13 AGOSTO 2017

HONEYPOT & HONEYNET

STUDIO E IMPLEMENTAZIONE

PAOLO IPPOLITO
M6300449

Sommario

Introduzione

Con l'evoluzione di internet e delle tecnologie oggigiorno è aumentato il rischio di attacchi informatici; in particolare spesso è possibile incappare in sistemi non configurati correttamente che permettono, tramite le tecniche di *information gathering*, uno studio approfondito delle sue vulnerabilità per poi poter in qualche modo minare una delle tre caratteristiche fondamenti della sicurezza (*Confidentiality, Integrity, Availability*).

Tra le moderne tecniche di difesa e di prevenzione contro questo tipo di attacchi c'è quella esporre un sistema fittizio creato ad arte per contenere configurazioni errate e vulnerabilità in modo da attrarre diversi attaccanti in modo da far ottenere informazioni errate durante la fase di information gathering e far *'perdere tempo'* durante l'attacco vero e proprio. Questi sistemi sono chiamati Honeynet, e sono composti da 2 o più nodi detti Honeypot.

In questo elaborato nella prima parte verrà introdotto il concetto di Honeypot, la classificazione in base al livello di interazione e verranno descritte le differenze tra una Honeypot e un IDS. Nella seconda parte verrà illustrato un esempio di implementazione di una honeynet a partire da honeyd.

Honeypot

Cos'è un honeypot

```
"A server that is configured to detect an intruder by mirroring a real
production system. It appears as an ordinary server doing work, but all the
data and transactions are phony. Located either in or outside the firewall,
the honeypot is used to learn about an intruder's techniques as well as
determine vulnerabilities in the real system"   [1]
```

Letteralmente 'barattolo del miele', un **honeypot** in ambito informatico è un nodo fittizio che simula il comportamento di un nodo reale in modo da poter ingannare un eventuale attaccante sia in fase di *information gathering*, sia in fase di attacco vero e proprio.

Un honeypot registra ogni interazione tra se stesso e l'utente e, dato che non espone alcun servizio effettivo ma soltanto fittizio, ogni utente che andrà a interagire con un honeypot sarà un utente malevolo. In questo modo è possibile ottenere diversi vantaggi:

- L'utente malevolo sprecherà tutto il tempo e le risorse per manomettere un nodo fittizio
- E' possibile ottenere informazioni su tecniche di information gathering o di intrusione non conosciute precedentemente
- E' possibile tracciare l'utente malevolo se quest'ultimo non ha adottato tecniche di impersonation o non ha utilizzato delle proxychain
- E' possibile contrattaccare l'utente malevolo, ad es. piazzando all'interno degli honeypot dei file (pdf o word) con all'interno macro o scripting virus .

Tramite gli honeypot è possibile scoprire le vulnerabilità *0-days*, ovvero le vulnerabilità ancora non emerse o non condivise dagli hacker, che sono naturalmente anche le più pericolose in quanto non esistono patch o versioni software che le risolvono.

Tassonomia

Classificazione per scopo

Una prima classificazione degli honeypot può essere applicata all'uso di cui se ne fa [2]. Possiamo definire 2 categorie:

- Honeypot di Produzione: sono sistemi che, inseriti in una rete di nodi, cercano di prevenire e individuare attacchi, e possono essere usati per effettuare un'analisi offline degli attacchi in modo da poter capire le tecniche di attacco e il motivo dell'attacco stesso (compromissione del sistema, furto di dati sensibili, ...)
- Honeypot di Ricerca: sono utilizzati principalmente per studiare i cyber-attacchi e per avere maggior informazioni sui 'nemici'. Può essere visto come uno strumento 'didattico'.

La differenza tra le due tipologie di honeypot la si può notare in fase di progettazione: mentre quelli di Produzione sono progettati per la prevenzione e l'individuazione andando eventualmente ad inserire appositamente delle vulnerabilità, quelli di Ricerca sono progettati in modo da poter studiare gli attacchi ed eventualmente scoprire le vulnerabilità *0-days*.

