ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

Campus di Cesena - Scuola di Scienze

Corso di Laurea in Scienze e Tecnologie Informatiche

ANALISI DI ATTACCHI BASATI SU PORT SCANNING

Relazione finale in Reti di Calcolatori

Relatore

Gabriele D'Angelo

Presentata da

Giuseppe Giorgino

Terza sessione

Anno Accademico 2013 / 2014

Indice

1. Port Scanning

1.1 Panoramica sul port scanning

Il tema che ci accingiamo a trattare è il port scanning il quale fa parte di una problematica importante nell'Information Technology, cioè la sicurezza delle reti.

Da quest'ultima dipende l'integrità della nostra privacy, delle nostre risorse e dei nostri dati, un argomento che sempre più sta acquisendo visibilità e rilevanza e sebbene Internet nacque con ideali di condivisione e fiducia, etica tipica dell'ambiente accademico, oramai è un pensiero abbandonato a fronte di un mero scopo di lucro.

L'obiettivo di questa tecnica consiste nella raccolta di importanti informazioni riguardanti uno o più host collegati alla rete, in particolar modo si concentra sulla possibilità di stabilire quali porte, e di conseguenza quali servizi, siano in ascolto sulla macchina. Se il concetto di porta nell'ambito delle reti di calcolatori non è ancora noto al lettore, lo sarà a breve, infatti sarà oggetto di approfondimento nel secondo capitolo.

L'attività di port scanning di per sé non risulta pericolosa, viene utilizzata comunemente dagli amministratori di sistema per gestire controlli e manutenzione, tuttavia queste informazioni se ottenute illecitamente da malintenzionati, possono essere utilizzate per portare a termine un attacco vero e proprio, tale da poter sottrarre per esempio un qualsiasi dato sensibile di un' azienda o di un privato. In pratica è come se un ladro tentasse di controllare se una qualsiasi porta o finestra di casa, sia aperta o meno.

Per dimostrare come questa tecnica abbia una potenzialità enorme, vorrei presentare brevemente il lavoro svolto da un anonimo ricercatore [1], il quale dopo alcuni tentativi di port scanning svolti quasi per caso, si rese conto dell'incredibile numero di device aventi porte aperte su servizi al quale si poteva accedere semplicemente inserendo credenziali di default. In ogni vittima inseriva inoltre un eseguibile, il quale gestiva in automatico operazioni di port scanning e di intrusione creando così un vero e proprio port scanner distribuito. Questa operazione è ovviamente illegale ed eticamente sbagliata, anche se l'autore sottolinea come il codice inserito

venisse eseguito con bassa priorità dalla CPU e solo per un numero limitato di giorni, senza quindi la volontà di ledere la privacy o i dati del malcapitato. In totale riesce così a creare una botnet[1] di 420.000 device, attraverso la quale riesce a determinare la geolocalizzazione di tutti gli indirizzi IP che rispondono a richieste di ping oppure che possiedono porte aperte, fornendo così quasi un censimento degli indirizzi IPv4 assegnati nelle varie parti del mondo. L'immagine 1. ci mostra oltre 460 milioni di indirizzi IPv4 raggiungibili, osservati fra il giugno del 2012 e l'ottobre dello stesso anno.

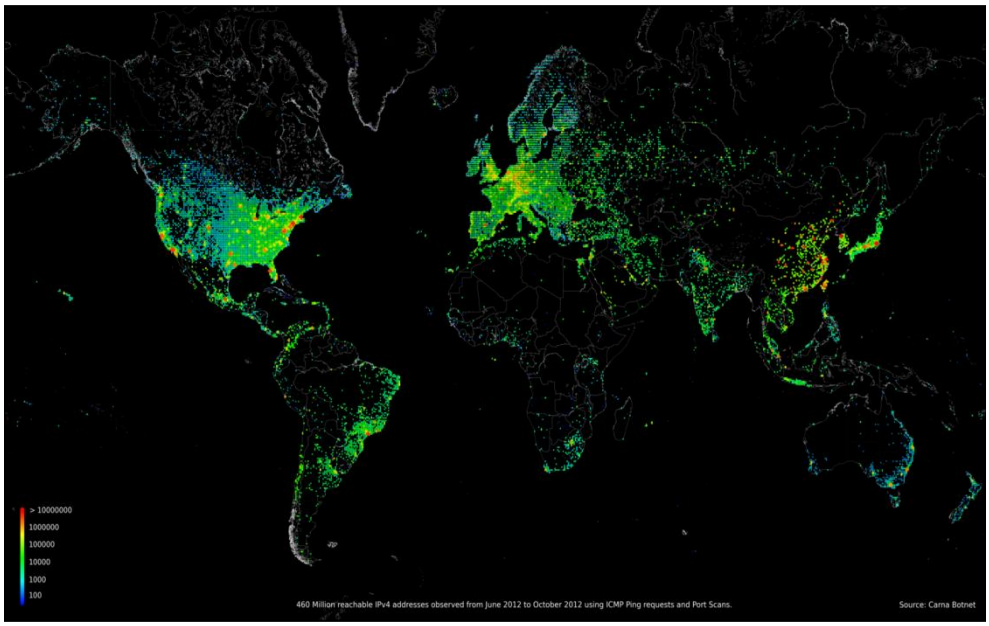


Immagine 1. Censimento degli indirizzi IPv4 nel 2012 [1]

Sicuramente non può essere un censimento vero e proprio, ma ci fornirà un'idea della vastità di Internet e ci fa riflettere sia sulla potenzialità del port scanning dal punto di vista di un attaccante, ma anche di come sia importante difendersi per non essere nelle mani di un potenziale malintenzionato.

[1] Una botnet è una rete costituta da host infettati, chiamati zombie o bot, i quali verranno controllati da un'unica entità chiamata botmaster. Il controllore avrà la possibilità di gestire i sistemi compromessi per poter effettuare attacchi su larga scala.

Sarà proprio questo un punto focale dell'elaborato, dimostrare come tecniche, suggerimenti o strumenti riguardanti questo argomento, possano essere manipolati da un attaccante per mettere in atto intrusioni ma d'altro canto devono anche essere un'importante fonte di conoscenza per potersi difendere, un dualismo che porteremo avanti come filo comune in tutta la trattazione.
Già da queste poche parole è facile accorgersi come l'argomento non sia sempre di rapida comprensione, sarà quindi fornita inizialmente una base di conoscenze minime ma sufficienti per poter avanzare nella lettura senza doversi prima necessariamente informare da altre fonti. Il lettore di conseguenza sarà condotto per tutto l'elaborato il più possibile per mano attraverso argomenti più o meno avanzati, fornendo riassunti, brevi spiegazioni, note e quant'altro che faranno da guida in particolar modo a chi possiede solamente una conoscenza delle reti poco più che basilare. Non mancheranno di certo opportuni riferimenti biografici a libri, documentazioni ufficiali o materiale online atto a garantire il giusto approfondimento anche ai più curiosi ed interessati.

1.2 Obiettivi

In questo paragrafo si desidera provvedere a dare una visione d'insieme di quella che sarà l'attività svolta nello sviluppo della tesi. Un punto focale di interesse sarà costituito, come già esposto nell'introduzione precedente, dal fornire una trattazione il più possibile completa e precisa sull'attività di port scanning e da tutti quegl'elementi che la compongono. Non volendosi però fermare ad una sola descrizione degli argomenti, si è deciso quindi di sviluppare un'analisi sulle principali tecniche utilizzate prendendo in esame diverse variabili, come utilizzo della CPU e della banda, tempo utilizzato, ecc., le quali verranno considerate per porre punti di confronto tra le suddette tipologie d'attacco esaminate. Per poterle studiare faremo uso di un sistema di honeypot[2] il quale ci mette a disposizione un ambiente facilmente

[2] Honeypot è un sistema o componente hardware o software usato come esca a fini di protezione da attacchi informatici.

individuabile ed attaccabile ed inoltre tutta una serie di strumenti che permettono di registrare le attività intrusive dalle quali estrarremo le informazioni più utili allo svolgimento del progetto. Il terzo punto di maggior attenzione sarà occupato nel rendere il più omogeneo e coeso possibile l'unione delle due precedenti attività, onde evitare che il lettore abbia il sentore di leggere solo una lunga serie di informazioni.

2. Conoscenze essenziali

2.1.Nozioni su porte, protocollo ICMP e segmenti TCP

In questo capitolo si vuole fornire un background di conoscenze adeguato e preciso, per tutti quegli argomenti con i quali il lettore potrebbe non avere dimestichezza. Riprendiamo quindi il concetto, espresso nell'introduzione, del funzionamento generale del port scanning, cioè l'invio di messaggi all'host bersaglio, per stabilire quali porte siano aperte e i relativi servizi attivi in base alle risposte ricevute. Da questa definizione molto semplicistica, possiamo già definire due domande: cosa si intende per porte nell'ambito delle reti di calcolatori? Che tipologia di pacchetti vengono inviati o ricevuti? Per rispondere alla prima domanda è sufficiente pensare a come, in un ambiente multitasking, sia necessario che in un medesimo server siano in esecuzione più servizi in contemporanea e tutti raggiungibili anche da un medesimo client, è quindi indispensabile uno strumento in grado di permettere questo servizio di multiplexing, cioè le porte. Esse non sono altro che numeri individuati tra 1 e 65535 (per TCP lo 0 è riservato e non utilizzato mentre per UDP il numero di porta del mittente è opzionale ed il valore 0 ne indica appunto l'assenza), e possono essere categorizzate in 3 grandi intervalli: Well Known Ports (1-1023), Registered Ports (1024-49151), e Dynamic and/or Private Ports (49152-65535). Le porte note sono assegnate dalla IANA (Internet Assigned Numers Authority) e per lo più su di esse sono in ascolto applicazioni di sistema e con funzionalità di server, per esempio programmi che implementano il protocollo HTTP sono su porta 80, FTP sulla 21, SSH la 22 e SMTP la 25.

Normalmente una porta assume lo stato di aperta o chiusa ma gran parte dei port scanner moderni le classifica in ulteriori quattro categorie. Questi stati aggiuntivi non sono proprietà intrinseche delle porte stessa ma sono appunto suddivisioni atte a fornire una visione più completa della situazione della porta. Vedremo quindi brevemente queste sei suddivisioni:

1. Open – su questa porta vengono accettate connessioni TCP o UDP per l'applicazione che su di essa è in ascolto. Trovare questa tipologia di porte è l'obiettivo principale del port scanning poiché rappresentano

una possibile via d'attacco, mentre gli amministratori di rete e i sistemisti tendono a chiuderle o proteggerle con firewall senza limitare però l'uso dell'utente.

2. Closed – su questa porta è possibile sia inviare che ricevere pacchetti ma la particolarità è che non è in ascolto alcuna applicazione su di essa, ciò non di meno per alcune tipologie di port scanning come il ping scanning, di cui parleremo nei prossimi capitoli, risulta essenziale sapere se un host è attivo su un certo indirizzo IP.
3. Filtered – nel caso in cui un packet filter[3] sia abilitato sulla macchina bersaglio, il lavoro del port scanner si complica parecchio poiché risulta difficile accertarsi con sicurezza se una porta in questo caso sia aperta o chiusa. A volte queste tipologie di porte rispondono con un messaggio ICMP (Internet Control Message Protocol), la cui trattazione seguirà a breve, di tipo 3 codice 13 (destination unreachable: communication administrative prohibited) anche se la maggior parte dei packet filter semplicemente ignora i tentativi di connessione, costringendo così al rinvio dei pacchetti d'intrusione per doversi accertare che non siano stati scartati a causa della congestione piuttosto che dal firewall, rallentando di conseguenza tutta l'attività d'attacco.
4. Unfiltered – questa porta risulta accessibile ma non classificabile come aperta o chiusa. Questo stato risulterà utile solo in determinare tipologie di port scanning.
5. Open|filtered – questo stato è determinato dall'incertezza nel classificare una determinata porta come aperta o filtrata, ciò accade nelle scansioni in cui la suddetta porta non risponda in alcun modo a causa di packet filter, di problemi di congestione o simili.
6. Cosed|filtered – questo stato è usato quando non si è grado di determinare se una porta sia chiusa o filtrata ed è usato solamente per la tecnica di port scanning chiamata Idle scan (vedi paragrafo 4.1).

[3] Un packet filter è una tipologia di firewall, il quale svolge un importante operazione di controllo dei diversi header dei pacchetti esaminati ed attraverso regole prestabilite ne permette o meno il passaggio.

Poco sopra abbiamo accennato ai messaggi ICMP ma cosa sono esattamente? Per prima cosa è bene precisare che il protocollo ICMP, descritto nell'RFC 792 [2], gestisce l'error reporting del protocollo IP poiché esso offre un servizio di tipo best-effort e di conseguenza non sono previsti meccanismi di verifica né di recupero o notifica dei datagrammi persi, indi per cui si rende necessario il protocollo ICMP per la trasmissione dei messaggi di errore e di controllo. Esso è incluso in tutte le implementazioni IP ed è un protocollo a basso livello che si appoggia direttamente su IP in una sorta di sub-strato. I messaggi ICMP sono incapsulati direttamente nei datagrammi IP, fornendogli di fatto da vettore di trasporto. Definiamo inoltre qui di seguito la struttura dei messaggi ICMP.

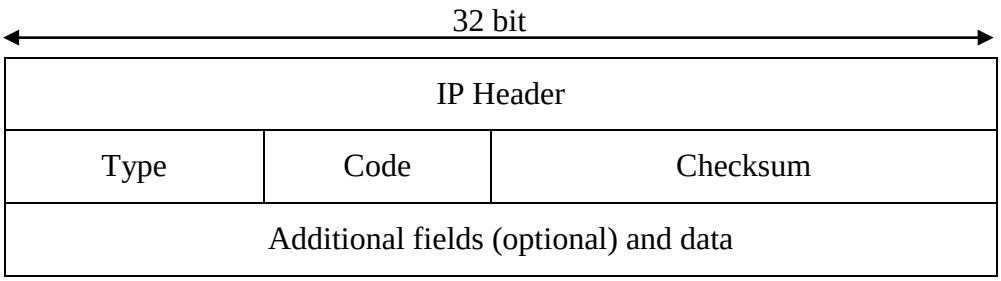


Immagine 2. Struttura di un messaggio ICMP

Type (8 bit) – definisce il tipo di messaggio ICMP

Code (8 bit) – descrive ulteriormente il tipo di errore o la notifica riscontrata.

Checksum (word da 16 bit) – controlla la correttezza del messaggio

Data - dipendono dal tipo di messaggio ICMP, in genere intestazione e parte dei dati del datagramma che ha generato l'errore.

La tipologia di messaggi che ci interessa maggiormente sono quelli di tipo 3 (Destination Unreachable), esso viene generato da un gateway nel caso in cui non sia raggiungibile la relativa sottorete oppure l'host, mentre viene notificato da un host quando è presente un errore sull'indirizzo dell'entità di livello superiore a cui trasferire il datagramma. I code possibili con un messaggio di tipo 3 sono:

Code	Descrizione
0	Destination network unreachable
1	Destination host unreachable
2	Destination protocol unreachable
3	Destination port unreachable
4	Fragmentation required, and DF flag set
5	Source route failed
6	Destination network unknown
7	Destination host unknown
8	Source host isolated
9	Network administratively prohibited
10	Host administratively prohibited
11	Network unreachable for TOS
12	Host unreachable for TOS
13	Communication administratively prohibited
14	Host Precedence Violation
15	Precedence cutoff in effect

Immagine 3. Elenco di code possibili con relative descrizioni

In conclusione è opportuno notificare anche i messaggi di tipo 0 codice 0 (Echo reply) e quelli di tipo 8 codice 0 (Echo request) i quali vengono utilizzati nell'utility ping.

Per poter individuare quali porte siano aperte o meno, molte delle tecniche utilizzate sfruttano alcuni campi presenti nell'header dei segmenti inviati tramite il protocollo di trasporto TCP. Per fornire un'idea più chiara e ampia di quali siano questi campi e di come funzionino, provvediamo quindi ad un'esposizione della struttura di un segmento TCP.

Bit offset	Bits 0-3	4-7	8-15								16-31
0	Source port										Destination port
32	Sequence number										
64	Acknowledgment number										
96	Data offset	Reserved	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window size
128	Checksum										Urgent pointer
160	Options (optional)										
160/ 192+	Data										

Immagine 4. Segmento TCP

- Source port (16 bit): identifica il numero di porta dell'host mittente
- Destination port (16 bit): identifica il numero di porta dell'host destinatario
- Sequence number (32 bit): rappresenta il numero del primo byte del segmento nel flusso di byte nel caso in cui il flag di SYN sia 0, mentre se è stato settato a 1, il numero di sequenza funge da Initial Sequence Number (*ISN*) e viene negoziato all'apertura della connessione in modo casuale.
- Acknowledgment number (32 bit): numero di riscontro, ha significato solo se il flag ACK è impostato a 1, e conferma la ricezione di una parte del flusso di dati nella direzione opposta, indicando il valore del prossimo sequence

number che l'host mittente del segmento TCP si aspetta di ricevere (non è nient'altro che il numero di sequenza del prossimo byte atteso dall'altro lato).

- Data offset (4 bit): indica la lunghezza (in word da 32 bit) dell'header, quest'ultimo infatti può variare da 5 word (20 byte) a 15 word (60 byte) di lunghezza, in base al campo facoltativo options.
- Reserved (4 bit): bit non utilizzati e predisposti per sviluppi futuri del protocollo, di default sono settati a 0.
- Flags (8 bit): bit per il controllo e la gestione del protocollo:

    1. CWR (Congestion Window Reduced) – se acceso, indica la ricezione da parte dell'host sorgente di un segmento TCP con il flag ECE impostato a 1, cioè il mittente sta notificando al ricevente che la finestra di congestione (congestion window) è stata ridotta a causa dei vari meccanismi che gestiscono il controllo della congestione (questo flag non è presente nell'RFC 793 [3], cioè quello che nel 1981 definì lo standard per l'header dei segmenti TCP, ma bensì è stato aggiungo nell' RFC 3168 [4] del 2001)
    2. ECE (ECN-Echo) – se risulta settato a 1 durante il three way handshake ciò significa che l'host è in grado di supportare l'ECN (Explicit Congestion Notification). Quest'ultimo definisce una tecnica differente per il controllo della congestione rispetto a quella standard, infatti piuttosto che droppare i pacchetti, i router impostano il flag CE (congestion experienced) nell'header IP. Nel caso in cui un host riscontra una congestione in corso può così settare a 1 il flag ECE, notificarlo al mittente il quale risponderà con un segmento il cui flag CWR sarà acceso e attiverà i vari meccanismi (anche questo flag è stato aggiunto solamente nell'RFC 3168).
    3. URG - se attivo indica che nel flusso sono presenti dati urgenti.
    4. ACK – se attivo indica che il campo acknowledgment number è valido.
    5. PSH – se attivo indica che i dati in arrivo non devono essere bufferizzati, ma inviati direttamente all'applicazione.
    6. RST – se attivo determina il reset immediato del collegamento. Durante una normale connessione, a meno che non venga inviato un

segmento con il flag SYN acceso verso una porta disabilitata, è raro che venga utilizzato, ma è comunque protagonista di alcune tecniche di attacco informatico oppure, come vedremo, di port scanning.

7. SYN – se attivo indica che l'host mittente vuole aprire una connessione TCP con il destinatario, verrà inoltre specificato il valore dell'ISN nel campo sequence number. L'host origine dovrà quindi attendere un segmento SYN/ACK.
8. FIN - se attivo indica che si desidera chiudere la connessione in quel momento aperta con l'host destinatario. Quest' ultimo dovrà quindi rispondere con un segmento ACK per poi chiudere la connessione e inviare un FIN. Nel momento in cui l'host origine ricevere il FIN incomincia un'attesa temporizzata e risponderà con un segmento ACK, anche nel caso ne riceva più di uno a causa di un rinvio. La connessione sarà completamente chiusa quando verrà ricevuto anche il segmento ACK.

- Windows size (16 bit): indica il numero di byte che il mittente è in grado di accettare a partire da quello specificato dall'acknowledgment number.
- Checksum (16 bit): un campo di controllo per verificare l'integrità del segmento.
- Urgent point (16 bit): rappresenta l'offset, a partire dal numero di sequenza, per indicare l'ultimo byte dei dati urgenti.
- Option: opzioni facoltative.
- Data: rappresenta il payload da trasmettere.

2.2. Nozioni su NMap e gli IDS

Un altro punto fondamentale per cui ritengo approfondire e provvedere ad un' esposizione, riguarda il port scanner NMap [5] e le sue caratteristiche e feature principali. Introduciamo già in questo punto della trattazione questo software poiché potremmo farne riferimento, ben prima dell'approfondimento che sarà sviluppato in un capitolo ben preciso, questo perché risulterà utile al lettore un confronto concreto tra alcuni concetti puramente teorici e i corrispettivi funzionamenti pratici con il sopraccitato port scanner.

