源端與目的地是可以連通的，每次送出封包後，都會從目的地傳回一個回應封包，Code欄位已經全部測試過，並沒有出現錯誤，Identifier、Sequence Number也已經分別進行測試，並沒有出現錯誤。

Type	Code	Checksum
Identifier		Sequence Number

表4-1　Type 0、Type 8封包欄位

Type 3 (Destination Unreachable)，是一個錯誤回報訊息，如圖4-6，Code共有15種，用於表示目的地無法到達，表4-2為Type 3的封包欄位，我們將對Code與Unused欄位進行模糊測試，當Router或主機進行封包的運送後，發現封包無法到達下一個地點，隨即將封包進行丟除的動作，並將回傳一個無法到達目的地的錯誤訊息回去，裡面會包覆著原本進行傳輸的IP封包以及ICMP的請求封包，Type 3並沒有Identifier、Sequence Number欄位，取而代之的是Unused，為一個4 bytes的欄位，作為保留欄位並且必須要為0，當未來有需要的話會進行擴展功能，Internet Header + 64 bits of Original Data Datagram代表原本進行請求但被捨棄的IP及ICMP封包。

Type	Code	Checksum
Unused		
Internet Header + 64 bits of Original Data Datagram		

表4-2　Type 3封包欄位

Code	Mean
0	Network Unreachable
1	Host Unreachable
2	Protocol Unreachable
3	Port Unreachable
4	Fragmentation Needed and DF set
5	Source Route Failed
6	Destination network unknown
7	Destination host unknown
8	Source host isolated
9	Communication with destination network administraively prohibited
10	Communication with destination host administraively prohibited
11	Network unreachable for type of service
12	host unreachable for type of service
13	Communication Administratively Prohibited
14	Host Precedence Violation
15	Precedence cutoff in effect

圖4-6　Type 3 Code種類與意義

Type 4 (Source Quench)，是一個錯誤回報訊息，Code只有0，表4-3為Type4的封包欄位，我們主要對Code與Unused欄位進行模糊測試，當路由器負載時，用來遏止來源繼續發送訊息，其功能主要是用來作為IP流量控制和壅塞控制的一種方法。

Type	Code	Checksum
Unused		
Internet Header + 64 bits of Original Data Datagram		

表4-3　Type 4封包欄位

Type 5 (Redirect)，是一種錯誤回報訊息，Code 有 0、1、2、3，分別為 Redirect Datagram for the Network (or subnet)、Redirect Datagram for the Host、Redirect Datagram for the Type of Service and Network、Redirect Datagram for the Type of Service and Host，表 4-4 為 Type 5 封包欄位，功能為用來回報重新導向路由的路徑，Type 5 與其他的錯誤訊息不同的是路由器並沒有收到的資料進行丟棄的動作，而是將它傳送到正確的路由器，也就是圖 4-7 中的 Gateway Address 欄位。

Type	Code	Checksum
Gateway Internet Address		
Internet Header + 64 bits of Original Data Datagram		