Classificazione per livelli di interazione

Gli Honeypot si possono raggruppare in base al livello di interazione che può avere con l'utente [2], e si suddividono in:

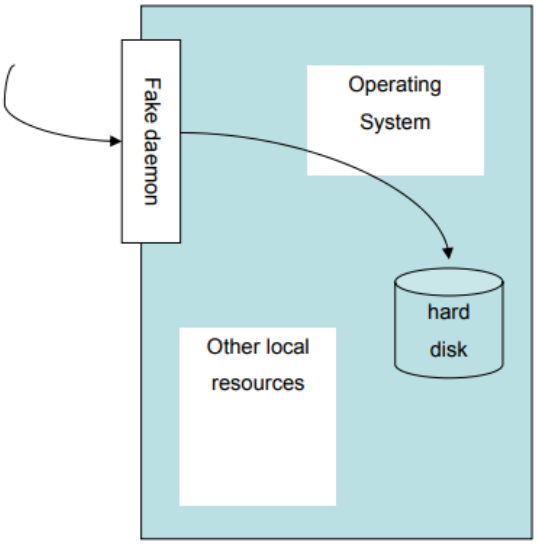


Fig. 1 - Low-Interaction Honeypot [3]

- Bassa interazione (*Low-Interaction*): l'interazione con l'hacker simulata è monodirezionale, il sistema espone dei servizi non interattivi che possono quindi soltanto ricevere dei pacchetti; l'intento principale è quello di intercettare il traffico non autorizzato.

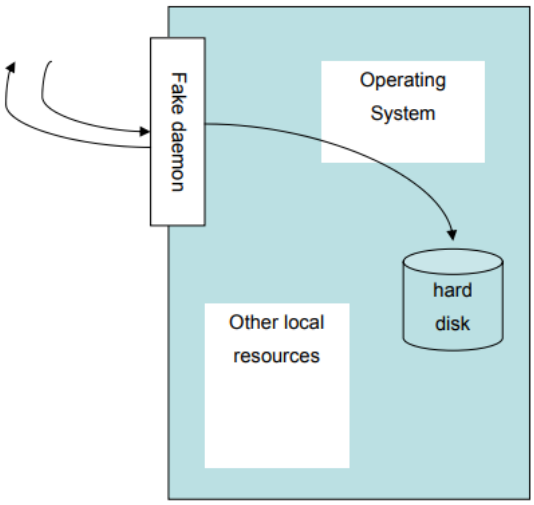


Fig. 2 - Medium-Interaction Honeypot [3]

- Media interazione (*Medium-Interaction*): l'interazione con l'hacker simulata non si limita soltanto all'esposizione di servizi, ma anche ad una simulazione quanto più reale di quest'ultimi. Ad es. i servizi HTTP simulano un determinato web server con la possibilità di chiudere un TCP 3-way handshake e ritornare delle risposte HTTP con dei payload 'logici (pagine web, stringhe json, documenti xml,etc..).

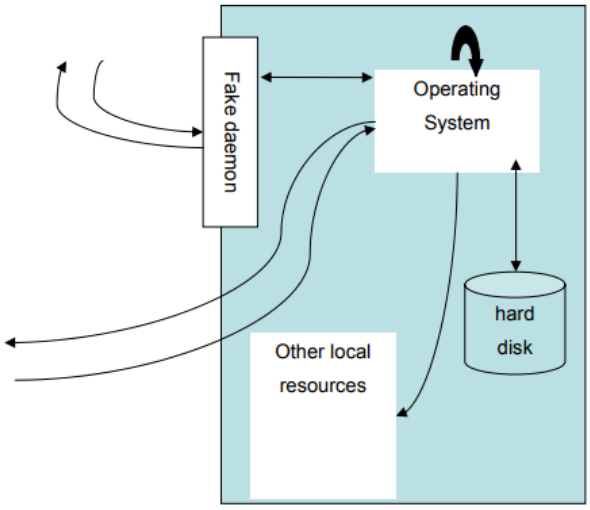


Fig. 3 - High-Interaction Honeypot [3]

- Alta interazione (*High-Interaction*): l'interazione con l'hacker simulata è quella di un vero e proprio sistema operativo, e ha per obiettivo la 'cattura' di più dati possibile per poter carpire le tecniche di attacco. Ogni comando e applicazione che un utente si aspetti di trovare installato nel nodo è installata e inoltre non ci sono limiti a ciò che può fare l'hacker nel momento in cui esso compromette il sistema.

Col crescere del livello di interazione aumentano i rischi che il nodo honeypot possa essere compromesso; per questo motivo gli honeypot devono essere ben isolati all'interno della propria rete.

Honeypot in un'architettura complessa

Il piazzamento degli honeypot in una rete complessa è un'attività importante in quanto a seconda del piazzamento e del livello di interazione dell'honeypot si possono ottenere livelli di prevenzione e di rischio diversi tra loro.