NMap è un software open source per la network exploration e il security auditing ed è in grado di effettuare scansioni rapidamente sia di reti di grandi dimensioni che di singoli host. Tutto ciò è possibile anche grazie all'utilizzo di raw socket le quali, diversamente da quelle standard, permettono di modificare le normali caratteristiche dell'header di un pacchetto, di fatto non seguendo più le formattazioni standard stabilite dai protocolli di trasporto. Esse sono disponibili nativamente in tutti i moderni sistemi operativi anche se dalla versione XP SP2 di Windows in poi, sono presenti alcune restrizioni mentre nei sistemi UNIX sono sufficienti i privilegi d'amministratore per una gestione completa. Il port scanning non è l'unica tecnica possibile con questo software, tramite apposite opzioni per esempio è possibile effettuare l'host discovery, esso consiste nella ricerca di tutti quei host attivi o interessanti all'interno di una rete. Il termine interessante può variare in base alle proprie esigenze, per esempio può essere necessario solo la ricerca per identificare gli host sulla propria rete, oppure cercare quali hanno una specifica applicazione in esecuzione, in pratica le necessità sono tante e differenti e per questo motivo le opzioni di host discovery sono molteplici. NMap permette inoltre di scegliere come venga effettuata questa feature, un esempio può essere l'invio di pacchetti TCP SYN su determinate porte in attesa di segmenti RST o SYN/ACK i quali indicano che l'host è disponibile e attivo, oppure semplicemente viene effettuato un ping sugli indirizzi IP specificati controllando quali di essi rispondano. Risulta di notevole interesse la capacità di NMap di individuare con buona precisione, quale servizio possa essere in ascolto su una specifica porta nota, per far ciò utilizza un database di oltre 2200 servizi noti. La reale attenzione però va posta sul servizio di service

detection, in pratica vengono interrogate nuovamente le porte aperte per scovare quante più informazioni possibili. Grazie all'interpretazione delle risposte ricevute è possibile determinare l'applicazione, la versione, l'hostname, il tipo di device, la famiglia di sistema operativo ed altri dettagli. Raramente tutte queste informazioni vengono rilasciate così facilmente, e saranno quindi necessari alcuni stratagemmi che possono essere anche personalizzati attraverso diverse opzioni. La conoscenza accurata della versione di un determinato software è basilare per poter identificare quali vulnerabilità poter sfruttare e di conseguenza permette una gestione più efficace dell'intera attività di attacco. Una delle caratteristiche più famose di NMap consiste nell'OS detection. Grazie all'analisi del TCP/IP stack fingerprinting, cioè una raccolta passiva di tutta una serie di attributi di configurazione ottenuti da un device remoto durante le comunicazioni di rete, è possibile identificare quale sistema operativo giri sull'host. Viene utilizzato un database di oltre 1500 "OS fingerprints" conosciuti i quali verranno confrontati con i risultati ottenuti da decine di test sui bit di risposta ad una ben precisa serie di pacchetti TCP e UDP.

Abbiamo visto come NMap sia in grado attraverso tutta una serie di opzioni e tecniche, di rilevare debolezze e di fornire importanti informazioni attraverso i quali è possibile poi effettuare un attacco. Per difenderci da queste intrusioni possiamo fornirci di un firewall oppure di un Intrusion Detection System (IDS). Esso è un device hardware od un'applicazione software impiegato per il monitoraggio di reti o di sistemi e nell'identificazione di attività malevoli. Inoltre possono essere suddivisi tra quelli network based (NIDS) e host based (HIDS), i primi vengono situati in punti strategici in modo che possa sniffare il traffico sul segmento di rete attestato, da e verso gli host, nel modo più efficace possibile. I dati così ottenuti vengono analizzati e confrontati con un database di signature di attacchi (signature matching) costantemente aggiornato dal produttore dell'IDS, se questa strategia fallisce viene esaminato il flusso di traffico (network analysis) ed in base ad algoritmi o strategie implementate nell'NIDS, si rende così possibile l'identificazione di tutti quei attacchi non ancora memorizzati come tali nel database. Tra gli IDS network based più conosciuti sicuramente c'è Snort [6], il quale verrà analizzato e approfondito più avanti. Per quanto riguarda gli IDS host based, essi

analizzano tutta una serie di componenti dell'host in questione e degli esempi possono essere i file di log del sistema, le system call e tutte quelle operazioni che modificano vari elementi del file system. In base ovviamente all'HIDS scelto, cambieranno sia gli elementi vagliati che la strategia applicata. Un esempio di HIDS è Aide [7]. Nonostante gli IDS abbiano funzioni simili ai firewall, in quanto sono dispositivi per la sicurezza della propria rete e per farlo sfruttano l'analisi del traffico, essi differiscono dal momento che i primi valutano una sospetta intrusione laddove si verifichi ed eventualmente la segnalano ma non possono prevenirle, compito del quale invece si occupa il firewall, il quale è in grado di bloccare o filtrare i pacchetti in input o in output ma d'altro canto non può segnalare gli attacchi in atto.

2.3 Nozioni su Honeypot e Honeynet

Per poter studiare le caratteristiche principali delle diverse tipologie di port scanning si è optato per l'utilizzo di honeypot, essi costituiscono un sistema costruito ad hoc per poter essere facilmente individuato ed attaccato da un potenziale intruso. Il termine stesso "honeypot", letteralmente "vasetto di miele", rende l'idea dell'utilizzo, infatti gli attaccanti dovranno essere attirati da risorse facilmente violabili ma in realtà prive di alcun valore o da servizi non integrati in un reale ambiente di produzione, così da veicolare le intrusioni lontane dal reale sistema da proteggere. Inoltre questo ambiente permette la registrazione e la memorizzazione di tutte le informazioni utili lasciate durante lo svolgimento dell'attacco, inoltre un grosso vantaggio sarà dato dal fatto che il traffico verso quest'ultimo sarà limitato e da considerarsi tutto potenzialmente ostile, azzerando nella quasi totalità i falsi positivi. Da questa considerazione è possibile evincere come il costo di implementazione di un honeypot sia basso, in quanto dovrà analizzare solamente pacchetti rivolti verso di esso. Un altro vantaggio importante viene dalla prerogativa che tale strumento venga posto come punto finale di una comunicazione, di conseguenza le operazioni saranno svolte in chiaro e quindi registrate in tale modo indipendentemente dalle tecniche di offuscamento potenzialmente adottate.
Riassumendo possiamo esporre gli obiettivi solitamente desiderabili da un honeypot:

- la raccolta di dati sulle azioni dell'attaccante in modo da poter acquisire il maggior numero di informazioni sugli strumenti e le strategie prescelte;
- il sistema può essere rilevato, attaccato e compromesso ma dovrà comunque rimanere assolutamente isolato nel caso venga a sua volta violato.

D'altro canto come ogni sistema, non è esente da svantaggi che sono comunque da tenere bene in considerazione. Il primo da evidenziare è il limitato campo visivo, gli unici eventi analizzabili sono quelli rivolti direttamente all'honeypot stesso, infatti nel caso ci sia un attacco diretto alla rete da proteggere, si viene completamente tagliati fuori e non è quindi possibile intervenire. Inoltre essendo vulnerabile per sua natura si rende necessario un mascheramento della propria identità, in mancanza di esso per l'attaccante diventerà palese l'utilizzo di tale strumento e conseguentemente dirotterà l'attacco verso altri obiettivi.

Nei successivi sottoparagrafi presenteremo diverse tipologie di classificazione con le quali è possibile suddividere gli honeypot, in particolare vedremo una distinzione secondo la tecnologia con la quale vengono implementati, secondo il loro livello di interazione e per scopo.

2.3.1 Classificazione per tecnologia

- Honeypot standard: esso è composto da software e hardware, semplicemente con un qualsiasi server o pc.
- Honeytoken: esso non è legato all'hardware. Per esempio può essere un account mail, una chat o un sito web.
- Honeynet: è una rete di honeypot standard.

2.3.2 Classificazione per livello di interazione

Il livello di interazione rappresenta il grado di libertà di azione consentita all'attaccante e delle diverse modalità con cui può accedere al sistema. Maggiore è l'interazione e più alto sarà il livello di rischio e la probabilità di una compromissione ma si potranno raccogliere anche una maggior quantità di dati.

- Honeypot a bassa interazione: essi si limitano all'emulazione di servizi di rete e sistemi operativi con vulnerabilità note. Il rischio per le macchine ospitanti è ridotto in quanto eventuale codice malevolo non viene eseguito realmente, inoltre sono economici da un punto di vista di risorse necessarie al mantenimento e facili nell'implementazione e nella gestione di reti sia di piccole che di grandi dimensioni ma risultano anche facilmente identificabili per esempio a causa del traffico in uscita del tutto assente.
- Honeypot ad alta interazione: essi solitamente sono costituiti da computer o macchine virtuali che hanno lo scopo di mettere a disposizione servizi e sistemi reali che potranno essere attaccati e deliberatamente modificabili, da qui la necessità di disporre anche di meccanismi di ripristino. Il grosso vantaggio si ha nella mole di dati ottenibili a discapito di complessità e costi

nell'installazione e nel mantenimento più elevati rispetti ad honeypot a bassa interazione.

Per porre un confronto tra le opzioni di classificazione appena proposte è bene tener conto di vari aspetti come il calcolo dei costi necessari, la quantità e il valore dei dati ottenibili e i rischi da dover correre. In genere i sistemi a bassa interazione permettono un'installazione semplice e rapida, con un conseguente vantaggio anche dal punto di visto di tempo e di competenze richieste, inoltre poiché i servizi sono solamente emulati si otterranno con alta probabilità dati di valore inferiore a sistemi ad alta interazione che permettono la creazione di ambienti più realistici basandosi di fatto su un ambiente operativo vero e proprio. Proprio per quest'ultimo motivo è facile intuire come la complessità elevata per la manutenzione e l' installazione e la necessità di un monitoraggio costante porti a costi più elevati per l'implementazione. Inoltre a livelli di interazione maggiore proporzionalmente avremo rischi più elevati.

	Bassa interazione	Alta interazione
Tipo di interazione	Emulazione	Reale/Virtualizzata
Installazione e manutenzione	Semplice	Complessa
Costi	Bassi	Elevati
Monitoraggio	Assente	Necessario
Acquisizione dati	Limitata	Elevata
Rischi	Bassi	Medio/Alti

Immagine 5. Tabella dei confronti

2.3.3 Classificazione per scopo

- Production honeypot: lo scopo di tali sistemi è l'aumento della sicurezza di una rete, come per esempio una aziendale, e solitamente sono honeypot a bassa interazione, semplici ed economici senza troppe pretese per la raccolta dati.
- Research honeypot: lo scopo di tali sistemi è la ricerca, in particolar modo delle metodologie, degli strumenti e delle motivazioni utilizzate dall'intruso. Solitamente vengono sfruttati da enti di ricerca, governative o militari, ed in gran parte sono honeypot ad alta interazione.

2.3.4 Honeyd

Honeyd è un esempio di honeypot a bassa/media interazione di cui faremo uso, ed è un progetto open source sviluppato da Niel Provos dell'Università del Michigan le cui caratteristiche principali sono:

- emulazione di un elevato numero di host con la possibilità di implementare anche reti di grandi dimensioni (di fatto delle vere e proprie Honeynet);
- alta configurabilità;
- supporto nell'emulazione di più router per la gestione di più sottoreti;
- integrazione tra reti fisiche e virtuali.

Questo strumento permette di emulare il comportamento di tutta una serie di sistemi operativi, di router, access point, ecc., offrendo inoltre la possibilità di assegnare ad una certa porta un servizio noto basato sul protocollo TCP/IP, anch'esso emulato, oppure di eseguire un qual si voglia script. Grazie alla sua versatilità non solo, come già ripetuto, possiamo emulare una completa infrastruttura di rete in un'unica macchina fisica, ma abbiamo la possibilità di settare diversi parametri come larghezza di banda, numero di hop, latenza ecc..
Per quanto riguarda l'architettura di Honeyd possiamo suddividerla in 5 diversi componenti:

- Packet dispatcher
- Database di configurazione
- Personalità engine
- Gestione di protocolli
- Componente di routing opzionale

Il primo di questa serie viene interpellato nel momento in cui arrivi un pacchetto ad uno degli indirizzi IP assegnati all'honeypot, tale modulo ha quindi il compito di controllare la lunghezza dell'indirizzo IP e l'integrità del pacchetto stesso attraverso il relativo checksum. Vengono supportati i protocolli TCP, UDP e ICMP, se non conformi ad essi il pacchetto sarà scartato. I segmenti TCP o i datagrammi utente UDP quando vengono ricevuti saranno inviati a moduli, differenti ma con medesima funzione nel caso si riceva un messaggio ICMP, aventi il compito di interfacciarsi con gli script che emulano il relativo servizio. Il passo successivo consiste nell'interrogare il database di configurazione il quale mantiene in memoria tutti i file di configurazione delle macchine simulate dall'honeypot. A seconda dell'indirizzo IP di destinazione, tale pacchetto sarò indirizzato verso il modello della macchina corrispondente, se assente ne verrà assegnato uno di default. Questo passaggio è necessario in quanto senza di esso il processo di personalizzazione dei pacchetti di risposta da parte del personality engine non sarebbe possibile. Questo modulo infatti apporta modifica al contenuto dell'header del pacchetto grazie al quale risulterà compatibile con lo stack di rete utilizzato dal sistema operativo simulato. Honeyd attraverso questa peculiarità riesce ad ingannare l'analisi dell'OS fingerprinting svolto da strumenti con Nmap e Xprove [8], così che anche quest'ultimi siano ingannati nel caso svolgano questo tipo di identificazione. Chiamiamo in causa questi due software proprio perché Honeyd sfrutta il fingerprinting database di Nmap come riferimento per le personalità TCP e quello di Xprobe per quelle ICMP.
In questo capitolo abbiamo voluto dare solo un'idea di insieme delle caratteristiche di questo strumento senza volerci addentrare nei dettagli di implementazione che saranno però ovviamente esposti nei prossimi capitoli.

3.Implementazione di Honeyd

3.1 Configurazione

In questo paragrafo vogliamo descrivere i punti di maggior rilievo nell'implementazione del sistema di honeypot attraverso il software Honeyd. Per avere a disposizione tutti gli strumenti necessari in modo semplice e veloce abbiamo sfruttato una distribuzione open source di Linux chiamata HoneyDrive [9], la quale viene distribuita sotto forma di OVA[4] (Open Virtualization Appliance) e si basa su Xubuntu Desktop 12.0.4.4. All'interno sono stati installati più di 10 software honeypot, opportunamente configurati e pronti all'utilizzzo. Non è necessaria alcuna installazione, dovremo solo avere a disposizione un programma per la virtualizzazione, noi utilizzeremo VWware Workstation 10 [10], ed importare il file OVA ed avremo così facilmente accesso a tutti i software necessari per le nostre analisi e la nostra implementazione.

L' immagine 5 qui di seguito è la rappresentazione grafica della rete che si desidera implementare attraverso gli honeypot e con l'ausilio di un'adeguata configurazione del tool HoneyD.

[4] Per definire cosa sia un OVA, è bene prima spiegare e dare una semplice definizione di OVF (Oper Virtual Format). Esso consiste in uno standard aperto per la creazione e il rilascio di software eseguibile su macchine virtuali ed è composto da un descrittore, un file in XML che contiene i metadati, uno o più file immagine ed altri file opzionali. L'intera directory contente tutti i suddetti dati può essere più comodamente distribuita attraverso un archivio OVA.

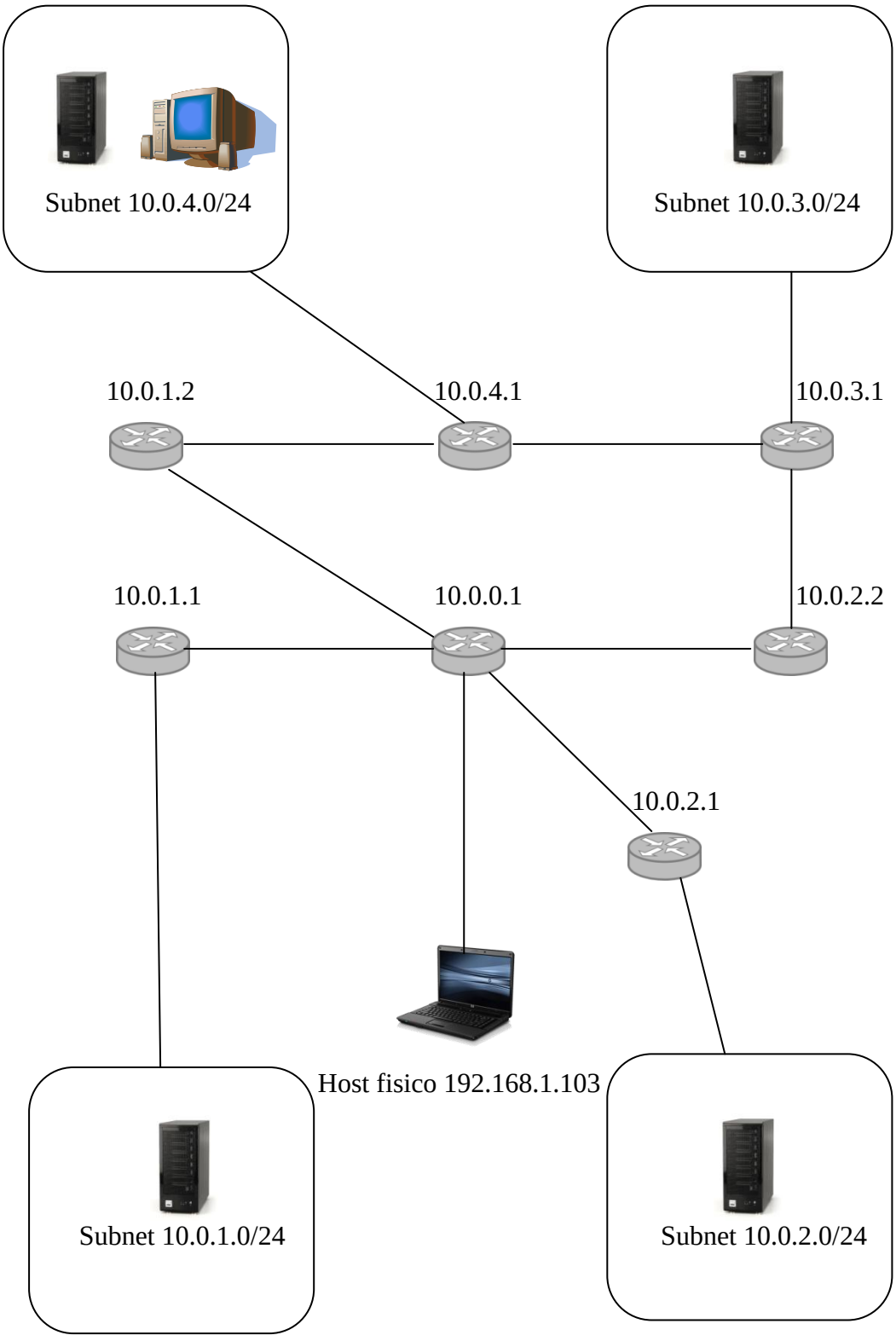

Subnet 10.0.4.0/24
Subnet 10.0.3.0/24
10.0.1.2
10.0.4.1
10.0.3.1
10.0.1.1
10.0.0.1
10.0.2.2
10.0.2.1
Host fisico 192.168.1.103
Subnet 10.0.1.0/24
Subnet 10.0.2.0/24

Immagine 6.