表4-4　Type 5封包欄位

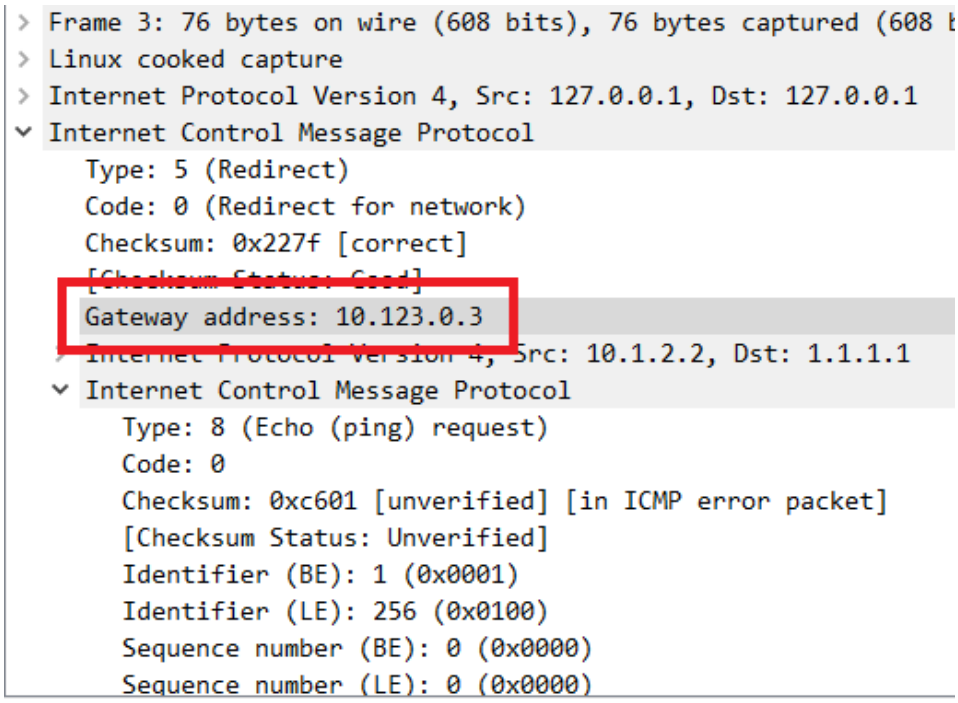


圖4-7　Type 5 Wireshark Gateway

Type 10 (Router Solicitation)與Type 9 (Router Advertisement)，是一種詢問訊息，Type 10 Code只有0，Type 9 Code有0、16，分別為Normal router advertisement、Does not Router common traffic，表4-5為Type 10欄位圖，表4-6為Type 9欄位圖，來源端主機一開始會對發出RS的廣播請求Router，當Router接收到來源端主機的請求後，Router會藉由來源端主機的IP進行RA的廣播去進行回應的動作，在Type 10欄位部分，Reserved為保留欄位，並不具有任何功能，Type 9欄位部分，Num Addrs代表Router進行廣播的數量，Addr Entry Size代表每個Router地址中的訊息(32 bit的訊息量)，Lifetime代表Router進行廣播時可存在的最大秒數，Router Address代表進行廣播的Router位址，Preference Level代表回傳給來源端主機後會被進行承認的等級，回傳Router的等級越高，越容易被來源端主機認可。

Type	Code	Checksum
Reserved		

表4-5　Type 10封包欄位

Type	Code	Checksum
Num Addrs	Addr Entry Size	Lifetime
Router Address[1]		
Preference Level[1]		
Router Address[2]		
Preference Level[2]		
……		
……		

表4-6　Type 9封包欄位

Type 11 (Time Exceeded)，是一種錯誤回報訊息，Code有0跟1，分別為 time to live exceeded in transit 與 fragment reassembly time exceeded，表4-7為Type 11的封包欄位，我們主要對Code與Unused欄位進行模糊測試，功能為當封包在傳送中逾時被被丟棄後進行告知的動作。

Type	Code	Checksum
Unused		
Internet Header + 64 bits of Original Data Datagram		

表4-7　Type 11封包欄位

Type 12 (Parameter Problem on a Datagram)，是一種錯誤回報訊息，Code有0、1、2，分別為Pointer indicates the error、Missing a Required Option、Bad Length，表4-8為Type 12的封包欄位，當路由器或目的地主機發現Datagram的任何欄位有定義不清楚的或數值不見的情況，這個封包就會被丟掉，並且送出一個錯誤訊息回傳送端，其中Pointer為指標，如圖4-8所見，若指標為0，代表檢測到錯誤。

Type	Code	Checksum
Pointer	Unused	
Internet Header + 64 bits of Original Data Datagram		