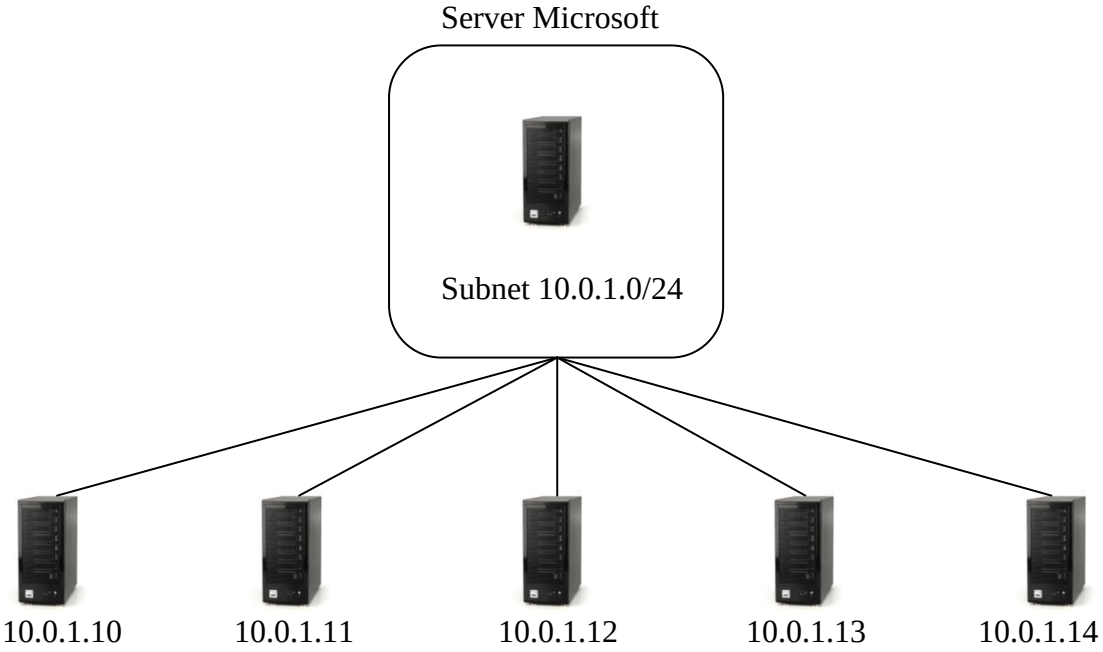


Immagine 7.

Questa subnet è composta da 5 server basati su Windows. I dettagli della loro configurazione e dei servizi attivi su essi e dei successivi elementi delle sottoreti che vedremo, verranno esposti nei prossimi sottoparagrafi.

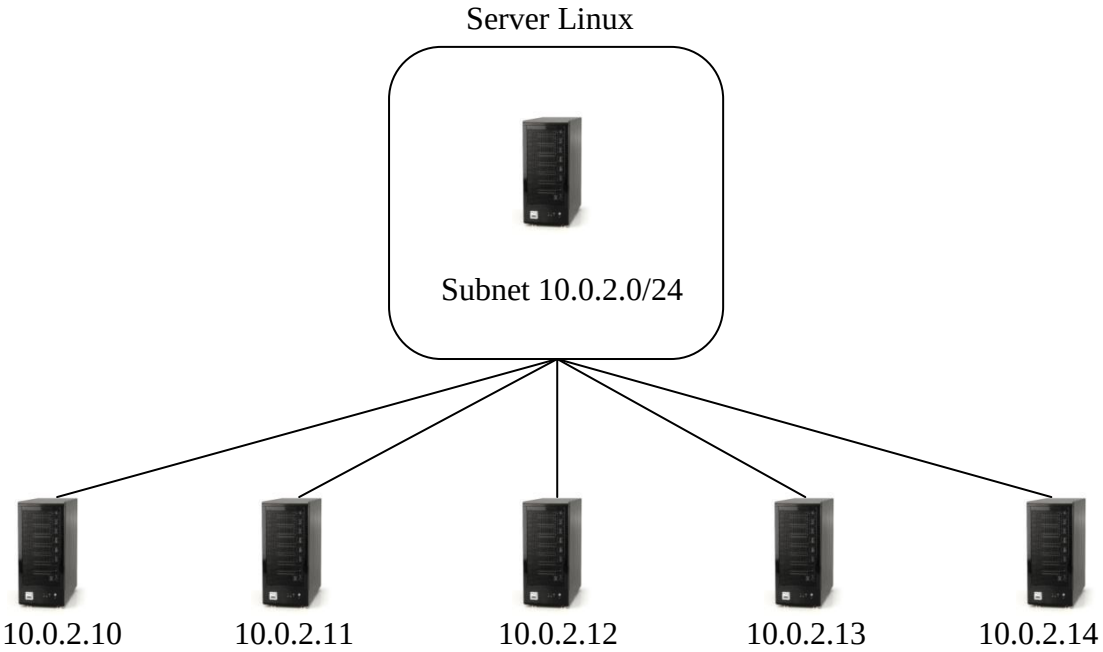


Immagine 8.

La composizione di questa subnet è costituita da 5 server basati su Linux.

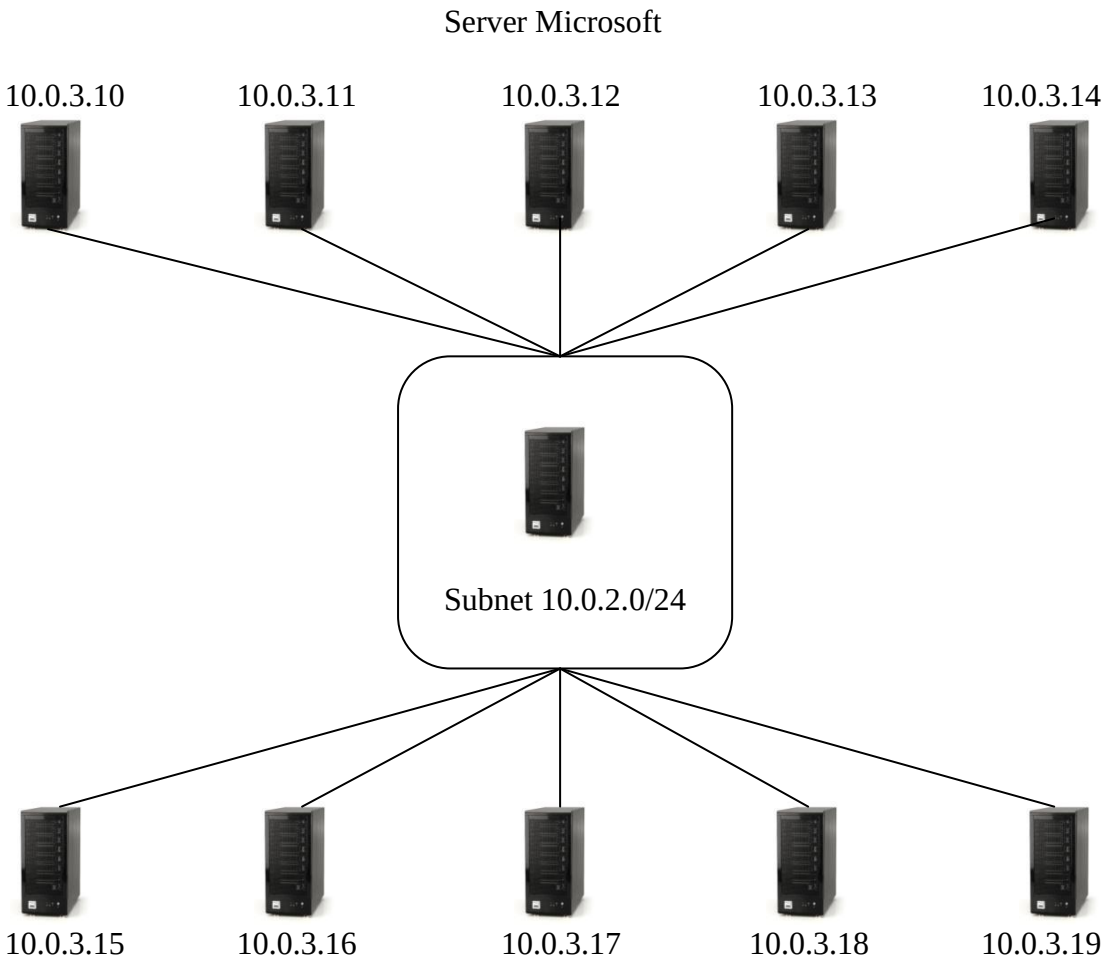


Immagine9.

Come mostrato in figura, abbiamo 10 elementi che costituiscono questa sottorete. I primi 5, cioè dall'indirizzo IP 10.0.3.10 al 10.0.3.14 sono costituiti da server Microsoft mentre i successivi da server Linux che definiremo come web, in quanto avranno solamente servizi attivi in un reale server web.

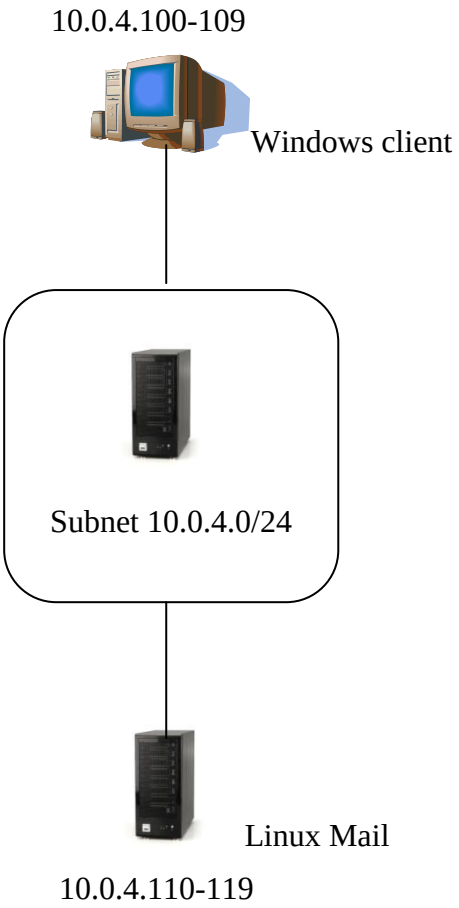


Immagine 10.

In questa sottorete avremo invece un totale di 20 host. I primi 10, come mostrato in figura quindi 10.0.4.100 a 10.0.4.109 saranno configurati come Windows client, mentre i restanti elementi sono implementati come server Mail Linux.

Riassumiamo quindi in una comoda tabella tutti gli elementi della rete, i relativi servizi e le personalità simulate (per una spiegazione di cosa si intende per personalità rimandiamo il lettore a pagina 38). Inoltre verrà associato ad ogni servizio, lo script relativo attraverso il quale l'honeypot potrà simularlo. Quelli proposti sono stati forniti direttamente da Honeyd ma è possibile collegare qualsiasi script.

Apparato	Personalità	Servizi e porta	Implementazione
Router	Cisco 1601R router running IOS 12.1(5)"	Telnet: 23	router-telnet.pl
Server Microsoft	Microsoft Windows NT 4.0 SP3	FTP: 21 HTTP: 80	ftp.sh iisemul8.pl
Client Microsoft	Microsoft Windows XP Professional SP1	FTP: 21 SMTP: 25 HTTP: 80 Pop3: 110 IMAP: 143 LDAP: 389 VNC: 5901 SNMP: 161	msftp.sh exchange-smtp.sh iis.sh exchange-pop3.sh exchange-imap.sh ldap.sh vnc.sh fake-snmp.pl
Server Linux	Linux 2.4.20	FTP: 21 SSH: 22 Telnet: 23 SMTP: 25 Finger: 79 HTTP: 80 POP3: 110 IMAP: 143	proftpd.sh ssh.sh telnetd.sh sendmail.sh fingerd.sh apache.sh qpop.sh cyrus-imapd.sh

		IPP: 515 Squid: 3128 Portmap: 111 SNMP: 161 SysLog : 514	lpd.sh squid.sh bportmapd.pl fake-snmp.pl syslogd.sh
Server Web Linux	Linux 2.4.20	FTP: 21 HTTP: 80 SNMP: 161	proftpd.sh apache.sh fake-snmp.pl
Linux Mail	Linux 2.4.20	POP3: 110 IMAP: 143 SNMP: 161	qpop.sh cyrus-imapd.sh fake-snmp.pl

Immagine 11.

Forniamo quindi una breve descrizione dei servizi che gli honeypot configurati dovranno simulare:

- FTP (File Tranfert Protocol): protocollo per la trasmissione di dati tra host basato su TCP.
- SSH (Secure SHell): protocollo di rete per la comunicazione e l'invio sicuro di dati. Viene stabilito un canale crittografato, il quale connette un SSH client con un SSH server.
- Telnet: telnet è un protocollo di rete utilizzato per fornire una comunicazione bidirezionale e text-oriented, utilizzato solitamente per fornire all'utente sessioni di login remoto tra host.
- SMTP (Simple Mail Trasnfer): protocollo per la trasmissione di e-mail. Esso permette soltanto di inviare messaggi di posta, ma non di richiederli ad un server.
- Finger: Nei sistemi unix-like, finger è un semplice protocollo di rete per lo scambio di informazioni utente e di status human-oriented.
- HTTP (HyperText Tranfer Protocol): protocollo utilizzato come principale sistema per la trasmissione dati sul web.

- POP3 (Post Office Protocol): protocollo utilizzato da un client e-mail per richiedere mail da un server attraverso una connessione TCP.
- IMAP (Internet Message access Protocol): protocollo per il recupero e la memorizzazione di email, e viene utilizzato in alternativa a POP3.
- IPP (Internet Printing Protocol): protocollo di rete per la stampa in remoto, così come la gestione delle operazioni di stampa, della risoluzione, ecc..
- Squid-http: Squid è un popolare software open source con funzionalità di proxy, solitamente utilizzato con l'utilizzo del protocollo HTTP.
- Portmap: Il servizio Portmap è un daemon di assegnazione della porta dinamica per servizi RPC (Remote Procedure Call), cioè una comunicazione inter-process che permette ad un programma di attivare una subroutine o una proceduta su una macchina presente in un altro spazio di indirizzi (comunemente un altro computer su una rete condivisa).
- SNMP (Simple Network management Protocol): consente la configurazione, la gestione e il monitoring di apparati collegati in una rete, riguardo a tutti quegli aspetti che richiedono azioni di management.
- LDAP (Lightweight Directory Access Protocol): protocollo standard per l'interrogazione e la modifica dei servizi di directory.
- VNC (Virtual Network Computing): software per il controllo remoto e l'amministrazione del proprio host a distanza: Risulta necessario installare un server VNC sulla propria macchina, ed attraverso un client VCN è possibile ricevere un'immagine dello schermo ed inviare input di tastiera e mouse.
- SysLog (SYStem LOG): protocollo utilizzato per la trasmissione di informazioni di log.

Per la corretta implementazione della topologia della rete come appena descritta, è necessaria un'adatta configurazione del file .config di Honeyd attraverso il quale l'applicazione sarà in grado di gestire ogni caratteristica decisa della rete.

Mostriamo qui di seguito il file HoneydConfiguration.config

```
###Windows XP Client
create winclient
set winclient personality "Microsoft Windows XP Professional SP1"
add winclient tcp port 445 open
add winclient tcp port 139 open
add winclient udp port 138 open
add winclient tcp port 21 "sh /usr/share/honeyd/scripts/win32/win2k/msftp.sh"
add winclient udp port 137 open
add winclient tcp port 135 open
set winclient default tcp action reset
set winclient default udp action reset
set winclient default icmp action open

###Windows server
create windows
set windows personality "Microsoft Windows NT 4.0 SP3"
set windows default tcp action reset
set windows default udp action reset
set windows default icmp action open
add windows tcp port 21 "sh /usr/share/honeyd/scripts/win32/win2k/msftp.sh"
add windows tcp port 25 "sh /usr/share/honeyd/scripts/win32/win2k/exchange-smtp.sh"
add windows tcp port 80 "sh /usr/share/honeyd/scripts/win32/win2k/iis.sh $ipsrc"
add windows tcp port 110 "sh /usr/share/honeyd/scripts/win32/win2k/exchange-pop3.sh"
add windows tcp port 143 "sh /usr/share/honeyd/scripts/win32/win2k/exchange-imap.sh"
add windows tcp port 389 "sh /usr/share/honeyd/scripts/win32/win2k/ldap.sh"
add windows tcp port 5901 "sh /usr/share/honeyd/scripts/win32/win2k/vnc.sh"
add windows udp port 161 "perl /usr/share/honeyd/scripts/unix/general/snmp/fake-snmp.pl public private -config=/usr/share/honeyd/scripts/unix/general"
```

```
###Linux Mail
create linux_mail
set linux_mail personality "Linux 2.4.20"
set linux_mail default tcp action reset
set linux_mail default udp action block
set linux_mail default icmp action open
set linux_mail uptime 73921
add linux_mail tcp port 110 "sh /usr/share/honeyd/scripts/unix/linux/suse8.0/qpop.sh $ipsrc $sport $ipdst $dport"
add linux_mail tcp port 143 "sh /usr/share/honeyd/scripts/unix/linux/suse8.0/cyrus-imapd.sh $ipsrc $sport $ipdst $dport"
add linux_mail udp port 161 "perl /usr/share/honeyd/scripts/unix/general/snmp/fake-snmp.pl public private -config=/usr/share/honeyd/scripts/unix/general"

###Linux Web
create linux_web
set linux_web personality "Linux 2.4.20"
set linux_web default tcp action reset
set linux_web default udp action block
set linux_web uptime 13282
add linux_web tcp port 21 "sh/usr/share/honeyd/scripts/unix/linux/suse8.0/proftpd.sh $ipsrc $sport $ipdst $dport"
add linux_web tcp port 80 "sh /usr/share/honeyd/scripts/unix/linux/suse8.0/apache.sh $ipsrc $sport $ipdst $dport"
add linux_web udp port 161 "sh /usr/share/honeyd/scripts/unix/general/snmp/fake-snmp.pl $ipsrc $sport $ipdst $dport"

###Linux Server
create linux_dev
set linux_dev personality "Linux 2.4.20"
set linux_dev default tcp action reset
set linux_dev default udp action reset
set linux_dev default icmp action open
```

```
set linux_dev uptime 8324
add linux_dev tcp port 21 "sh /usr/share/honeyd/scripts/unix/linux/suse8.0/proftpd.sh $ipsrc $sport $ipdst"
add linux_dev tcp port 22 "sh /usr/share/honeyd/scripts/unix/linux/suse8.0/ssh.sh $ipsrc $sport $ipdst $dport"
add linux_dev tcp port 23 "sh /usr/share/honeyd/scripts/unix/linux/suse8.0/telnetd.sh $ipsrc $sport $ipdst $dport"
add linux_dev tcp port 25 "sh /usr/share/honeyd/scripts/unix/linux/suse8.0/sendmail.sh $ipsrc $sport $ipdst $dport"
add linux_dev tcp port 79 "sh /usr/share/honeyd/scripts/unix/linux/suse8.0/fingerd.sh $ipsrc $sport $ipdst $dport"
add linux_dev tcp port 80 "sh /usr/share/honeyd/scripts/unix/linux/suse8.0/apache.sh $ipsrc $sport $ipdst $dport"
add linux_dev tcp port 110 "sh /usr/share/honeyd/scripts/unix/linux/suse8.0/qpop.sh $ipsrc $sport $ipdst $dport"
add linux_dev tcp port 143 "sh /usr/share/honeyd/scripts/unix/linux/suse8.0/cyrus-imapd.sh $ipsrc $sport $ipdst $dport"
add linux_dev tcp port 515 "sh /usr/share/honeyd/scripts/unix/linux/suse8.0/lpd.sh $ipsrc $sport $ipdst $dport"
add linux_dev tcp port 3128 "sh /usr/share/honeyd/scripts/unix/linux/suse8.0/squid.sh $ipsrc $sport $ipdst $dport"
add linux_dev udp port 111 "perl /usr/share/honeyd/scripts/unix/general/rpc/bportmapd.pl -proto udp -host usr/share/honeyd/scripts/unix/general/rpc/hosts/debian -srcip $ipsrc -dstip $ipdst -srcport $srcport -dstport $dport -logfile /var/log/honeyd -logall"
add linux_dev udp port 161 "perl /usr/share/honeyd/scripts/unix/general/snmp/fake-snmp.pl public private -config=/usr/share/honeyd/scripts/unix/general"
add linux_dev udp port 514 "sh usr/share/honeyd/scripts/unix/linux/suse8.0/syslogd.sh $ipsrc $sport $ipdst $dport"
```

```
###Reazioni di default
create default
set default default tcp action block
set default udp action block
set default default icmp action block
### Cisco router
create router
set router personality "Cisco 1601R router running IOS 12.1(5)"
set router default tcp action reset
set router default udp action reset
add router tcp port 161 "perl /usr/share/honeyd/scripts/snmp/fake-snmp.pl public
private -config=/usr/share/honeyd/scripts/unix/general"
add router tcp port 80 open
add router tcp port 23 "perl /usr/share/honeyd/scripts/router-telnet.pl"


###Server nella subnet 10.0.1.x
bind 10.0.1.10 windows
bind 10.0.1.11 windows
bind 10.0.1.12 windows
bind 10.0.1.13 windows
bind 10.0.1.14 windows

###Server nella subnet 10.0.2.x
bind 10.0.2.10 linux_dev
bind 10.0.2.11 linux_dev
bind 10.0.2.12 linux_dev
bind 10.0.2.13 linux_dev
bind 10.0.2.14 linux_dev


###Server nella subnet 10.0.3.x
bind 10.0.3.10 windows
bind 10.0.3.11 windows
```

bind 10.0.3.12 windows
bind 10.0.3.13 windows
bind 10.0.3.14 windows
bind 10.0.3.15 linux_web
bind 10.0.3.16 linux_web
bind 10.0.3.17 linux_web
bind 10.0.3.18 linux_web
bind 10.0.3.19 linux_web

##Client nella subnet 10.0.4.x
bind 10.0.4.100 winclient
bind 10.0.4.101 winclient
bind 10.0.4.102 winclient
bind 10.0.4.103 winclient
bind 10.0.4.104 winclient
bind 10.0.4.105 winclient
bind 10.0.4.106 winclient
bind 10.0.4.107 winclient
bind 10.0.4.108 winclient
bind 10.0.4.109 winclient
bind 10.0.4.110 linux_mail
bind 10.0.4.111 linux_mail
bind 10.0.4.112 linux_mail
bind 10.0.4.113 linux_mail
bind 10.0.4.114 linux_mail
bind 10.0.4.115 linux_mail
bind 10.0.4.116 linux_mail
bind 10.0.4.117 linux_mail
bind 10.0.4.118 linux_mail
bind 10.0.4.119 linux_mail

```
bind 10.0.1.1 router
bind 10.0.1.2 router
bind 10.0.2.1 router
bind 10.0.2.2 router
bind 10.0.3.1 router
bind 10.0.4.1 router

###entry point
route entry 10.0.0.1 network 10.0.0.0/21

##Impostazioni di routing
bind 10.0.0.1 router
route 10.0.0.1 link 10.0.0.0/24
route 10.0.0.1 add net 10.0.1.0/24 10.0.1.1
route 10.0.0.1 add net 10.0.2.0/24 10.0.2.1
route 10.0.0.1 link 10.0.1.1/32
route 10.0.1.1 link 10.0.1.0/24
route 10.0.2.1 link 10.0.2.0/24
route 10.0.1.2 add net 10.0.4.0/24 10.0.4.1
route 10.0.2.2 add net 10.0.3.0/24 10.0.3.1
route 10.0.3.1 link 10.0.3.0/24
route 10.0.4.1 link 10.0.4.0/24
route 10.0.0.1 add net 10.0.4.0/24 10.0.1.2
route 10.0.0.1 add net 10.0.3.0/24 10.0.2.2
bind 192.168.1.103 to eth0
```

Osservando il codice precedente, probabilmente gran parte dei comandi utilizzati risulterà sconosciuto ma nonostante siano comunque intuitivi, si preferisce fornire un'adeguata documentazione, e per un elenco anche dei comandi che non tratteremo rimandiamo il lettore a [11].

```
create windows
Sintassi: creation := "create" <template-name>|
```

Immagine 12.

Create è il primo comando che osserviamo nel codice e ha la funzione di creazione di un template, il quale sarà differente per ogni sistema da voler implementare, e permette la gestione dei servizi simulati ed eventualmente di legare tale sistema ad un certo indirizzo IP da noi esposto. Il nome del template è case-sensitive, deve risultare univoco e non può corrispondere ad una keyword, ma non solo infatti non può incominciare con un numero, né uno spazio vuoto e né un carattere ASCII esteso. Nel caso non siano rispettate tali regole Honeyd non sarà in grado di caricare correttamente la configurazione e di conseguenza verrà eseguito un messaggio di errore.

```
set windows personality " Microsoft Windows NT 4.0 SP3"
Sintassi: set ::= "set" <template name> "personality" <personality name>
```

Immagine 13.

Con il comando di set è possibile cambiare la configurazione del template. In particolare possiamo scegliere la "personalità" da attribuire. Essa determina il comportamento dello stack di rete e può essere scelta da un certo numero di popolari sistemi operativi ricavati da un preciso file di Nmap chiamato nmap.assoc, e fornito direttamente durante l'installazione. Occorre prestare molta attenzione sulla stringa del corrispondente SO deciso poiché venendo effettuato un preciso confronto con il file delle personalità, una non totale corrispondenza porterebbe inevitabilmente ad incorrere in un errore.

set router default udp action reset
Sintassi: set ::= "set" <template-name> "default" <nome protocollo> "action" <tipo action>

Immagine 14.

Il parametro action determina come l'host reagisce di default nel momento in cui riceva un pacchetto da un preciso protocollo. Le opzioni possono essere block, reset, od open.

- Open: specifica che tutte le porte sono aperte di default. Questo settaggio ha effetto solo sui pachetti UDP e TCP.
- Block: specifica che tutti i pacchetti di uno determinato protocollo vengano scartati di default. L'honeypot di conseguenza non risponderà a nessun pacchetto per quel protocollo e può risultare utile nel caso si voglia simulare il comportamento di un firewall.
- Reset: indica che tutte le porta sono chiuse di default. Se una porta TCP risultasse chiusa, l'honeypot risponderà con un TCP RST al segmento SYN ricevuto, mentre nel caso di una porta UDP, ci sarà in risposta un messaggio ICMP di porta irraggiungibile.

add windows tcp port 21 "sh /usr/share/honeyd/scripts/win32/win2k/msftp.sh"
Sintassi: addition = "add" <template-name> <protocol name> "port" <port-number> <name service> (ultimo parametro opzionale)

Immagine 15.

Il comando add richiede di specificare il nome del template da configurare, un protocollo, un numero di porta e un file che esegua un dato servizio. Nell'esempio proposto nell'immagine qui sopra, è possibile osservare che nel nostro caso il protocollo scelto sarà TCP, il numero di porta invece 21 ed infine avremo lo script da eseguire con il relativo path alla directory in cui è salvato. Quando un host remoto

tenterà di stabilire una connessione TCP su porta 21, Honeyd incomincerà un nuovo processo che eseguirà lo script indicato, simulando di fatto tale servizio.

```
bind 10.0.1.10 windows
bind 192.168.1.103 to eth0
Sintassi: binding = "bind" < ip-address> < template-name "
```

Immagine 16.

Questo comando viene utilizzato per assegnare ad un template un determinato indirizzo IP. Se un pacchetto arriva ad un indirizzo IP al quale non è stato assegnato alcun template, Honeyd ne utilizza uno di default. Nel caso in cui il sistema riceva un pacchetto con un determinato indirizzo IP di destinazione, Honeyd consulta la configurazione per determinare se la porta risulti aperta ed eventualmente quale servizio sia da attivare quando la connessione verrà stabilita. Il momento in cui avviene l'associazione tra template ed indirizzo IP, determina inoltre di fatto la creazione di un nuovo template con nome corrispondente all'indirizzo IP e una copia della completa configurazione dell'originale. Questo meccanismo porta ad un grosso vantaggio: è possibile utilizzare tutta una serie di comandi di configurazione senza che tali cambiamenti si possano ripercuotere sul template di origine. Inoltre con lo stesso comando è possibile integrare al sistema virtuale, un host fisico attraverso una determinata interfaccia, nel nostro caso, come mostrato nella figura 16., quella ethernet. Sarà proprio su questa macchina, l'origine dalla quale faremo partire i diversi comandi e applicazioni utilizzate nei test, di cui parleremo a breve, e per le analisi che invece presenteremo nel prossimo ed ultimo capitolo.

Per creare la topologia del routing della nostra rete viruale con honeyd, abbiamo a disposizione diversi comandi, il primo di questi che andremo ad esporre è route entry.

```
route entry 10.0.0.1 network 10.0.0.0/21
Sintassi: route = "route" "entry" <ip-address> "network" <ip network>
```

Immagine 17.

Attraverso il file di configurazione ci verrà permesso di specificare l'indirizzo IP del primo dispositivo dalla quale partirà il routing. Solitamente la topologia virtuale di routing viene rappresentata come un albero. Il nodo entry rappresenta la radice mentre ogni router successivo viene presentato come nodo interno, gli archi che li uniscono non sono altro che i collegamenti tra i router stessi, ognuno dei quali avrà caratteristiche di latenza e perdita di pacchetti propri. I nodi foglia corrispondono alle sottoreti. Honeyd supporta multipli entry point ma è una funzione di cui non faremo uso.

```
route 10.0.0.1 link 10.0.0.0/24
Sintassi: route = "route" <ip-address> "link" <ip network>
```

Immagine 18.

Il comando link permette individualmente ad ogni router di essere configurato in modo da poter raggiungere una specifica rete.

```
route 10.0.0.1 add net 10.0.1.0/24 10.0.1.1
Sintassi: route = "route" <ip-address> "add" "net" <ip network> <ip-address>
```

Immagine 19.

Il comando add net, setta la rete per la quale il router è il responsabile

Honeyd inoltre ci mette a disposizione tutta una serie di variabili le quali potranno risultare utili nell'esecuzione di determinati script:

- IPSRC corrisponde all'indirizzo IP del mittente.

- IPDST corrisponde all'indirizzo IP di destinazione.
- SPORT corrisponde al numero di porta del mittente.
- DPORT corrisponde al numero di porta del destinatario.
- TYPE corrisponde al tipo di protocollo utilizzato (UDP, TCP o ICMP).

Tali variabili possono essere utilizzate per rendere gli script maggiormente dinamici. Per esempio, durante una comunicazione è possibile memorizzare l'indirizzo IP del mittente, per poi salvarlo in un record del log file oppure sfruttato se fosse necessario l'invio di dati in risposta.

Presentiamo inoltre una panoramica su tutti i possibili flag e i corrispondenti parametri, i quali ci permetteranno di indicare molti aspetti nell'esecuzione di Honeyd.

```
honeyd [-dP] [-l logfile] [-s servicelog] [-p fingerprints] [-0 p0f-file] [-x xprobe] [-a assoc] [-f file]  [-i interface] [-u uid] [-g gid] [-c host:port:username:password] [--webserver-port port] [--webserver-root path] [--rrdtool-path path] [--disable-webserver] [--disable-update] [--fix-webserver-permissions] [-V|--version] [-h|--help] [--include-dir] [net ...]
```

Immagine 20.

Di seguito, presentiamo in dettaglio il valore dei parametri principali e la relativa funzione:

- -f configfile: probabilmente il flag più importante. Indirizza Honeyd nell'individuare quale sia il file di configurazione da utilizzare. Questo file contiene informazioni su tutti gli honeypot virtuali, la loro topologia, e quali servizi presentano.
- -i interface: di default, Honeyd utilizza la prima interfaccia di rete a disposizione per ascoltare il traffico in arrivo. Nel caso in cui, siano presenti più interfacce è però bene specificare manualmente quale si desidera utilizzare.

- -d: Questi flag indica ad Honeyd di essere eseguito in modalità debug. Tutti i messaggi di stato saranno stampati a video nel terminale utilizzato. Può risultare molto comodo quando andiamo a testare i file di configurazione e per imparare i vari meccanismi di funzionamento di Honeyd. Una volta avuta la certezza che ogni settaggio ed ogni funzionalità sia svolta correttamente, questo parametro può essere omesso così che Honeyd possa essere eseguito in background.
- -l logfile: abilitando questa flag verrà memorizzato nella directory prescelta, il file di log contenente tutta una serie di informazioni riguardanti i pacchetti inviati e ricevuti. Ricordando inoltre che Honeyd deve necessariamente avere i diritti di scrittura sulla directory e sul file. Questo flag è disattivato di default.
- -s service log: permette la scrittura di log di informazioni ottenuti dai servizi simulati. Di default non è attivo.
- -p fingerprints: indica il percorso dal data file in cui sono presenti i fingerprint di Nmap. Se si utilizza la cartella d'installazione di default di honeyd, questo file si trova in /usr/local/share/honeyd/nmap.print. In questo caso non è necessario indicare tale flag.
- -0 p0f-file: indica il percorso al passive fingerprinting database. Esso permette al programma di identificare il sistema operativo dell'host remoto. Se l'applicativo è stato installato correttamente, non si necessita di questo flag.
- -x xprob: indica il percorso al Xprobe fingerprint database. Esso permette ad Honeyd di restituire la corretta risposta ICMP verso i tool di ICMP fingerprinting.
- -a assoc: indica il percorso al database che associa il fingerprinting di Xprobe e quello di Nmap. Questo file è necessario per combinare i benefici di entrambi questi database.

3.2 Test

In questo paragrafo vogliamo testare, con un semplice ping e traceroute[5], se gli host della nostra rete virtuale siano attivi e se la topologia venga correttamente seguita.
Eseguiamo il primo test di ping sul dispositivo con indirizzo IP 10.0.1.1 e presentiamo gli screenshot dei risultati che abbiamo ottenuto:

```
honeydrive@honeydrive:~/Desktop$ sudo honeyd -d -f HoneydConfiguration.conf
Honeyd V1.5c Copyright (c) 2002-2007 Niels Provos
honeyd[4929]: started with -d -f HoneydConfiguration.conf
honeyd[4929]: listening promiscuously on eth0: (arp or ip proto 47 or (udp and s
rc port 67 and dst port 68) or (ip )) and not ether src 00:0c:29:63:5a:0e
honeyd[4929]: Running with root privileges.
honeyd[4929]: Sending ICMP Echo Reply: 10.0.1.1 -> 192.168.10.20
honeyd[4929]: Sending ICMP Echo Reply: 10.0.1.1 -> 192.168.10.20
honeyd[4929]: Sending ICMP Echo Reply: 10.0.1.1 -> 192.168.10.20
honeyd[4929]: Sending ICMP Echo Reply: 10.0.1.1 -> 192.168.10.20
```

Immagine 21.

Per prima cosa mandiamo in esecuzione honeyd ed utilizziamo solamente i flag –d (debug) e –f (per determinare il file di configurazione) per il semplice motivo che non necessitiamo di qualcosa di più complesso poiché dobbiamo solamente verificare il corretto funzionamento degli honeypot.

[5] Traceroute è un software in grado di ricavare gli indirizzi IP di ogni router attraversato per raggiungere il destinatario. Nell'ambiente Windows il comando per eseguirlo è tracert, mentre per Linux esiste il programma traceroute.

Immagine 22.

Come possiamo vedere attraverso la figura 22. l'host ci risponde normalmente e senza alcun problema.

Proviamo lo stesso procedimento ma questa volta sull'host 10.0.4.100.

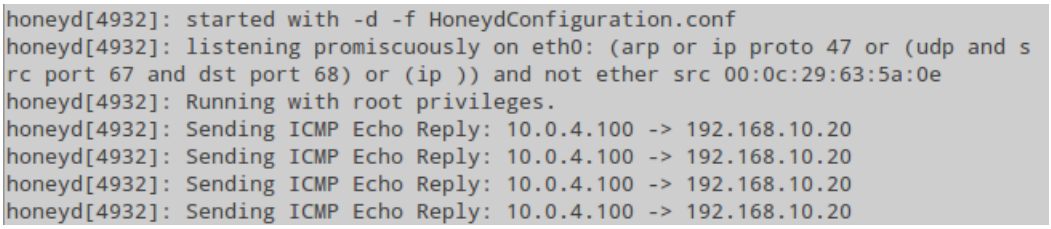


Immagine 23.

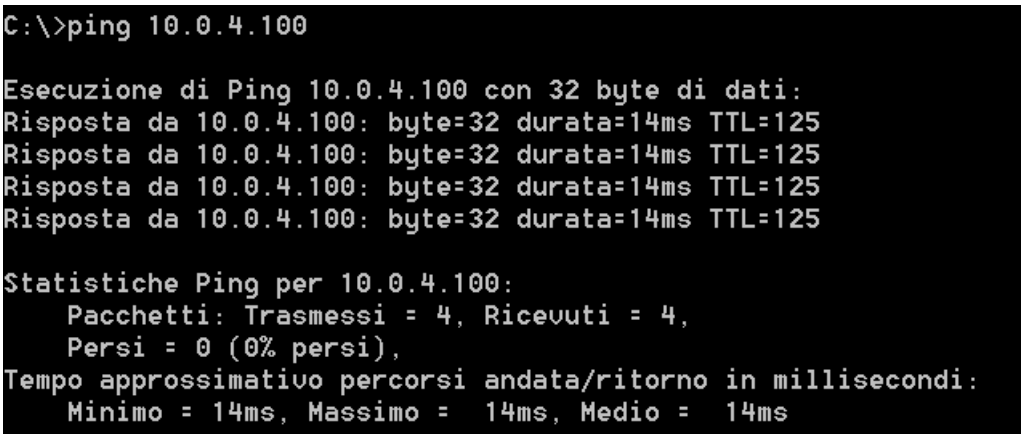


Immagine 24.

Anche in questo caso non risultano esserci problemi, e possiamo inoltre notare una piccola differenza in termini di tempo impiegato per il semplice motivo che tale host si trova topologicamente più lontano dal router di entry point.
Passiamo quindi all'applicazione traceroute, e la testiamo sempre sull'host 10.0.4.100, per controllare se la topologia della rete sia rispettata.

```
honeydrive@honeydrive:~/Desktop$ sudo honeyd -d -f HoneydConfiguration.conf
Honeyd V1.5c Copyright (c) 2002-2007 Niels Provos
honeyd[4965]: started with -d -f HoneydConfiguration.conf
honeyd[4965]: listening promiscuously on eth0: (arp or ip proto 47 or (udp and s
rc port 67 and dst port 68) or (ip )) and not ether src 00:0c:29:63:5a:0e
honeyd[4965]: Running with root privileges.
honeyd[4965]: Connection: udp (192.168.10.20:137 - 10.0.4.100:137)
honeyd[4965]: Connection established: udp (192.168.10.20:137 - 10.0.4.100:137)
honeyd[4965]: Connection to closed port: udp (192.168.10.20:137 - 10.0.0.1:137)
honeyd[4965]: Connection to closed port: udp (192.168.10.20:137 - 10.0.0.1:137)
honeyd[4965]: Connection to closed port: udp (192.168.10.20:137 - 10.0.0.1:137)
honeyd[4965]: Connection to closed port: udp (192.168.10.20:137 - 10.0.1.2:137)
honeyd[4965]: Connection to closed port: udp (192.168.10.20:137 - 10.0.1.2:137)
honeyd[4965]: Connection to closed port: udp (192.168.10.20:137 - 10.0.1.2:137)
honeyd[4965]: Connection to closed port: udp (192.168.10.20:137 - 10.0.4.1:137)
honeyd[4965]: Connection to closed port: udp (192.168.10.20:137 - 10.0.4.1:137)
honeyd[4965]: Connection to closed port: udp (192.168.10.20:137 - 10.0.4.1:137)
honeyd[4965]: Sending ICMP Echo Reply: 10.0.4.100 -> 192.168.10.20
honeyd[4965]: Sending ICMP Echo Reply: 10.0.4.100 -> 192.168.10.20
honeyd[4965]: Sending ICMP Echo Reply: 10.0.4.100 -> 192.168.10.20
```

Immagine 25.

```
C:\>tracert 10.0.4.100

Traccia instradamento verso 10.0.4.100 su un massimo di 30 punti di passaggio

  1    <1 ms    <1 ms    <1 ms  10.0.0.1
  2     7 ms     7 ms     6 ms  10.0.1.2
  3    15 ms    15 ms    13 ms  10.0.4.1
  4    15 ms    15 ms    13 ms  10.0.4.100

Traccia completata.
```

Immagine 26.

Come possiamo osservare i dispositivi che vengono individuati sono:

1. 10.0.0.1
2. 10.0.1.2
3. 10.0.4.1
4. 10.0.4.100

Se osserviamo figura 6, c'è un'effettiva corrispondenza tra i vari passaggi necessari per arrivare da punto di entrata all'host desiderato, quindi possiamo affermare che la nostra configurazione ci permette di ottenere in modo corretto la topologia desiderata.

4. Tecniche di port scanning

Si desidera descrivere in maniera approfondita, ogni tecnica di port scanning che utilizzeremo nei successivi test, i quali verranno inoltre analizzati e confrontati rispetto a tutta una serie di parametri, i cui dettagli saranno poi forniti nel prossimo capitolo. Per poter fornire queste informazioni sono stati consultati diversi articoli scientifici o libri [12], [13], [14], [15], [16], [17] e che consigliamo al lettore qualora volesse avvalersi per ulteriori chiarimenti.

Buona parte delle tecniche più comuni tentano di colpire porte TCP, poiché quest'ultime utilizzano un protocollo connection-oriented, per la quale si ottiene un feedback utile all'attaccante. Questa tipologia di attacchi viene definita come TCP scanning e si basano sull'instaurazione di una connessione TCP, anche se in pochi casi verrà effettivamente completamente stabilita. Alcuni tipi di TCP scan sono: TCP connect, SYN, FIN, ACK, XMAS, NULL. Una precisazione doverosa consiste nello specificare cosa si intende per scansione stealth, essa semplicemente indica quanto essa sia progettata per non essere individuata dagli strumenti di controllo come gli IDS.