表4-8　Type 12封包欄位

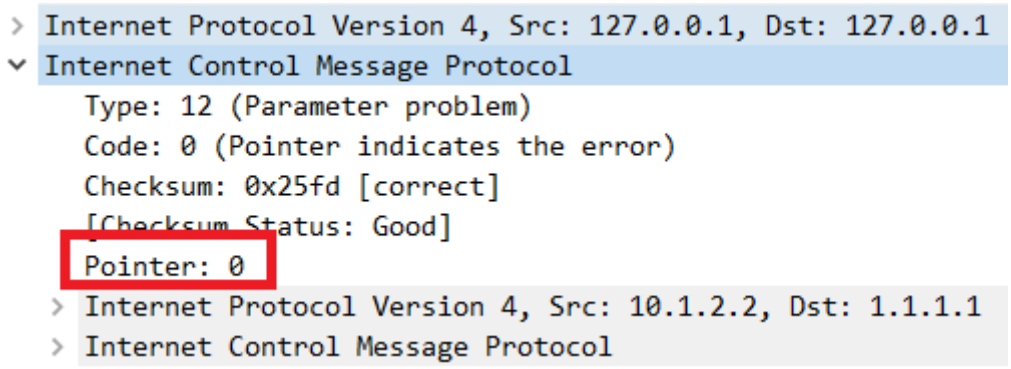


圖4-8　Pointer為0

Type 13 (Timestamp Request)與Type 14 (Timestamp Reply)，是一種詢問訊息，只有Code0，Type 13功能為要求對方送出時間訊息，用以計算路由時間的差異，而Type14是用於回應Type13用，表4-9為Type 13與14的封包欄位，可以看到除了基礎欄位以外還多了Originate Timestamp、Receive Timestamp、Transmit Timestamp，分別記錄出發時的起始時間、接收時的時間以及傳送的所耗費的時間，我們將對Code與Identifier、Sequence Number、Originate Timestamp、Receive Timestamp、Transmit Timestamp欄位進行模糊測試，如圖4-9可以看到時間來源端的要將送出時的國際標準時間填入Originate Timestamp的欄位中，而另外兩個則是填為0，當目的端收到請求後，如圖4-10，會將另外兩個時間戳記欄位填上數值。

Type	Code	Checksum
Identifier		Sequence Number
Originate Timestamp		
Receive Timestamp		
Transmit Timestamp		

表4-9　Type 13、14封包欄位

```
Originate timestamp: 16131062 (4 hours, 28 minutes, 51.062 seconds after midnight UTC)
Receive timestamp: 0 (0 seconds after midnight UTC)
Transmit timestamp: 0 (0 seconds after midnight UTC)
```

圖4-9　Type 13時間戳記數值

```
Originate timestamp: 16131062 (4 hours, 28 minutes, 51.062 seconds after midnight UTC)
Receive timestamp: 26169576 (7 hours, 16 minutes, 9.576 seconds after midnight UTC)
Transmit timestamp: 26169576 (7 hours, 16 minutes, 9.576 seconds after midnight UTC)
```

圖4-10　Type 14時間戳記數值

Type 15 (Information Request)與Type 16 (Information Reply)，是一種詢問訊息，只有Code 0，表4-10為Type 15與Type 16的封包欄位，在RARP（Reverse Address Resolution Protocol）協定應用前，此訊息用於在開機時取得網路訊息，當來源端想知道某個主機的IP時，可以透過Type 15與Type 16進行訊息請求回應。

Type	Code	Checksum
Identifier		Sequence Number

表4-10　Type 15、16封包欄位

Type 17 (Address Mask Request)與Type 18 (Address Mask Reply)，是一種詢問訊息，只有Code 0，Type 17是用來查詢子網路的Mask，Type18用於回應Type 17，表4-11為Type 17與Type 18的封包欄位，當我們想知道某個子網路遮罩時(Mask)，可由來源端發出一個Type 17的請求gateway，若該gateway無反應，如圖4-11可以利用廣播的方式將此訊息送出，等待gateway的回應。

Type	Code	Checksum
Identifier		Sequence Number
Address Mask		

表4-11　Type 17、18封包欄位

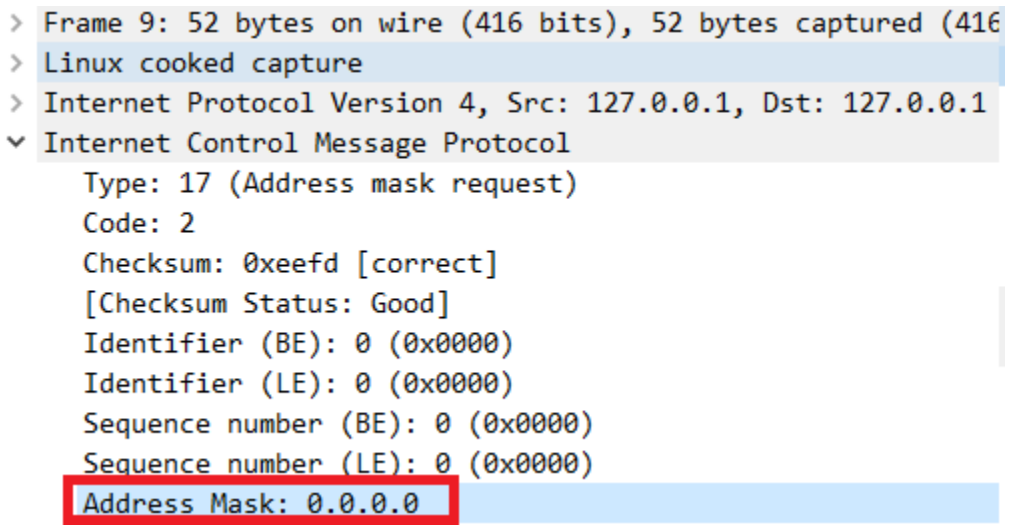


圖4-11　Request利用廣播送出

Type 30 (Traceroute)，是一種詢問訊息，現今的Traceroute多半仍是以TTL去計算，運行原理是因為使用IP的option所附加的資訊去進行追蹤，表4-12為Type 30的封包欄位，但是如圖4-12所視，在Wireshark中並無法被讀取大部分的欄位，ID Number代表從IP option中提取的號碼，並不是IP Header中的ID，Unused代表未使用的被保留欄位，Outbound Hop Count為從IP option中提取的跳站數，代表所需經過Router數量，Return Count為從IP option中提取的返回數量，代表從Router中所返回的次數，Output Link Speed代表以發送與返回次數的每秒連結數率，若無法確定這個值，其值預設應為0，Output Link MTU代表發送與返回的最大傳輸單元。

Type	Code	Checksum
ID Number		Unused
Outbound Hop Count		Return Hop Count
Output Link Speed		
Output Link MTU		

表4-12　Type 30封包欄位

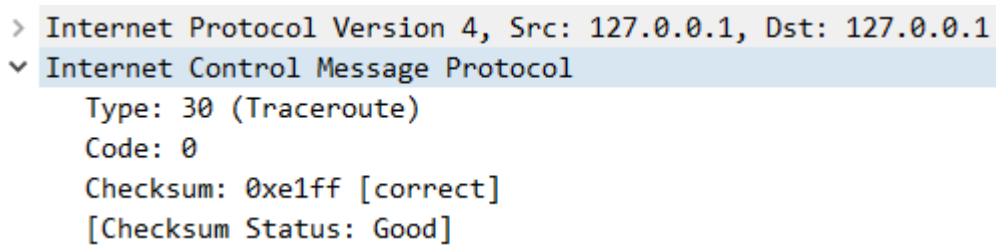


圖4-12　Type 30 Wireshark上無法讀懂欄位

Type 31 (Datagram Conversion Error)，一種錯誤訊息，表4-13為Type 31的封包欄位，但是如圖4-13所視，在Wireshark中並無法被讀取大部分的欄位，當Datagram在網路層無法被有效的轉換時，就會發出這樣的錯誤訊息，Pointer to problem area會指出錯誤部分是從哪邊開始出錯，Copy of datagram that could not be converted會把錯誤部分的Datagram放到這個欄位。

Type	Code	Checksum
Pointer to problem area		
Copy of datagram that could not be converted		

表4-13　Type 31封包欄位

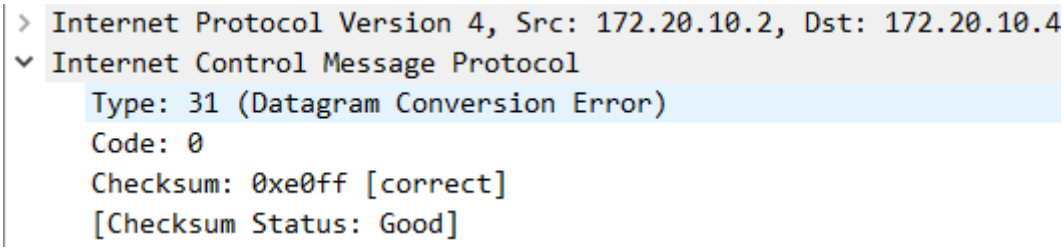


圖4-13　Type 31 Wireshark上無法讀懂欄位

Type 37 (Domain Name Request)與Type 38 (Domain Name Reply)，是一種詢問訊息，表4-14為Type 37的封包欄位，表4-15為Type 38的封包欄位，但是如圖4-14與圖4-15所視，在Wireshark中並無法被讀取大部分的欄位，功能是藉由Type 37進行單播或廣播的請求回應，當目的地IP收到請求後，會藉由Type 38回應一個Domain Name給來源端IP，Time-To-Live代表封包可存活的時間，Names代表所要求的Domain Name，長度由Datagram的長度決定，若沒有Domain Name(長度為0)，則回覆不存在Domain Name，當有多個Domain Name被回應時，應該把這些Domain Name都列出來，不符合MTU的Domain Name也不會被進行回應的動作。