Vediamo quindi nel dettaglio le tecniche poco sopra citate, partiamo proprio da queste poiché sono tra le più utilizzate.

4.1 TCP Scanning

A. TCP Connect Scan

La tecnica chiamata Connect Scan è sicuramente tra le più semplici e banali da implementare, tuttavia è anche molto semplice da rilevare in quanto è sufficiente memorizzare un log delle connessioni ed osservare se da uno stesso mittente, in un intervallo temporale ristretto, ci siano stati tentativi di instaurare su porte differenti un gran numero di connessioni. L'instaurazione di una connessione con una determinata porta indica che essa è aperta, una volta identificata dall'attaccante, la connessione viene chiusa. A causa del gran numero di pacchetti inviati il metodo risulta molto lento e per questo più che altro viene utilizzato su un numero molto ristretto di porte.

Il comando di Nmap da utilizzare per effettuare tale tecnica è: -sT.

1. Velocità: Il TCP Connect scanning è molto lento.
2. Stealth: Il TCP Connect scanning è facilmente individuabile e richiede l'invidio di un gran numero di pacchetti.
3. Porte aperte: Si rileva una porta aperta, nel caso sia possibile completare l'handshake a 3 vie.
4. Porte chiuse: Si rileva una porta chiusa, nel caso non sia possibile completare l'operazione di handshaking.
5. Porte filtrate: Non è possibile distinguere una porta chiusa da una porta filtrata.
6. Porte non filtrate: È possibile rilevare una porta non filtrata solo nel caso essa corrisponda ad un servizio TCP/IP attivo.

B. SYN Scan

Il SYN scan è tra le tecniche di port scanning più utilizzate, questo grazie alla velocità infatti è possibile scansionare migliaia di porte per secondo. L'attaccante quindi controlla ogni singola porta inviando un pacchetto SYN, quelle aperte risponderanno con un SYN|ACK, mentre quelle chiuse con un RST, di conseguenza non verrà completato l'handshake a 3-vie necessaria per stabilire una connessione TCP, ma bensì essa cadrà nel momento in cui la vittima invierà la risposta. Mentre nel caso non si riceva alcun riscontro, o si riceva un messaggio ICMP di tipo 3 (Porta non raggiungibile), ciò implica che la porta sarà probabilmente filtrata da un packet filter. Questa tecnica, come molte altre di cui parleremo a breve, necessitano di pacchetti formati in modo particolare e diversificato in base alla situazione e all'esigenza dell'attacco. A livello di velocità il SYN scan risulta notevolmente più rapido del Connect Scan, in quanto in questa tecnica è assente l'overhead dato da una instaurazione completa della connessione, ogni qual volta si scansiona una porta. Una problematica è la facilità con cui può essere identificata, poiché gran parte degli IDS, oramai ci riescono senza troppa difficoltà, infatti per esempio è sufficiente notare un gran numero di pacchetti con il flag di SYN settato, provenienti da un singolo host.
Il comando di Nmap da utilizzare per effettuare tale tecnica è: -sS.

1. Velocità: Il TCP SYN scanning è veloce in comparazione con altre tecniche di port scanning.
2. Stealth: Il TCP SYN scanning è stealth e la sua rilevazione causa molti falsi positivi.
3. Porte aperte: Si rileva una porta aperta, nel caso si riceva una risposta SYN|ACK.
4. Porte chiuse: Si rileva una porta aperta, nel caso si riceva una risposta RST.
5. Porte filtrate: Una non risposta, o un messaggio ICMP indica che la porta viene filtrata.
6. Porte non filtrate: Non è possibile distinguere una porta non filtrata.

C. FIN, Xmas e Null Scan

Queste tre tipologie di scansione sfruttano una piccola vulnerabilità nell' RFC 793 [3] per distinguere tra le porte open e closed. Quando si scansionano sistemi aderenti a questo testo RFC, qualunque pacchetto che non contenga i bit di SYN, RST, o ACK causerà un RST di ritorno se la porta risulterà chiusa mentre non si riceverà alcun riscontro se la porta è aperta. Nel caso del FIN scan, viene settato solamente il flag di FIN, nel Xmas scan si settano il FIN, PSH e URG accendendo il pacchetto come un albero di natale (per questo motivo gli è stato assegnato il curioso nome Xmas), mentre nel Null scan non viene acceso alcun flag. Questi tre tipi di scan sono esattamente identici nel comportamento ad eccezione delle attivazioni dei tre bit nei pacchetti TCP usati per la verifica delle porte. Se viene ricevuto un pacchetto RST, la porta è considerata chiusa, mentre l'assenza di risposta indica che la porta è aperta. La porta è marcata come filtrata se viene ricevuto un pacchetto ICMP unreachable (tipo 3, codice 1, 2, 3, 9, 10, o 13). Il vantaggio sostanziale di questi tipi di scan è che possono passare inosservati in certi firewall o packet filter. In ogni caso non è corretto fare cieco affidamento su questo, gran parte dei moderni prodotti IDS possono essere configurati in modo da rilevarli. Il grande svantaggio è che non tutti i sistemi seguono alla lettera la RFC 793. Un buon numero di sistemi manda risposte RST ai pacchetti di controllo indipendentemente dal fatto che le porte siano aperte o chiuse. Questo causa il fatto che tutte le porte appaiano come chiuse. I più diffusi sistemi operativi che fanno questo sono Microsoft Windows, e molti apparati Cisco,

BSDI, o IBM OS/400. Questo scan funziona applicato alla maggior parte dei sistemi UNIX.
I comandi di Nmap da utilizzare per effettuare le tecniche di FIN, Xmax e Null scan sono rispettivamente –sF, -sX e –sN.

D. ACK Scan

Questo metodo, soprattutto in passato, veniva utilizzato per identificare quali porte venissero filtrate da un certo firewall, quest'ultimi infatti non sempre bloccavano i pacchetti con il flag di ACK acceso, e ciò permetteva quindi di differenziare le porte filtrate o meno. Sia nel caso di porte aperte che in quelle chiuse, la vittima risponde con un pacchetto RST, mentre se la porta risulta filtrata non si ha una risposta. Quando questa tecnica è combinata con quella di SYN Scan, un attaccante è in grado di ottenere un quadro più completo per quanto riguarda il rule-set del firewall. Se l'invio di pacchetti SYN sollecita la riposta di un SYN|ACK o di un RST, ed inoltre se l'indirizzamento di pacchetti ACK porta ad un RST di risposta, la porta risulta non filtrata, ma se in quest'ultima condizione non si riceve un riscontro, la porta verrà definita filtrata. Nel caso un pacchetto SYN non generi né una riposta di SYN|ACK né di RST, ma invece un ACK provoca come risposta un RST, allora la porta anche in questo caso risulta filtrata. Nell'ultimo caso, cioè che non si riceva alcun tipo di riposta, si determina che la porta è stata bloccata da una specifica condizione del firewall.
Il comando di Nmap da utilizzare per effettuare tale tecnica è: -sA.
Riassumendo possiamo dire che:

1. Velocità: risulta veloce se confrontato ad altri tipi di port scanning.
2. Stealth: ACK scanning è di tipo stealth.
3. Porte aperte: Non è possibile determinare se una porta è aperta.
4. Porte chiuse: Non è possibile determinare se una porta è chiusa.
5. Porte filtrate: Risulta possibile se combinato con lo SYN Scan.
6. Porte non filtrate: Risulta possibile se combinato con lo SYN Scan.

Abbiamo iniziato nell'esposizione proprio di queste tecniche perché le ritroviamo comodamente di default tra le varie opzioni attivabili di Nmap e saranno sicuramente tra quelle che analizzeremo, ovviamente però non sono le uniche tecniche su porte TCP che si possono ottenere. Per completezza vogliamo presentarne anche altre, come per esempio: Fragmented Packets Scanning, Decoy Scanning, Ident Scanning e Proxy Scanning. Tutte queste possono comunque essere ricavate tramite l'opzione di Nmap –scanflags, essa consente una scansione custom specificando arbitrariamente i flag TCP necessari. Per farlo bisogna semplicemente definire quelli desiderati tramite la combinazione delle abbreviazioni con la quale solitamente si definiscono appunto i suddetti flag, cioè URG, ACK, PSH, RST, SYN e FIN. Inoltre è possibile specificare il tipo di interpretazione delle risposte, per esempio se si scegliere un SYN Scan, una non risposta definisce una porta filtered, mentre per un FIN Scan lo stesso comportamento identifica una porta open.

E. Fragmented Packets Scanning:

L'anonimato per quanto riguarda le scansioni di tipo SYN, FIN, XMAX o NULL può essere implementata utilizzando piccoli frammenti dei datagrammi IP. In questo modo, anche l'intestazione TCP verrà suddivisa a sua volta, rendendo più difficile per i packet filter o per il sistema di rilevazione di intrusioni di identificare o prevenire questi attacchi.

F. Decoy Scanning:

SYN, FIN, XMAS e NULL scan, ed altre forme non ancora discusse, possono essere rese ulteriormente più anonime, dal punto di vista dell'attaccante, utilizzando intrusioni di tipo decoy ("esca"). Questa tecnica utilizza lo spoofing degli indirizzi, cioè la falsificazione del proprio IP, per cui insieme ai pacchetti di scansione veri e propri ne vengono inviati anche molti del tutto simili ma con un indirizzo mittente diverso dal proprio. Quando quest'ultimi raggiungono la destinazione, il destinatario non avrà modo di distinguere tra i pacchetti veri e quelli fittizi. L'indirizzo IP dell'attaccante sarà comunque visibile alla vittima ma per un'eventuale IDS o amministratore di rete sarà più difficile identificare quale, tra tutte le scansioni ricevute, sia quella vera e quindi risalire all'indirizzo IP che ha

effettuato la scansione. I programmi che implementano questa tecnica permettono di specificare una lista di indirizzi IP. Si consiglia di scegliere comunque indirizzi plausibili come ad esempio altri computer connessi alla stessa ora e di evitare invece indirizzi di reti di note corporazioni che difficilmente lanciano scansioni di questo tipo. Se gli host utilizzati come esche non sono raggiungibili, il bersaglio sarà colpito da una grande quantità di segmenti SYN, tali da causare un SYN flooding[6]. Naturalmente, se tutti gli host decoy sono irraggiungibili, risulta ovvio che l'unica macchina contattabile è il vero mittente dell'attacco, quindi la scelta degli IP risulta fondamentale.

G. Ident Scanning

Alcuni port scanner (come ad esempio Nmap), offrono funzioni aggiuntive che mirano all'individuazione di caratteristiche e comportamenti dei processi che sono in ascolto nelle porte analizzate. Sfruttando l'Identification Protocol, specificato nell'RFC 1413 [18], l'attaccante tenterà di instaurare una connessione completa, (necessaria per la buona riuscita della tecnica) su una delle porte TCP da violare, ad esempio utilizzando una TCP Connect Scan, se ciò avviene allora sarà inviato un ident request per l'identd (Identification Protocol daemon) sulla porta TCP 113 dell'host bersaglio, di fatto si sta richiedendo l'identità del servizio a cui si sta connettendo. Questa operazione agevola la scoperta di eventuali account presenti sul server che possono dunque essere attaccati. L'Ident scanning viene anche chiamato reverse ident scanning, poiché anche se nell'originale RFC questo protocollo risulti utile per aiutare i server ad autenticare i client, l'attuale implementazione permette invece un uso al contrario, cioè i client identificano il server. Ovviamente questo scan non funziona se nell'host non è in esecuzione identd. E' possibile abilitare questa opzione su Nmap tramite il comando –I.

[6] Per SYN flooding si intende un attacco nel quale un utente malevolo invia una serie di richieste SYN, le quali stabiliranno una gran quantità di connessioni "mezze aperte" che limiteranno il più possibile le risorse del sistema oggetto dell'attacco.

H. Proxy Scanning

Un'interessante caratteristica del protocollo FTP è il supporto per le cosi chiamate "proxy FTP connection". Ciò permette all'utente di connettersi al server FTP e richiedere che il file sia inviato ad un differente FTP server. Tale caratteristica si presta per varie tipologie di abuso, cosicché molti server hanno smesso di supportarla. Uno utilizzo malevolo di questa peculiarità è la possibilità di far effettuare al server FTP un port scan verso altri host. Basta semplicemente richiedere al server FTP di inviare un file ad ognuna delle porte che vogliamo scansionare. Il messaggio di errore ci permetterà di dedurre se la porta sia aperta o meno. Questo è un ottimo modo per aggirare i firewall in quanto gli FTP aziendali sono spesso posizionati nella rete cosi da poter accedere a più host interni di quanto sia possibile fare da Internet.
Questa debolezza permette inoltre la possibilità di implementare ulteriori attacchi, tra cui il FTP bounce attack. Questa vulnerabilità è stata diffusa nel 1997, ed è stata risolta su gran parte dei sistemi. Esistono alcuni server ancora vulnerabili, ed ha senso provare ad utilizzarla quando ogni altra cosa fallisce. Se l'obiettivo è oltrepassare un firewall, è necessario effettuare una scansione sulla rete cercando di trovare la porta 21 aperta e provare quindi un FTP Bounce Scan su ognuno dei server individuati. Un port scanner, come Nmap, sarà in grado di evidenziare se un host è vulnerabile o meno a questa tecnica.

I. Idle Scan

Questa tecnica consente di analizzare un host senza inviargli direttamente pacchetti, ma passando attraverso una macchina chiamata zombie o bot (vedi nota [1]).
Il lato interessante di questa tecnica è che lo zombie può essere qualsiasi nodo di rete in grado di inviare e ricevere pacchetti sia dall'attaccante che dalla vittima e non è necessario che sia un sistema compromesso. Questo attacco è completamente invisibile: nei log della vittima come il responsabile della scansione risulterà lo zombie e non l'attaccante.
Questo attacco sfrutta una caratteristica intrinseca del protocollo IP. Tutti i pacchetti IP hanno nel loro header hanno un campo identification, utilizzato per distinguere i frammenti appartenenti a pacchetti diversi. La maggior parte dei sistemi operativi incrementa di 1 il contenuto di questo campo per ogni pacchetto inviato. Esaminando

questo valore, che chiameremo in seguito IPID (IP IDentification), in due momenti diversi possiamo quindi misurare il numero di pacchetti inviati da un sistema in quell'intervallo di tempo. Un Idle scan si svolge nel modo seguente:

1. L'attaccante invia un pacchetto, con i flag SYN ed ACK attivi, allo zombie, che risponderà con un RST. Il pacchetto di riscontro contiene un certo IPID, che l'attaccante memorizza.
2. L'attaccante invia un SYN alla vittima, modificando il pacchetto con l'indirizzo del bot.
3. Se la porta è aperta, la vittima risponderà inviando allo zombie un SYN|ACK; a sua volta, lo zombie invia un RST alla vittima in risposta. Se invece la porta è chiusa, la vittima invia un RST, a cui lo zombie non risponde.
4. L'attaccante invia un SYN|ACK allo zombie ed esamina l'IPID del pacchetto RST di risposta. Se l'IPID è aumentato di due rispetto al valore letto al punto 1 significa che lo zombie ha inviato due pacchetti per cui la porta è aperta. Se invece è aumentato solo di uno lo zombie ha inviato solo un pacchetto (la risposta all'attaccante), e la porta è chiusa.

Come dice il nome stesso, un Idle scan funziona solo se lo zombie utilizzato è "idle", ovvero non produce né riceve traffico altrimenti verrebbe modificato il valore dell'IPID rendendo inutile l'attacco falsificando il risultato. Come abbiamo visto, l'Idle scan permette in pratica di eseguire un SYN Scan dal punto di vista dello zombie. Proprio questo punto potrebbe risultare a sfavore dell'attaccante, in quanto è possibile che alcune porte abbiano comportamenti differenti in base al sistema da cui viene contattato. La difesa migliore da questo tipo di attacco è rompere l'ipotesi su cui si basa: se gli IPID non aumentano per ogni pacchetto inviato l'attacco diventa inefficace. Le ultime versioni di alcuni sistemi operativi (OpenBSD, Linux, Solaris) utilizzano una sequenza di IPID non facilmente prevedibile, e quindi impediscono gli Idle scan. Se aggiornare il sistema operativo non è un'opzione praticabile, è bene configurare un firewall per bloccare i pacchetti con indirizzi sorgente palesemente falsificati.

4.2 UDP Scanning

Passiamo quindi a parlare di una differente tipologia di port scanning, in quanto si basa nell'individuazione di porte UDP aperte e non più TCP. Tuttavia, essendo UDP un protocollo senza connessione, i riscontri ottenibili durante la comunicazione sono limitati, ciò porta inevitabilmente ad un utilizzo meno frequente di questa tecnica. Nonostante tutto comunque vedremo l'UDP Scan, supportata nativamente anche da Nmap e che analizzeremo.

J. UDP Scan

Un attaccante sceglie questa tipologia di port scanning se vuole ottenere informazioni per quanto riguarda lo status delle porte UDP. Questa tecnica consiste nell'invio di datagrammi utente UDP di 0 byte alla porta bersaglio, se essa è aperta solitamente non riceveremo alcuna risposta questo perché il protocollo UDP non richiede di stabilire una connessione come ad esempio in TCP con l'handshake a tre vie. Se riceviamo una risposta ciò può dipendere dal comportamento di una specifica applicazione e quindi non possiamo stabilire con certezza se la porta sia chiusa o meno. L'UDP scanning si basa in modo pesante sui messaggi di diagnostica ICMP, ma per esempio diversi firewall possono bloccare la riposta di messaggi di errore di questo tipo, prevenendo così di fatto questa tipologia di attacco. Inoltre questo metodo risulta non sempre efficiente a causa del meccanismo che governa la velocità dei messaggi ICMP, infatti nel kernel di Linux viene limitata la generazione dei messaggi di destination unreachable ad 80 per 4 secondi, con una penalità di ¼ di secondo se questo limite viene sorpassato. Per determinare se una porta risulti chiusa, si aspetta che in riposta al nostro pacchetto sia inviato un messaggio ICMP di tipo 3 (Porta irraggiungibile), mentre la ricezione di messaggi ICMP di tipo diverso determina che la porta bersaglio sia filtrata.
Il comando di Nmap da utilizzare per effettuare tale tecnica è: -sU.

1. Velocità: L'UDP scanning è molto lento a causa del limite di velocità dei messaggi ICMP.
2. Stealth: Questa tecnica è stealth.
3. Open Port: Si deduce che una porta sia aperta se non si riceve alcun pacchetto.

4. Porta chiusa: Si determina una porta chiusa attraverso la ricezione di messaggi di diagnostica ICMP.
5. Porta filtrata: Risulta possibile in base ad alcuni messaggi ICMP di riposta.
6. Porta non filtrata: Risulta possibile in base ad alcuni messaggi ICMP di riposta.

Una particolarità comune a tutte le tecniche visionate fino ad ora, è quella di destinare l'attacco verso uno o più target a partire da un singolo mittente. Queste tecniche per cui seguono un modello a single source (one-to-many). Un'altra famiglia di port scanning invece si basano su un modello distribuito, per cui abbiamo host multipli per l'attacco e una sola macchina bersaglio (many-to-one), oppure diversi mittenti e più host attaccati (many-to-many). Non analizzeremo port scanning che fanno parte di questa famiglia ma per completezza è bene comunque esporre e spiegare il funzionamento generale di queste tecniche.

4.3 Distributed port scanning

La raccolta distribuita di informazioni viene eseguita con l'uso sia di un modello many-to-one sia quello many-to-many. In questo caso, l'attaccante utilizza host multipli per eseguire tecniche di raccolta di informazioni, per far ciò si possono scegliere più vie: rate-limited o random. In una tecnica rate-limited, il numero di pacchetti mandati, ad ogni host, per lo scan è limitato. In particolare ogni tecnica od ogni tipologia di attacco avrà una soglia di pacchetti inviabili diversa. Diversamente si può scegliere di randomizzare le coppie IP-porte del mittente, oppure il tempo di delay per un pacchetto d'intrusione e l'altro. Ciò avviene, come dice il nome stesso, se si usa una strategia di tipo random. I port scanner distribuiti moderni operano da una singola locazione, amministrando e osservando tutte le operazioni che gli host addetti allo scanning (scanning agent), operano su reti anche molto distanti dal controllo centrale. La raccolta di tutti i risultati ricavati dalle varie scansioni avviene in un database centrale, ciò riduce drasticamente il tempo di segnalazione per così poi creare celermente uno snapshot in real-time dello stato di sicurezza della rete.

Tra i vari vantaggi che otteniamo scegliendo uno scanner che fa uso di un modello distribuito abbiamo:

- Gestione centralizzata: Tutte le attività su multiple reti sono condotte e gestite da una singola console e raccolte in un unico database server.
- Performance: Attacchi in parallelo su più reti migliorano di gran lunga le performance.
- Automazione: ogni processo effettuato dagli scanning agent possono essere automatizzati, programmati e controllati da una console centrale. I processi programmabili per esempio possono essere: tempo di partenza e di stop, aggiungere o aggiornare moduli per le attività di scanning, automatizzare i responsi al server centrale.
- Riservatezza: le comunicazioni tra la console di gestione e i vari client sono criptati.

Prendiamo come esempio il software DNmap [19] e, con un semplice schema della sua archiettetura, possiamo osservare e riassumere le caratteristiche appena proposte:

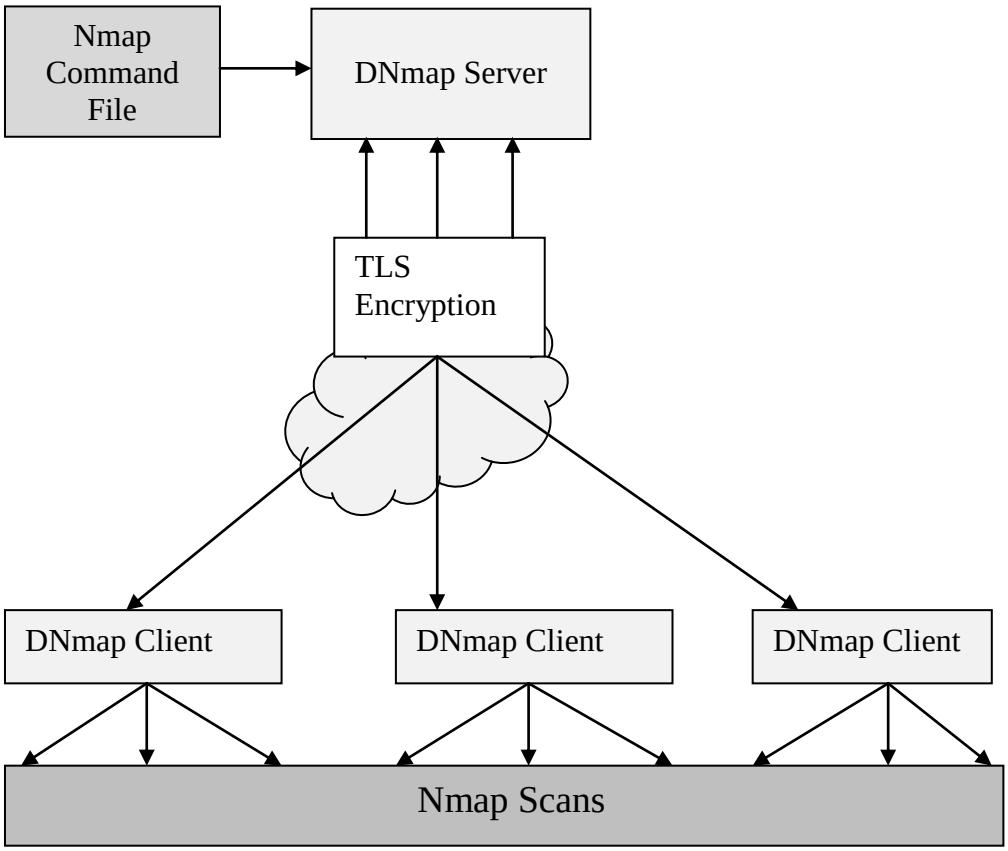


Immagine 26 [19].

5. Analisi del port scanning

5.1. Impostazioni

Nella prima parte del capitolo vogliamo porre all'attenzione del lettore, le analisi effettuate su attacchi di port scanning utilizzando tecniche di tipo SYN scan, ACK scan, FIN scan, Xmas scan e NULL scan, per poi passare al Connect scan ed infine all'UDP scan. Negli screenshot effettuati si può notare per ogni tecnica un sommario dell'attacco stesso, il tempo impiegato e il numero di pacchetti utilizzati. Essi sono tutti dati forniti da Nmap al termine della scansione, per essere precisi è stato adoperato Zenmap [5], cioè una versione di Nmap avente un'interfaccia grafica integrata. La scelta è ricaduta su questa versione dell'applicazione poiché agevola la comprensione delle informazioni trattate, le quali risulteranno più chiare rispetto alla versione unicamente testuale. In merito ai comandi di Nmap sfruttati per le prime analisi, possiamo affermare come sia stato scelto di svolgerli attraverso le impostazioni standard del software. Per cui il comando sarà formato unicamente dal tipo di comando scelto, dagli indirizzi IP degli honeypot della rete e dall'opzione –v che semplicemente aumenta la quantità di informazioni a video che Nmap dovrà fornire durante la scansione stessa. Non fornendo il numero di porte da dover analizzare, vengono scelte in automatico mille delle principali porte note TCP o UDP. Ulteriori opzioni verranno aggiunte, modificate e spiegate in modo più approfondito nella seconda parte del capitolo. Per quanto riguarda i grafici che riportiamo, essi sono costituiti da tre spezzate di colori differenti. La rossa rappresenta la percentuale media di utilizzo della CPU (asse delle ordinate) durante un intervallo di campionamento, nel nostro caso di un secondo, per una periodo di tempo di 100 secondi totali (asse delle ascisse). La linea di color blu rappresenta invece il numero di byte inviati per secondo nell'interfaccia di rete, attraverso la quali verranno svolti gli attacchi. Infine la spezzata verde identifica il numero di byte ricevuti per secondo, nella medesima interfaccia. Sia questo indice che il precedente, sono in scala 1:1000. Tali analisi sono tratti impostando opportunamente il software Performance Monitor, presente nativamente in tutti i dispositivi Windows, dalla versione XP e server NT 4.0 in poi.

Mentre per la rilevazione in real time dell'utilizzo della banda sfruttata dagli honeypot durante gli attacchi, abbiamo sfruttato l'applicazione open-source nload [20], usufruibile su tutte le distribuzioni GNU/Linux.

Per quanto riguarda la banda della connessione utilizzata, parliamo di una rete avente circa 6 Mbit/s in download e 0.8 Mbit/s in upload, quindi piuttosto limitata ma per le nostre analisi è un fattore che ci agevola, infatti sarà una dimostrazione di come questo attacco si comporti in condizioni non ideali. In alcuni casi sono state svolte le medesime scansioni facendo uso della rete fornita dall'università, la quale possiede ben altre caratteristiche: quasi 100 Mbit/s sia in download che in upload. I risultati così ottenuti, potranno essere confrontati e saranno frutto di spunti per alcune considerazioni. Inoltre la CPU montata sulla macchina su cui vengono svolte le rilevazioni è un Intel i7-4700MQ da 2.4 GHz. Invece per quanto riguarda gli honeypot ed il software Honeyd, le impostazioni rimangono quelle descritte nei precedenti capitoli, mentre la macchina virtuale utilizzata per eseguire il suddetto programma è stata installata su un secondo host fisico, opportunamente configurato per poter comunicare con il primo dispositivo da cui partiranno gli attacchi.

Presentiamo qui di seguito i risultati ottenuti per le tecniche precedentemente citate, per poi porre alcune osservazioni.

5.2. Analisi

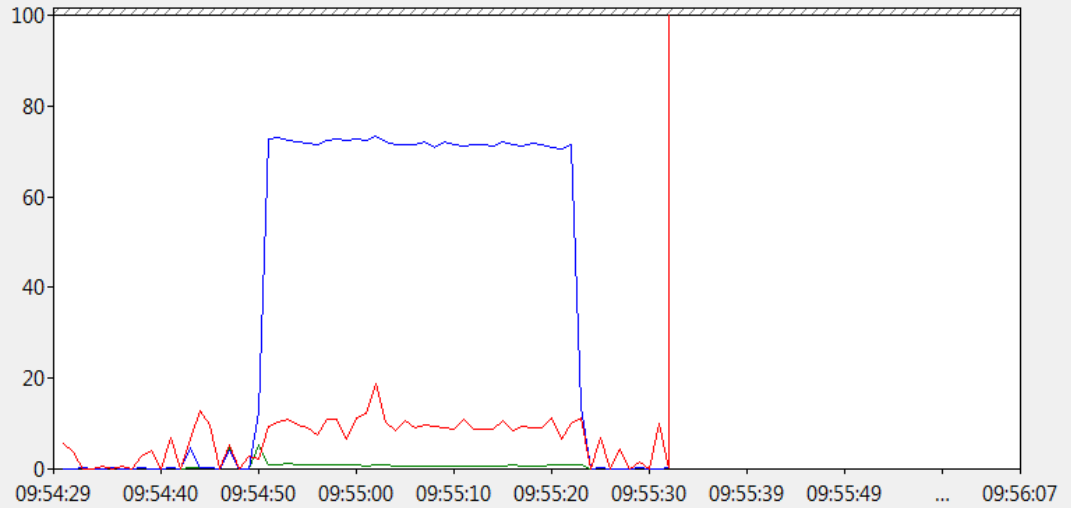
A. SYN Scan

Scan Summary

Nmap 6.47 was initiated at Tue Feb 24 09:54:41 2015 with these arguments:
nmap -sS -v 10.0.1.10-14 10.0.4.100-119 10.0.2.10-14 10.0.3.10-19

Verbosity: 1; Debug level 0

Nmap done: 40 IP addresses (40 hosts up) scanned in 39.70 seconds
Raw packets sent: 40096 (1.763MB) | Rcvd: 41163 (1.677MB)



Ultimo	215,345	Media	615,599	Min	0,000	Max	5.303,381
						Durata	1:40

Mostra	Colore	Scala	Contatore	Istanza	Padre	Oggetto	Computer
☑		1,0	% Tempo processore	_Total	---	Informazioni proces...	\\TONY-PC
☑		0,001	Byte inviati/sec	Realtek ...	---	Interfaccia di rete	\\TONY-PC
☑		0,001	Byte ricevuti/sec	Realtek ...	---	Interfaccia di rete	\\TONY-PC

Immagine 27.

B. ACK Scan

Scan Summary

Nmap 6.47 was initiated at Thu Feb 19 15:35:45 2015 with these arguments:
nmap -sA -v 10.0.1.10-14 10.0.4.100-119 10.0.2.10-14 10.0.3.10-19

Verbosity: 1; Debug level 0

Nmap done: 40 IP addresses (40 hosts up) scanned in 39.27 seconds

Raw packets sent: 40098 (1.604MB) | Rcvd: 40746 (1.650MB)

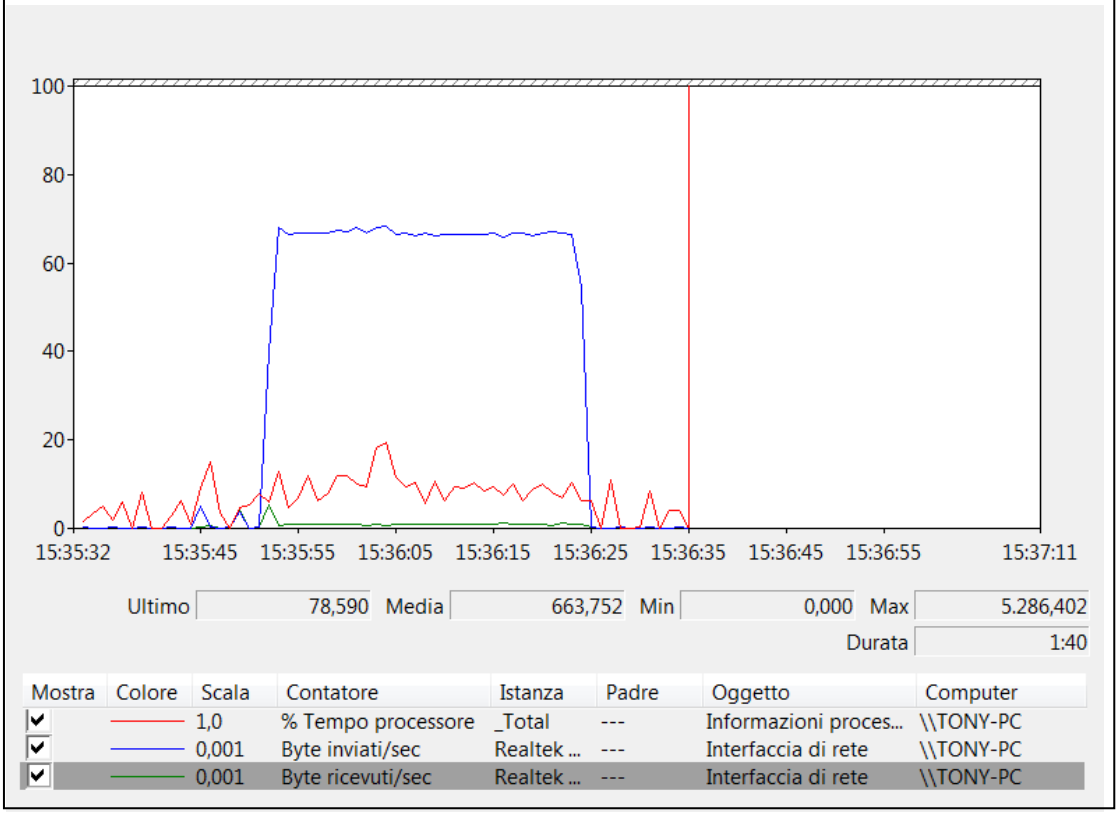


Immagine 28.

C. FIN Scan

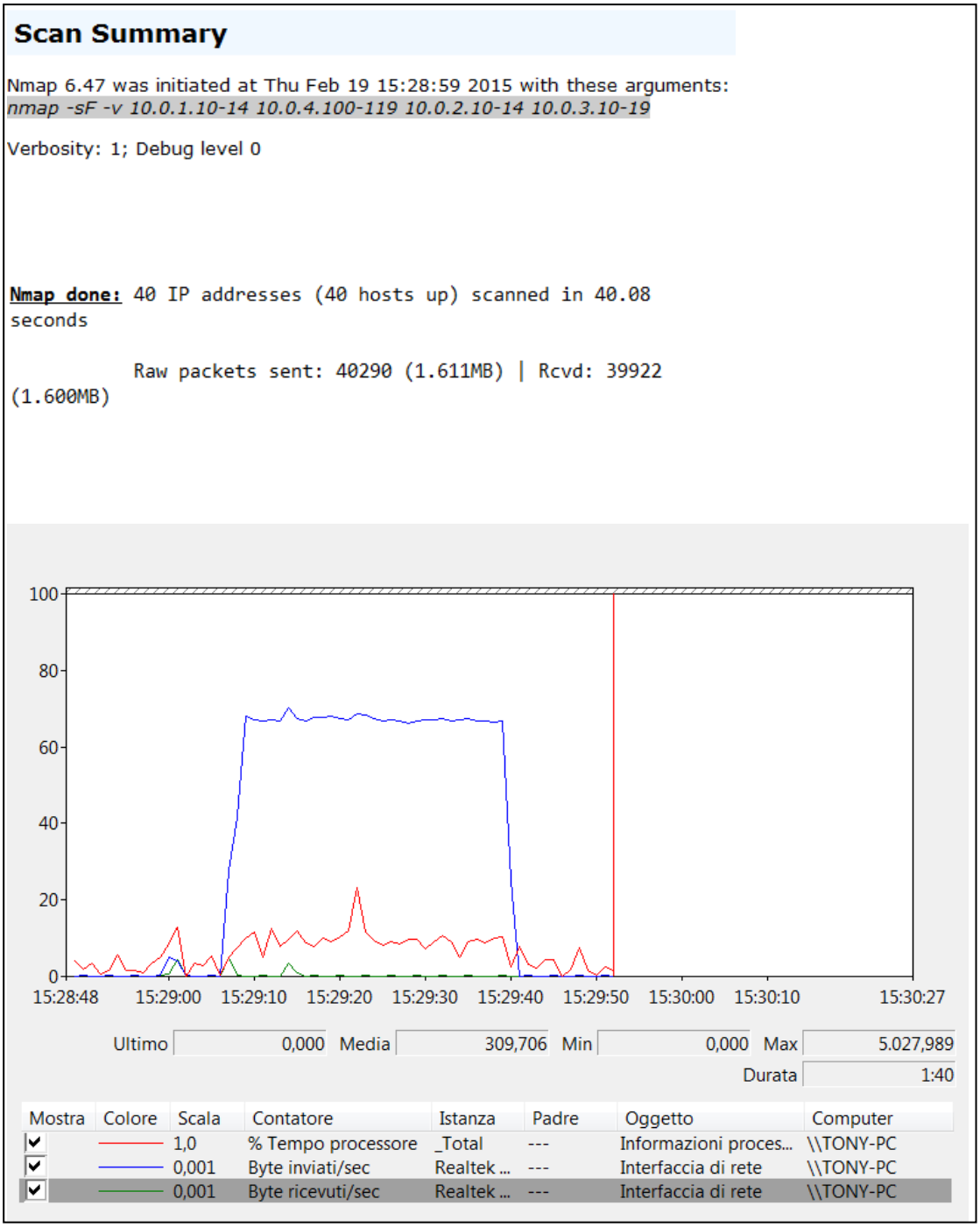


Immagine 29.

D. Xmas Scan

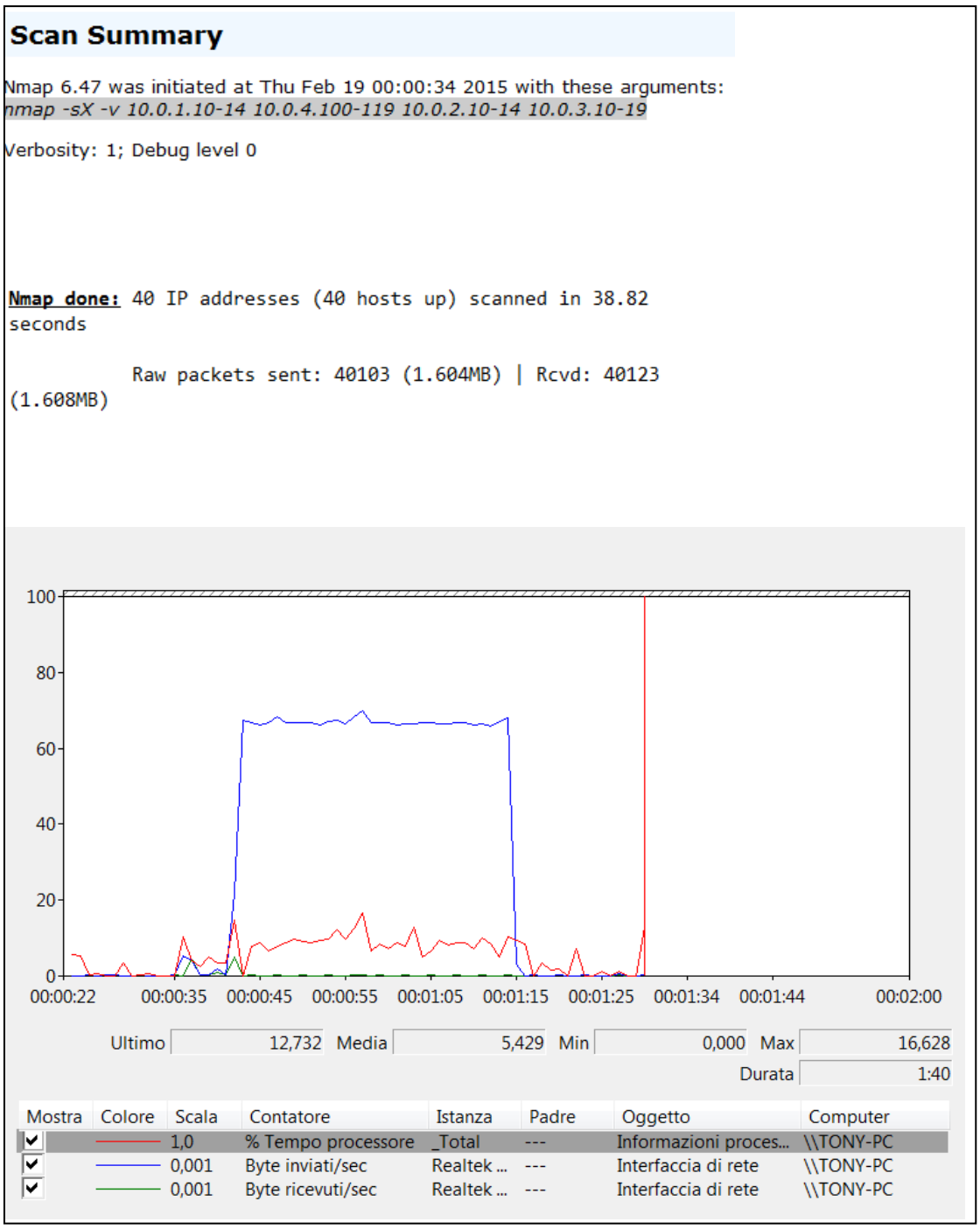


Immagine 30.