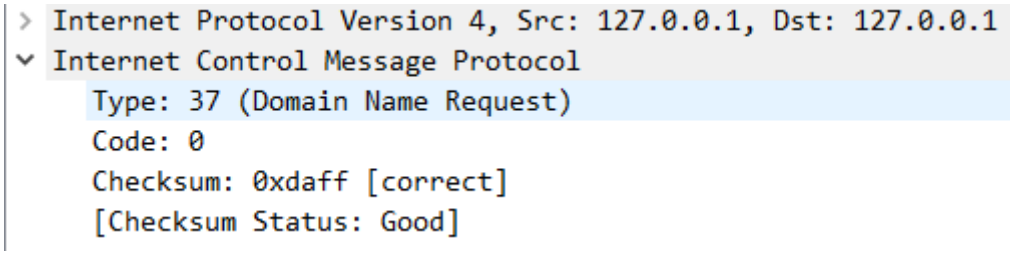


圖4-14　Type 37 Wireshark上無法讀懂欄位

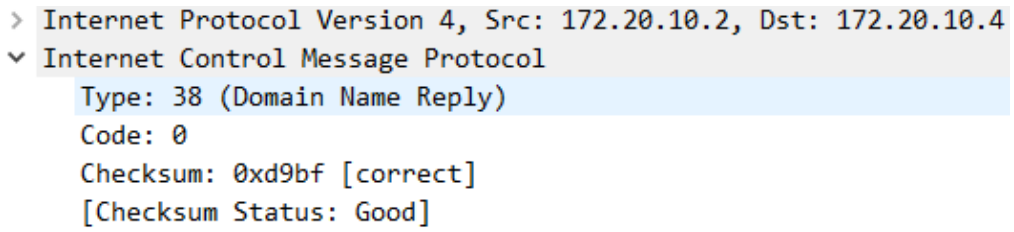


圖4-15　Type 38 Wireshark上無法讀懂欄位

Type	Code	Checksum
Identifier		Sequence Number

表4-14　Type 37封包欄位

Type	Code	Checksum
Identifier		Sequence Number
Time-To-Live		
Names		

表4-15　Type 38封包欄位

Type 40 (Photuris)，是一種錯誤訊息，Code有0、1、2、3、4、5，表4-16為Type 40的封包欄位，但是如圖4-16所視，在Wireshark中並無法被讀取大部分的欄位，本訊息的功能用於指出AH及ESP安全協議的錯誤，Reserved代表保留欄位，未來有可能被作為擴充功能，但目前無用並強制填為0，Pointer代表IP中違規的SPI(Security Parameters Index)，若SPI不存在，該欄位應填為0，Original Internet Headers + 64 bits of payload代表原始所送出的IP Header以及前8 bytes payload的資料。

Type	Code	Checksum
Reserved		Pointer
Original Internet Headers + 64 bits of payload		

表4-16　Type 40封包欄位

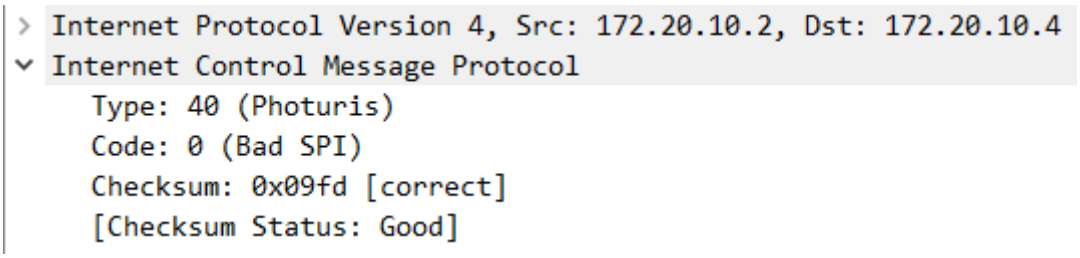


圖4-16　Type 40 Wireshark上無法讀懂欄位

Type 41 (ICMP messages utilized by experimental mobility protocols such as Seamoby)，表4-17為Type 41的封包欄位，但是如圖4-17所視，在Wireshark中並無法被讀取大部分的欄位，Type 41主要在說明IANA(網路指派機構)用來測試一些新功能的協定，其目的旨在對Candidate Access Router Discovery (CARD) Protocol 與 Context Transfer Protocol (CXTP)這兩個協定是被設計去加速無線網路到Router之間的切換速度，欄位部分，Subtype用於識別experimental mobility protocols，Reserved代表一個未定義的擴充欄位，除非實驗有特別定義，否則都是設值為0，Options作為experimental mobility protocols所定義的內容事項。

Type	Code	Checksum
Subtype	Reserved	
Options		

表4-17　Type 41封包欄位

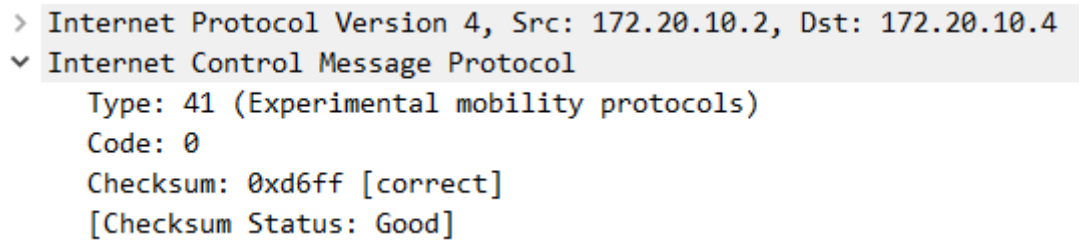


圖4-17　Type 41 Wireshark上無法讀懂欄位

Type 42 (Extended Echo Request)與Type 43 (Extended Echo Reply) 是一種詢問訊息，表4-18為Type 42的封包欄位，表4-19為Type 42的封包欄位，但是如圖4-18與圖4-19所視，在Wireshark中並無法被讀取大部分的欄位，從本地接口(Interface)發送Extended Echo Request (探測接口)，送至Proxy接口，Proxy Interface會停留在Proxy節點(Node)，當Proxy Interface收到ICMP的Extended Echo Request時，Proxy Node會建立好控制程序，若Proxy Node可以進行存取，Proxy節點會確認被探測接口的狀態，並回傳一個Extended Echo Reply，並表示被探測接口的狀態，在Type42中，Reserved代表著保留欄位，L代表local的意思，若被探測接口位在Proxy節點上進行設置L-bits的動作，若被探測接口直接連結到Proxy節點上則此欄位進行清除的動作，ICMP Extension這個欄位會對被探測的接口進行標記(Identifies)，Type 43中，若code不為0，則State欄位要設為0且收到時要被忽略，Res這個欄位必須要被填為0且回傳收到時會被忽略，若Code欄位為0且被探測接口在proxy節點上，probed要是進行活動的狀態的話則A-bit成立，否則A–bit欄位被清除，在A-bit成立的前提下，若使用IPv4則4的欄位成立(4-bit)，並且另一個欄位則會被捨棄掉(6)，相反的IPv6也是如此。

RFC 8335[10]中講述著一種PROBE的網路診斷工具，它的功能與Ping一樣可以作為查詢探測接口狀態的工具，不同點在於，它並不需要雙向連結(bidirectional connectivity)，但是它需要Probed和代理(Proxy)的雙向連結，Proxy可以與Probed Interface在同一節點(Node)上，也可位於與探測接口直接連結的接點。

Type	Code	Checksum		
Identifier		Sequence Number	Reserved	L
ICMP Extension				

表4-18　Type 42封包欄位

Type	Code	Checksum					
Identifier		SequenceNumber	State	Res	A	4	6

表4-19　Type 43封包欄位

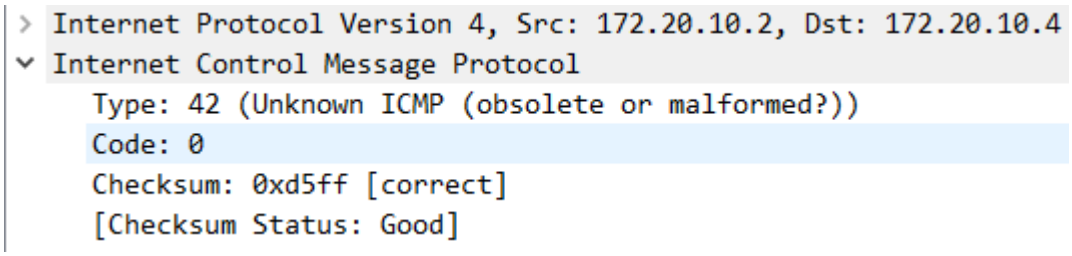


圖4-18　Type 42 Wireshark上無法讀懂欄位

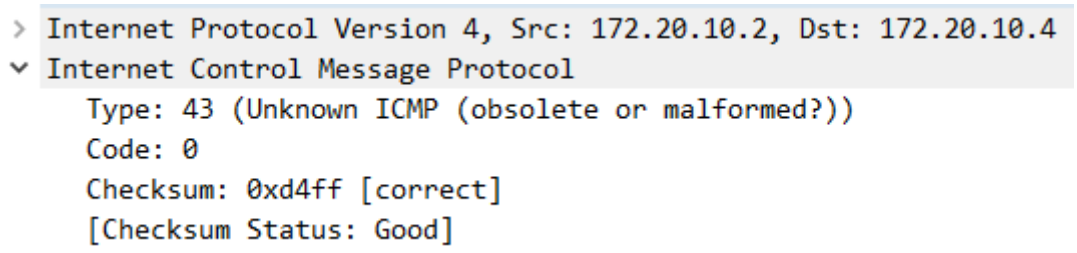


圖4-19　Type 43 Wireshark上無法讀懂欄位

從Type 0到Type 43中，我們將所有可以進行模糊測試的欄位都測試了一遍，但是並沒能夠找出存在的漏洞，於是我們決定對最常見的Type 8與Type 0再次進行測試，與之前不同的是，這次我們將輸入所有可能存在的測試資料進行測試，藉此來驗證ICMP協定到底有無漏洞。