E. NULL Scan

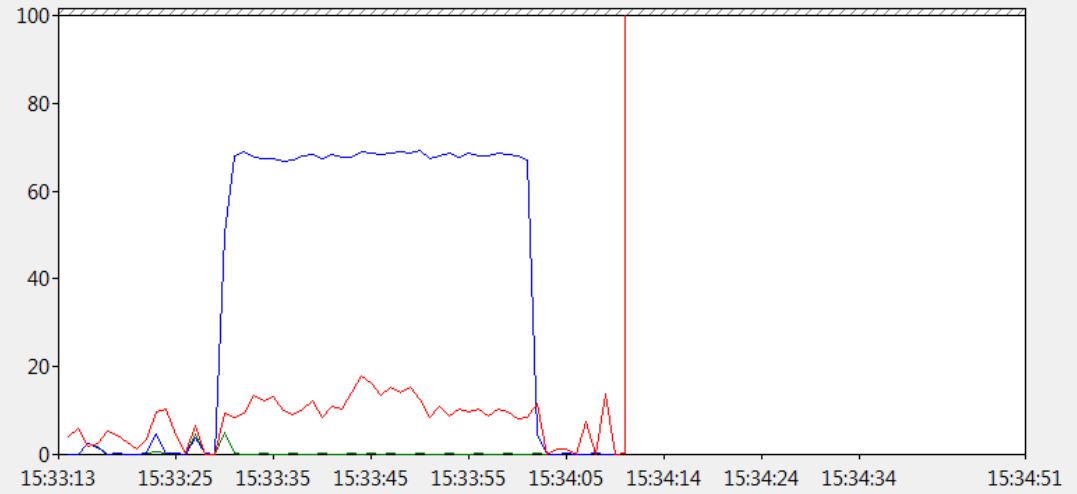
Scan Summary

Nmap 6.47 was initiated at Thu Feb 19 15:33:22 2015 with these arguments:
nmap -sN -v 10.0.1.10-14 10.0.4.100-119 10.0.2.10-14 10.0.3.10-19

Verbosity: 1; Debug level 0

Nmap done: 40 IP addresses (40 hosts up) scanned in 38.72 seconds

Raw packets sent: 40095 (1.603MB) | Rcvd: 40111 (1.607MB)



Ultimo	215,406	Media	345,100	Min	0,000	Max	5.065,947
						Durata	1:40

Mostra	Colore	Scala	Contatore	Istanza	Padre	Oggetto	Computer
☑		1,0	% Tempo processore	_Total	---	Informazioni proces...	\\TONY-PC
☑		0,001	Byte inviati/sec	Realtek ...	---	Interfaccia di rete	\\TONY-PC
☑		0,001	Byte ricevuti/sec	Realtek ...	---	Interfaccia di rete	\\TONY-PC

Immagine 31.

Siamo già in grado di esporre alcune osservazioni ottenute in seguito a queste prime analisi. Se osserviamo i grafici in corrispondenza dell'istante di inizio del port scanning, è possibile notare appunto un aumento della percentuale di utilizzo della CPU, con picchi attorno al 15-20%. Lo scanning durano sempre attorno ai 40 secondi, e per questo intervallo di tempo si noti appunto l'andamento differente della risorsa rispetto agli istanti che precedono e seguono l'attacco. Sulla durata totale della scansione, i primi 5 secondi sono sfruttati per eseguire ping sugli host bersaglio. Questa funzionalità viene solitamente sempre sfruttata, sarebbe inutile attaccare device non attivi, e per questo motivo non è stata disattiva nonostante noi sapessimo per certo che i dispositivi fossero up. A livello prestazionale, per tutte queste tecniche rileviamo una sostanziale uguaglianza questo è dovuto alla somiglianza nella loro implementazione. Esse si basano tutti sull'invio di segmenti TCP con particolari flag attivi, le differenze sostanziali si trovano in ciò che si cerca (vedi paragrafo precedente), condizione dipendente unicamente dalle esigenze dell'attaccante, e nel tipo di risposta ottenuta, variante che però non modifica in modo sensibile le performance. Per quanto concerne la banda utilizzata dagli honeypot per l'invio dei pacchetti di risposta otteniamo, anche in questo caso, un risultato equivalente per le diverse tecniche e che presentiamo attraverso uno screenshot rappresentativo per tutte quante.

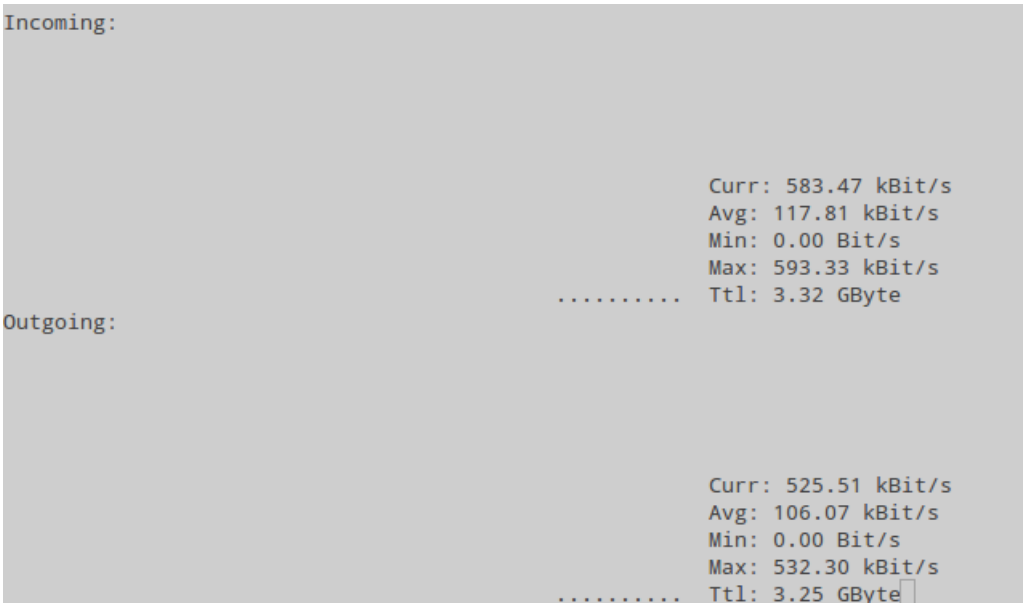


Immagine 32.

Osservando la precedente immagine è necessario comunque effettuare una piccola precisazione. I dati ottenuti in quest'ultimo caso sono in KBit/s mentre nei grafici precedenti il traffico è stato misurato in Kbyte/s. Per mantenere un'unità di misura equivalente è necessaria una semplice conversione. Riassumendo otteniamo quindi in entrata 72.93 Kbyte/s, mentre in uscita 65.69 Kbyte/s, valori comunque che difficilmente possono saturare una banda anche di una connessione mediocre, salvo ovviamente casi particolari in cui era già presente un intenso traffico non dipendente dall'attacco stesso. Le nostre rilevazioni sono state tutte effettuate in condizioni di traffico iniziale assente, così da presentare al lettore un dato privo di alcuna influenza esterna alla scansione. Durante lo svolgimento dell'attacco tali valori possono leggermente variare, i quali rimangono comunque nell'ordine di pochi Kbyte.
Lo SYN scan è stato eseguito anche tramite la connessione fornita dall'università. Prima di riportare le differenze che si sono evidenziate, è opportuno sottolineare come su questa rete sia presente un firewall. Questa restrizione, assente invece durante le analisi precedenti, blocca i messaggi di ping, costringendoci di fatto ad omettere questa funzionalità tramite il comando –Pn. Inoltre, per il medesimo motivo, ora delle 1000 porte scansionate per host, 999 risultano filtrate ed 1 chiusa.

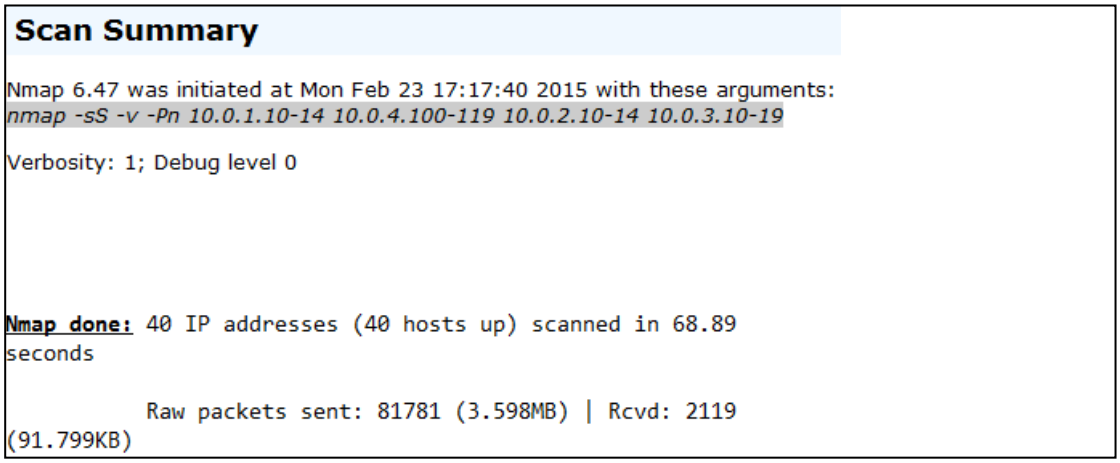


Immagine 33.

Un'ulteriore considerazione è possibile effettuarla osservando i secondi impiegati. Nonostante una banda più generosa, la durata dell'attacco è stata maggiore. La spiegazione è da ritrovare nel tempo necessario al firewall per l'analisi dei pacchetti sottoposti, un fattore che incide per circa un 58% nella tempistica totale dell'attacco.

Per quanto riguarda CPU e banda, rimandiamo il lettore alla visione della figura 27. poiché i dati ottenuti sono del tutto equiparabili. Unica nota da rilevare è il valore medio di 1 o 2 Kbyte/s in download, comportamento dovuto al filtraggio operato dal firewall. Il perché di tutte queste somiglianze nonostante una connessione più veloce verrà esposto nel prossimo paragrafo. Ora osserviamo le caratteristiche della TCP Connect scan.

F. Connect Scan

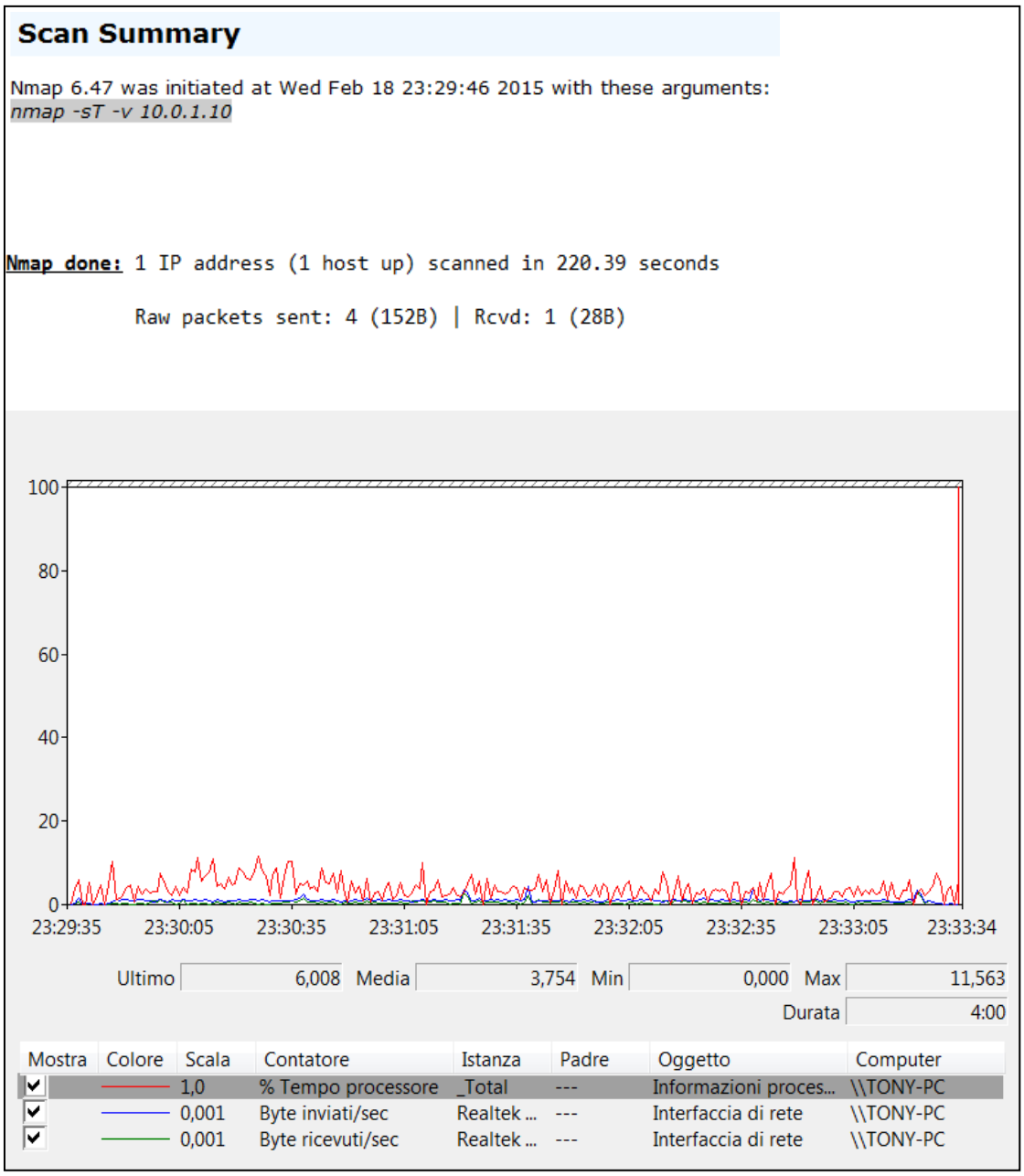


Immagine 34.

Il tempo impiegato da questa tecnica come è possibile evincere dalle figure, è davvero elevato confrontato con le altre principali metodologie d'attacco. Rimane comunque utile nel caso si volesse scegliere di adoperala in combinazione con altre, per esempio l'Identd Scanning (vedi paragrafo precedente), altrimenti solitamente tecniche come il SYN scan, sono da preferire. Per osservare la banda e la CPU utilizzata, presentiamo l'attacco svolto su un solo host, questo perché il grafico riportato in un intervallo di tempo così ampio, come quello di un attacco Connect scan su 40 host, risulterebbe poco chiaro, ed infatti una scansione del genere impiega oltre 22 minuti. Si nota come tutte le risorse siano utilizzate solo in minima parte e in forma ridotta rispetto alle altre tecniche, questo a causa dell'overhead necessario all'instaurazione della connessione, dilatando di fatto il delay tra un pacchetto di intrusione e il successivo.

G. UDP Scan

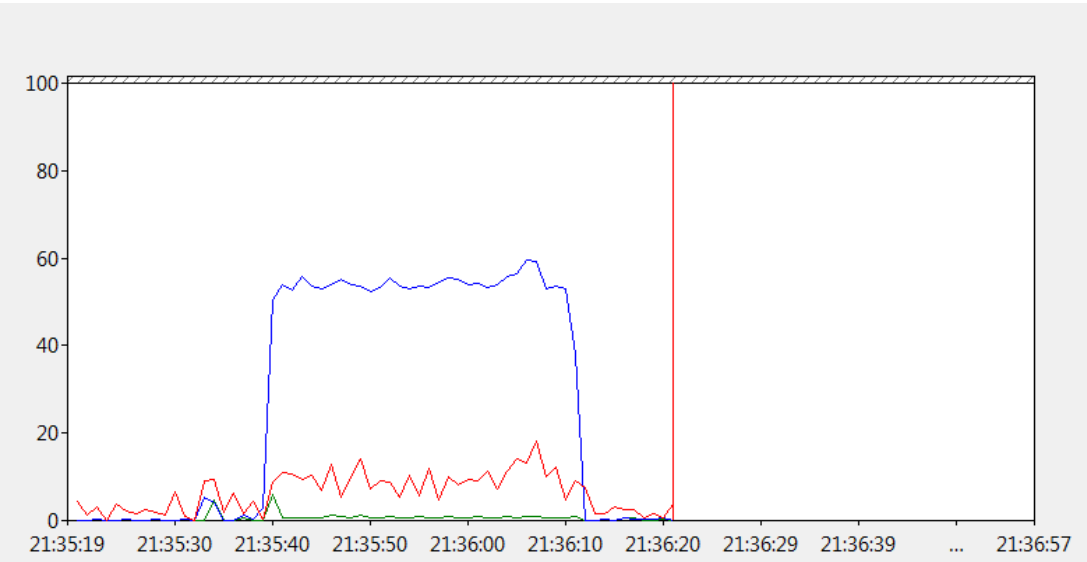
Scan Summary

Nmap 6.47 was initiated at Wed Feb 18 21:35:32 2015 with these arguments:
nmap -sU -v 10.0.1.10-14 10.0.4.100-119 10.0.2.10-14 10.0.3.10-19

Verbosity: 1; Debug level 0

Nmap done: 40 IP addresses (40 hosts up) scanned in 38.14 seconds

Raw packets sent: 40167 (1.159MB) | Rcvd: 40653 (2.292MB)



Ultimo	0,000	Media	618,956	Min	0,000	Max	5.877,680
						Durata	1:40

Mostra	Colore	Scala	Contatore	Istanza	Padre	Oggetto	Computer
☑		1,0	% Tempo processore	_Total	---	Informazioni proces...	\\TONY-PC
☑		0,001	Byte inviati/sec	Realtek ...	---	Interfaccia di rete	\\TONY-PC
☑		0,001	Byte ricevuti/sec	Realtek ...	---	Interfaccia di rete	\\TONY-PC

Immagine 35.

Il risultati ottenuti inizialmente per l'UDP scanning risultano particolari. Durante il precedente capitolo era stato spiegato come questa tecnica risultasse nella gran parte dei casi molto lenta, a causa del meccanismo che regola la velocità dei messaggi ICMP. Nel nostro caso perché non avviene questo rallentamento? Nel momento in cui assegniamo una personalità ad un honeypot, stiamo solamente fornendo un modello da seguire per lo stack di rete, la gestione dei messaggi ICMP invece è a livello di kernel e per questo motivo tale comportamento non viene simulato. Per ottenere risultati differenti da quelli precedenti è sufficiente modificare opportunamente il file di configurazione assegnando, per ogni porta non aperta del template prescelto, un comportamento di tipo block, in questo modo:

set winclient default udp action block

Di default avevamo assegnato il comando reset, che provocava l'invio di messaggi ICMP di tipo porta irraggiungibile mentre con block stiamo di fatto simulando un firewall, infatti verranno droppati tutti i pacchetti in arrivo alle porte che non risultano aperte ma non solo, infatti Honeyd gestirà i pacchetti in maniera tale che non vengano neppure segnalati nel log. Nmap interpreta questo comportamento come se il pacchetto non abbia mai raggiunto la destinazione, provocando un rallentamento nell'invio dei successivi datagrammi utente UDP, in modo da non intensificare la congestione che potrebbe affliggere la rete.
Per poter aver prestazione migliori è necessario effettuare più attacchi contemporaneamente oppure scegliendo manualmente solamente le porte che più interessano.

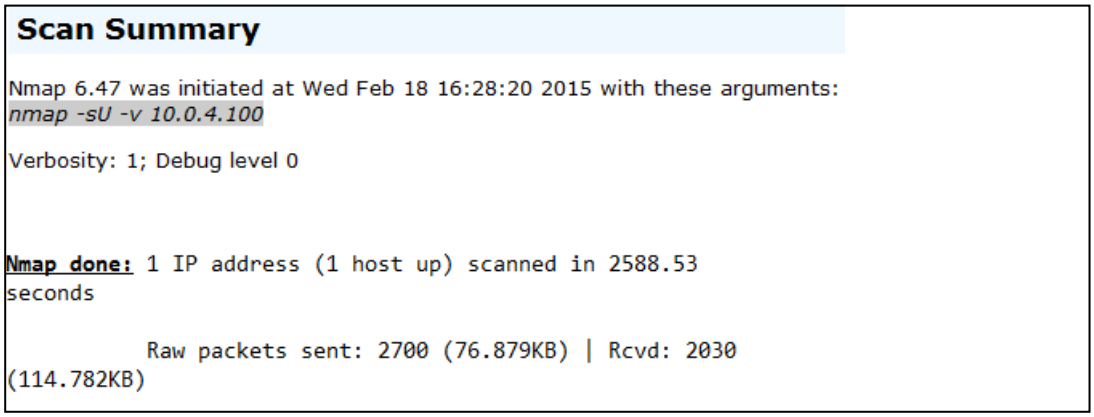


Immagine 36.

Per la scansione di un solo host, siamo arrivati ad un tempo necessario di oltre 41 minuti, la differenza è evidente, per cui questa tecnica risulta realmente utile se l'attaccante ha come scopo la scansione di porte ben precise e limitate, altrimenti il rischio di impiegare tempi lunghissimi risulta molto elevato. Inoltre, per quanto riguarda l'utilizzo di CPU e banda, i risultati e le motivazioni di tali comportamento sono del tutto simili e riconducibili ai dati ottenuti durante la Connect scan, per questa ragione si è scelto di omettere il grafico.

5.3. Timing di Nmap

In questo paragrafo proveremo a modificare le impostazioni standard di Nmap, per poi osservare come questi cambiamenti riescano o meno a modificare la quantità di risorse necessarie ad un attacco, od il tempo impiegato con le opzioni di default. Ci soffermeremo in particolare sul timing di attacco, spiegheremo cosa si intende, come viene gestito da Nmap e come può modificare la velocità del port scanning. A tal proposito il software ci mette a disposizione tutta una serie di comandi per gestire il delay tra un pacchetto intrusivo e l'altro, minore sarà questo intervallo di tempo, e più velocemente porteremo a termine l'attacco a discapito però di una più elevata probabilità con cui un IDS potrebbe rilevare l'attacco, per il semplice motivo che una gran quantità di pacchetti verso un ampio numero di porte durante un intervallo di tempo ristretto, risulta più sospetto di un'affluenza sporadica verso l'host stesso. Gli IDS solitamente basano il loro algoritmo di rilevazione su delle soglie. Il sistema controlla le possibili intrusioni per un determinato intervallo di tempo. Ciò aiuta a prevenire falsi positivi da utenti normali, inoltre è essenziale per risparmiare risorse, e per non rendere la ricerca di intrusioni in real time eccessivamente lento. La strategia dell'attaccante è quindi quella di mantenere la frequenza tra un tentativo di intrusione e l'altro, al di sotto della soglia.

Nmap per questo scopo fornisce i seguenti comandi:

--scan_delay <milliseconds>; --max_scan_delay <milliseconds>

Esso determina l'intervallo di tempo (espresso in millisecondi), che il software dovrà attendere almeno tra un pacchetto e l'altro, come spiegato poco sopra. Esso risulterà particolarmente utile quando sapremo a priori con quale frequenza avremo una

risposta dall'host bersaglio. Se tale valore corrisponde al delay prestabilito, non ci saranno tempi di attesa del tutto inutili, che un'impostazione di default potrebbe causare. Un'altra situazione di utilizzo, può nascere se dobbiamo risultare stealth nei confronti di uno specifico IDS, come appunto spiegato precedentemente. Nei casi in cui non si sia in grado di configurare opportunamente tali opzioni, Nmap ci viene incontro e fornisce delle impostazioni standard.

Attraverso il comando:

-T<0-5>: viene impostato un template di temporizzazione, a cifre maggiori corrisponde una più elevata velocità dell'eseguire la scansione. L'utente potrà quindi decidere l'aggressività desiderata in modo semplice, delegando a Nmap il comportamento da usare. Con il valore 0, il delay viene impostato a 5 minuti, mentre avremo intervalli di tempo di 15 secondi, 0.4 secondi, 10 millisecondi (il quale rappresenta il valore di default se tale comando viene omesso), 8 millisecondi e 5 millisecondi, rispettivamente per i valori 1, 2 ,3 ,4 e 5. Si è scelto quindi di testare nuovamente, nelle medesime condizioni, il SYN scan, tra le più rappresentative delle tecniche di TCP scanning, questa volta utilizzando questi comandi di timing per analizzare come questa impostazione sia in grado o meno di modificare le risorse impiegate, rispetto ad un uso standard.

Prima di tutto, nei casi di 0 e 1, abbiamo delay talmente ampi che è opportuno utilizzarli solamente se la necessità di avere un'alta possibilità di sfuggire alla identificazione di un IDS, risulta essere primaria per l'attaccante. Nel caso si volesse eseguire una scansione con il comando –T0 , su 1000 porte di un solo host, il tempo necessario si aggira attorno ai 2 giorni, mentre nel caso –T1 circa 4 ore. Per quanto concerne il comando –T2 invece, abbiamo ottenuto i seguenti risultati:

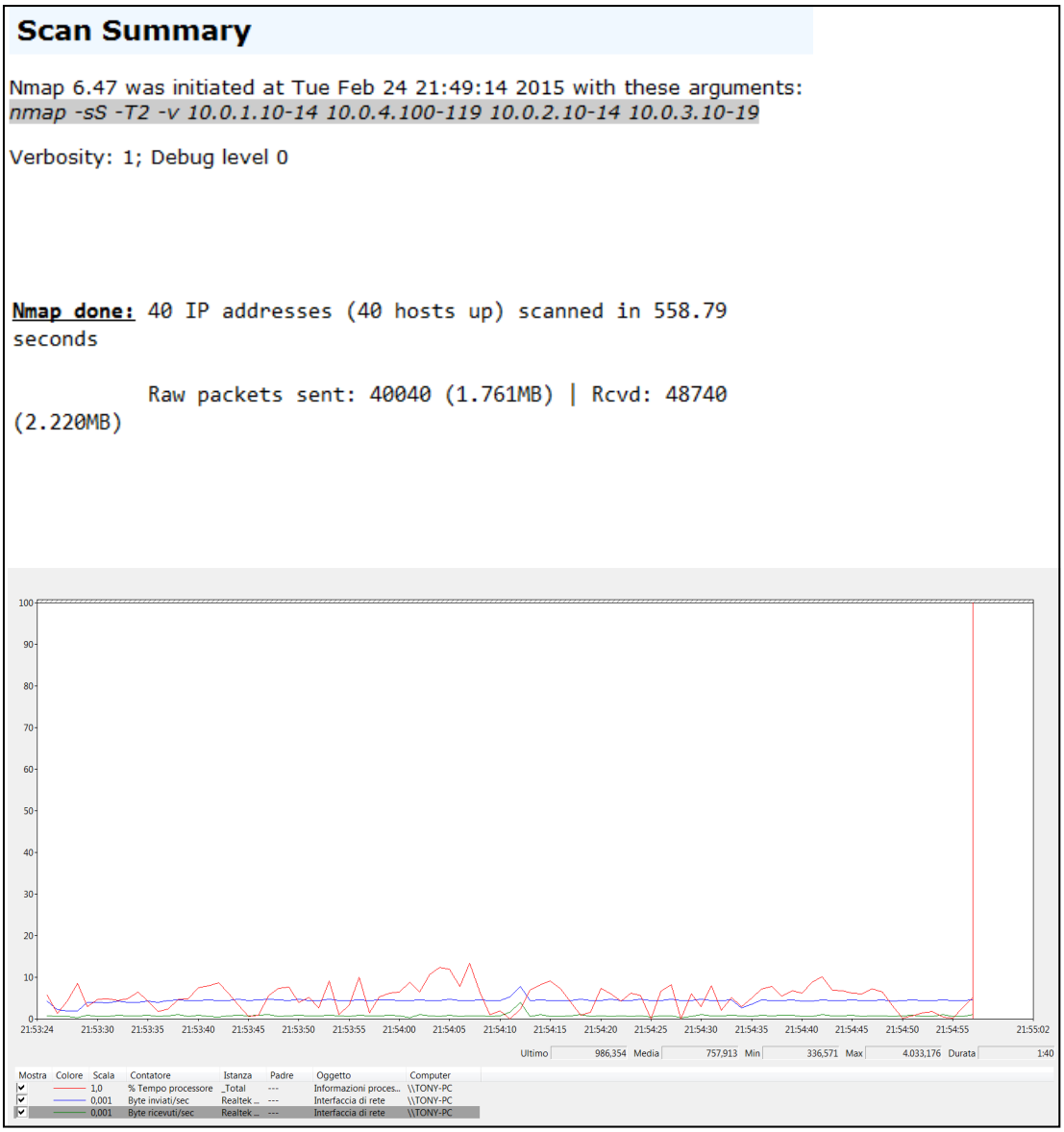


Immagine 37.

Riassumendo abbiamo 558.79 secondi impiegati, mentre per quanto riguarda CPU e banda, si può osservare come vengano sfruttate per quasi la metà rispetto all'impostazione standard. A discapito quindi di una dilatazione delle tempistiche per la scansione, abbiamo una riduzione notevole delle altre risorse, risultando utile nel caso si volesse interferire con il traffico dell'host bersaglio, solamente in maniera marginale. Nel caso degli attacchi con l'opzione –T4, non abbiano rilevato differenze degne di nota rispetto alla versione –T3 (cioè quella di default e già presentata), per questo motivo non abbiamo riaggiunto grafici o informazioni che sarebbero risultati ridondanti. I 2 millisecondi di differenza di delay, almeno nel nostro ambiente di analisi, non hanno portato a reali vantaggi.

Presentiamo ora i dati ottenuti con il comando –T5:

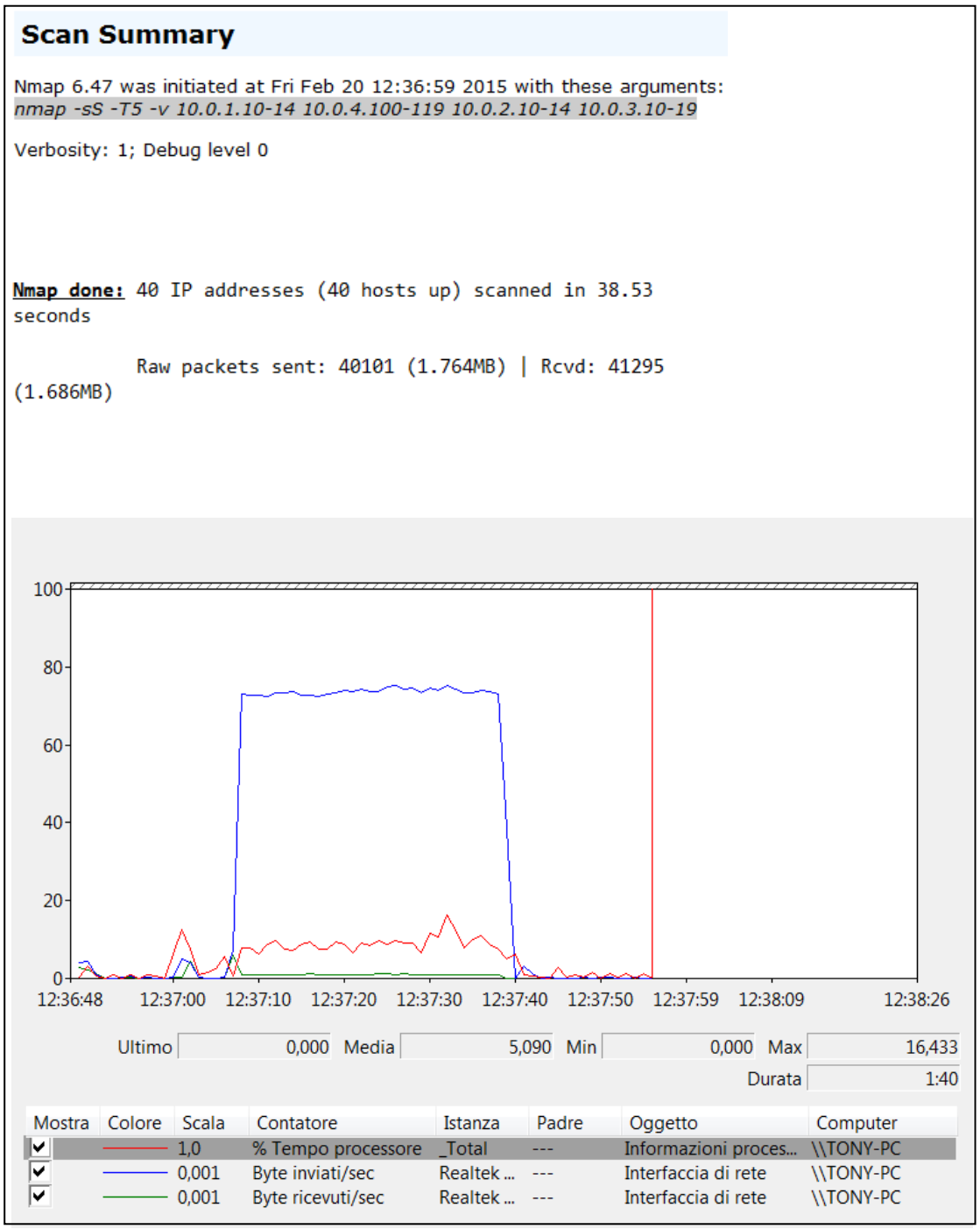


Immagine 38.

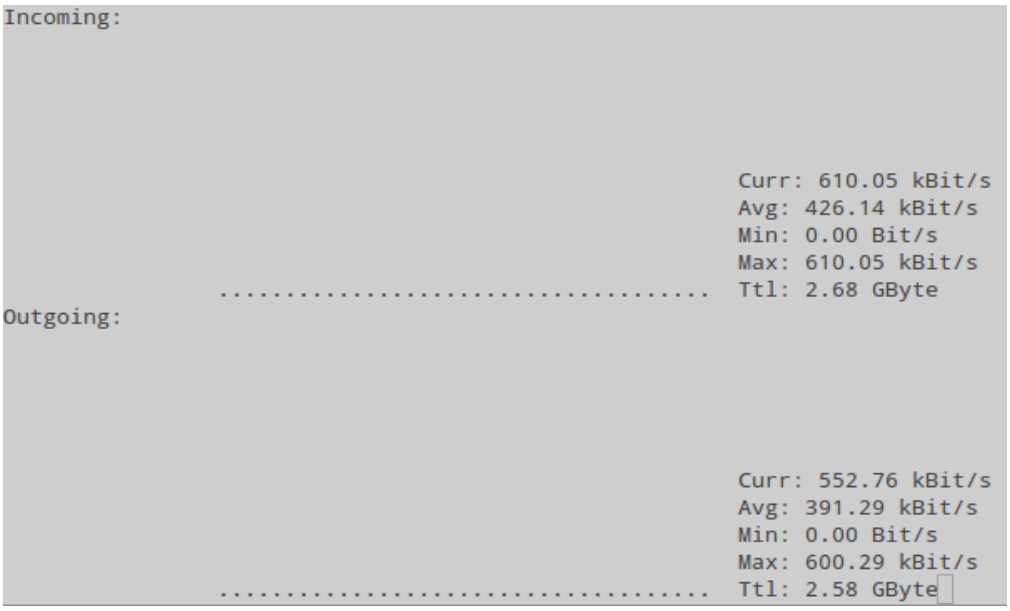


Immagine 39.

Le differenze principali si notano nel tempo impiegato, di circa un secondo in meno rispetto alla versione standard, e riguardo al traffico generato dagli honeypot. Dopo opportune conversioni otteniamo quindi in download 76.26 Kbyte/s mentre in upload 69.1 Kbyte/s, quindi rispettivamente circa 3.33 Kbyte/s e 3.41 Kbyte/s di differenza. Non saranno differenze sostanziali ma sono state comunque giustamente riportate per completezza.

Prima di presentare ulteriori osservazioni ricordiamo come anche in questo caso abbiamo svolto la medesima analisi con la connessione fornita dall'università, ottenendo risultati del tutto comparabili con le considerazione effettuate nel paragrafo precedente.

A fronte di informazioni così simili, nonostante le connessioni differiscano non poco, risulta evidente come la velocità dell'attacco e le altre caratteristiche analizzate non dipendano dalla connessione stessa, ma bensì dal delay impostato e dal livello di parallelizzazione dell'attacco gestito da Nmap. In particolare quest'ultimo fattore solitamente non viene modificato, infatti il software è in grado in totale autonomia di scegliere adeguatamente e con la massima ottimizzazione, il valore più adatto, anche se vengono comunque forniti comandi per determinare manualmente il numero minimo e massimo di millisecondi, che intercorrono tra due pacchetti intrusivi paralleli:

--min_parallelism <milliseconds>; --max_parallelism <milliseconds>.

La temporizzazione scelta per l'attacco desiderato, risulta quindi di fondamentale importanza, una decisione troppo aggressiva porta quasi inevitabilmente ad essere individuati da parte di un IDS, ma non solo, si mette a repentaglio l'ottenimento di un corretto risultato, in quanto più pacchetti vengono inviati in un intervallo di tempo ristretto, e minore sarà l'accuratezza ottenuta, e per di più viene incrementato il flusso di traffico in entrata e in uscita per la rete soggetta all'attacco (vedi figura 39). Mentre una scelta troppo conservativa potrebbe portare a lunghissime attese per il completamento della scansione. Oltre tutto un'impostazione manuale riduce l'abilità di Nmap di controllare dinamicamente il parallelismo, basandosi di fatto sulle condizioni della rete.

5.4. Considerazioni finali e possibili lavori futuri.

In conclusione si può riflettere su come gli attacchi basati sul port scanning non vadano ad inficiare sull'utilizzo della CPU in modo aggressivo, ma bensì i veri punti focali su cui l'attaccante deve riflettere sono la variabile tempo e banda della vittima. Attraverso queste analisi abbiamo osservato come le diverse tecniche hanno tempistiche diverse per il loro completamento, un aspetto che in base alle proprie esigenze va comunque trattato. Dopo tutte le analisi e le considerazioni svolte risulta evidente come il port scanning non sia una tipologia di attacco banale, molti possono essere i fattori da dover tenere in considerazione e anche la minima informazione riguardante il nostro bersaglio potrà rivelarsi utile. Venire a conoscenza del tipo di IDS utilizzato dalla vittima, ci permette di scegliere un timing adatto oppure la presenza o meno di un firewall potrà spingerci verso all'utilizzo di una tecnica piuttosto che un'altra e così ancora per altre possibili variabili. Attenzione, conoscenza e astuzia dell'attaccante risultano ben più efficaci di una connessione ultra veloce. I possibili lavori correlati possono essere molteplici, uno su tutti attacchi di port scanning distribuiti, ed eventualmente con opportuni confronti rispetto a diversi aspetti della loro controparte a mittente singolo.

6. Bibliografia

[1] Anonimo, *Internet Census 2012*, http://internetcensus2012.bitbucket.org/paper.html, 2012.

[2] RFC 792, *Internet Control Message Protocol*, https://tools.ietf.org/html/rfc792, 1981.

[3] RFC 793, *Transmission Control Protocol*, https://www.ietf.org/rfc/rfc793.txt, 1981.

[4] RFC 3168, *The Addition of Ecplicit Congestion Notification (ECN) to IP*, https://tools.ietf.org/html/rfc3168, 2001.

[5] Nmap, http://nmap.org/, 1999.

[6] Snort, https://www.snort.org/.

[7] Aide, http://aide.sourceforge.net/.

[8] Xprobe, http://sourceforge.net/projects/xprobe/.

[9] Honeydrive, http://sourceforge.net/projects/honeydrive/.

[10] VMWare Workstation, http://www.vmware.com/products/workstation/, Ultimo Accesso: 08.01.2015.

[11] N. Provos, T. Holz, *Virtual Honeypots. From Botnet Tracking to Intrusion Detection*, Addison-Wesley, 2008.

[12] J. Gadge, A.A. Patil, *Port scan detection in Networks*, ICON 2008. 16th IEEE International Conference on. 2008. IEEE, 2008.

[13] A. K. Kaushik, E.S. Pilli, R.C. Joshi, *Network Forensic System for Port Scanning Attack*, IEEE 2nd International Advance Computing Conference, 2010.

[14] C. B. Lee, C. Roedel, S. Elena, *Detection and characterization of port scan attacks*, Technical report. University of California, San Diego, CA., 2003, http://cseweb.ucsd.edu/users/clbailey/PortScans.pdf.

[15] M. De Vivo, E. Carrasco, G. Isern, G. O. De Vivo, *A review of port scanning techniques*, SIGCOMM Comput. Commun., Rev., 29, 41–48, 1999.

[16] M. H. Bhuyan, D. K. Bhattacharyya, J. K. Kalita. *Surveying port scans and their detection methodologies*, The Computer Journal, 54(10), 1565-1581, 2011.

[17] G. F. Lyon, *Nmap Network Scanning. Official Nmap Project Guide to Network Discovert and Security Scanning*, Sunnyvale, CA, Insecure.com LLC, 2008.

[18] RFC 1413, *Identification Protocol*, http://tools.ietf.org/html/rfc1413

[19] Dnmap, http://mateslab.weebly.com/dnmap-the-distributed-nmap.html.

[20] Nload, http://www.roland-riegel.de/nload/.