圖 4-20 與圖 4-21 分別為 Type 8 與 Type 0 中所有進行模糊測試後 Wireshark的pcapng檔，Type 8的總抓取封包數為528614，Type 0的總抓取封包數為401136，兩個Type大概都送出了13萬筆測資，所有的測試資料都跑完以後，模糊測試程式並沒有跑出找出漏洞的訊息，表示我們所測試的協定是安全無漏洞的。

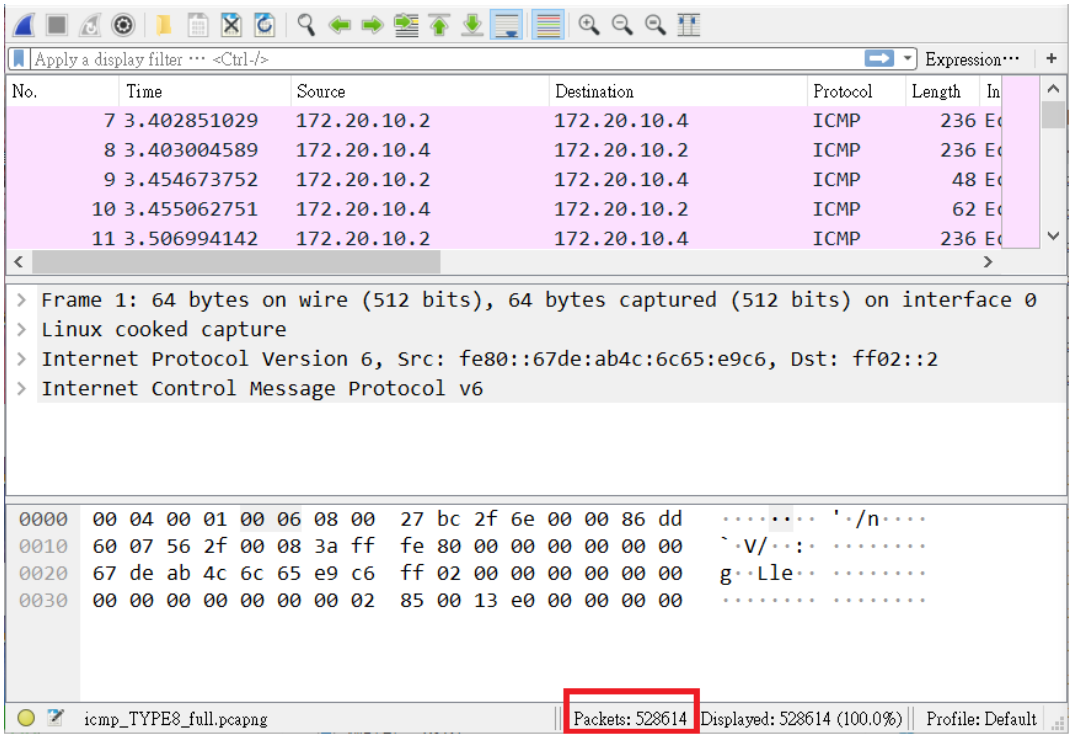


圖4-20　Type 8完整測試

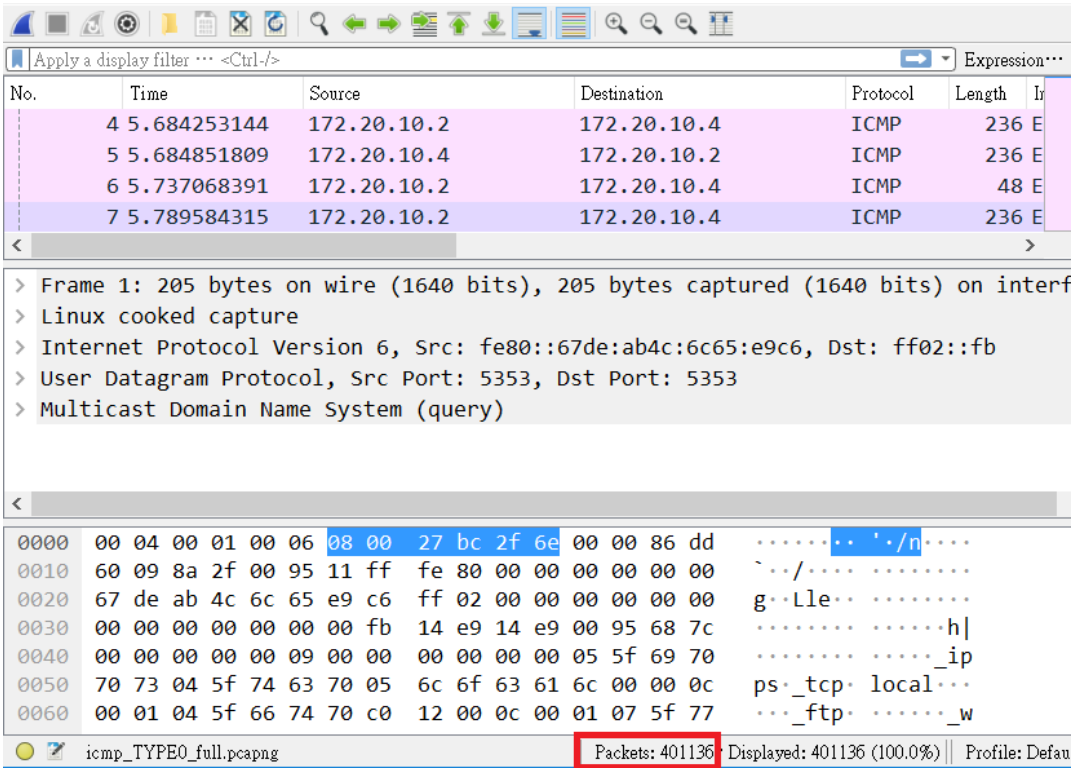


圖4-21　Type 0完整測試

第五章 結論

在現今網路發達的時代中，網路安全的議題是非常重要的，所以才出現了保障網路安全性的工具，因此像是模糊測試這樣的網路檢測工具開始慢慢地出現。本研究中主要探討ICMP協定的模糊測試，許多長久發展下的協定內容都非常的穩定，如:ICMP，即便如此，這些協定仍舊不斷的更新，舉例來說Type 42與Type 43為今年6月份出的最新協定。本次研究的重點在於將這些已經被定義在RFC[11]文件上的ICMP協定內容進行測試，以確認協定中是否存在漏洞。

我們將Type 0到Type 43中可以進行模糊測試的欄位都實際測試過了一遍，就如前面所敘述的問題，我們不太可能將所有在欄位中需要用到2bytes以上的測試資料全部跑過，因為這樣做效率會非常的差且無法達到我們所預期的產生隨機測試資料的成效，但是在對最常見的Type 0與Type 8進行模糊測試時，我們有將1 byte所有的測試資料進行實測，結果並沒有造成系統錯誤。藉此證明Type 0與Type 8並沒有漏洞。

本研究讓我們學習到了在ICMP協定上的功能是什麼以及在網路上是如何運作，在學習ICMP協定的過程中利用了Socket進行了Ping程式碼的撰寫，並且也理解到了模糊測試的技術與使用方法，最重要的是我們能夠理解整個研究的過程及方法，在未來我們接觸到類似的東西時可以更快的理解該如何去使用。

第六章 參考文獻

[1]DDos: https://myracloud.com/zh-hant/what-is-a-ddos-attack/

[2] Behrouz A.Forouzan（2011）。TCP/IP 通訊協定（莊承翊、張佑瑋、蕭衛鴻、蘇致遠，譯）。台北：全華。（原著第四版出版於2010年）

[3]Death of Ping(2016): https://en.wikipedia.org/wiki/Ping_of_death

[4]模糊測試(Fuzz testing): https://en.wikipedia.org/wiki/Fuzzing

[5]OSS-Fuzz(2016): https://github.com/google/oss-fuzz

[6] Peach Fuzz(2004): https://www.cnblogs.com/baoyu7yi/p/7264971.html

[7]ICMP協定內容:

http://www.pcnet.idv.tw/pcnet/network/network_ip_icmp.htm

[8]Github Ping程式碼(2014):

https://github.com/emamirazavi/python3-ping/blob/master/ping.py

[9]Github Traceroute程式碼(2010): https://gist.github.com/pnc/502451

[10]RFC8335(2018):https://tools.ietf.org/html/rfc8335

[11]RFC792(1981): https://tools.ietf.org/html/rfc792