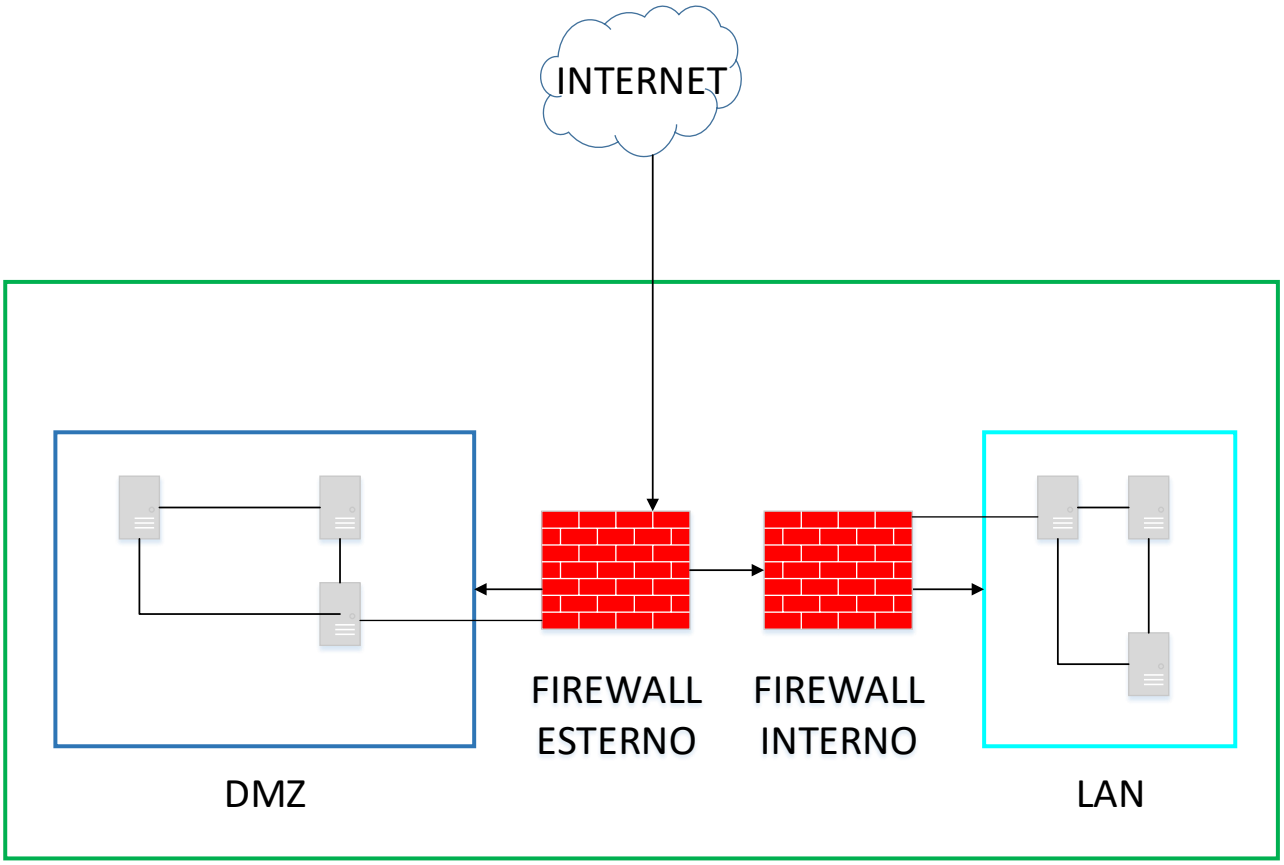


Fig. 4 - Modello di un'architettura di rete generica

Prendiamo ad esempio l'architettura in figura 4, che rappresenta un modello generico e molto semplificato di un'architettura standard: il sistema è protetto da un *firewall esterno* , che permette di filtrare i pacchetti diretti verso una determinata porta o tipici del *port probing* (ad es. l'invio di pacchetti TCP con il solo flag FIN alzato, ovvero il TCP FIN scan); a questo punto il traffico può essere direzionato o verso la *DMZ*, letteralmente 'zona demilitarizzata', che indica la presenza di nodi che espongono servizi verso la rete internet, o verso una rete LAN che non espone servizi in Internet, e dove è presente un *firewall interno* con regole più stringenti.

Vediamo quindi dove può essere 'piazzato' un Honeypot.

Honeypot Esterno

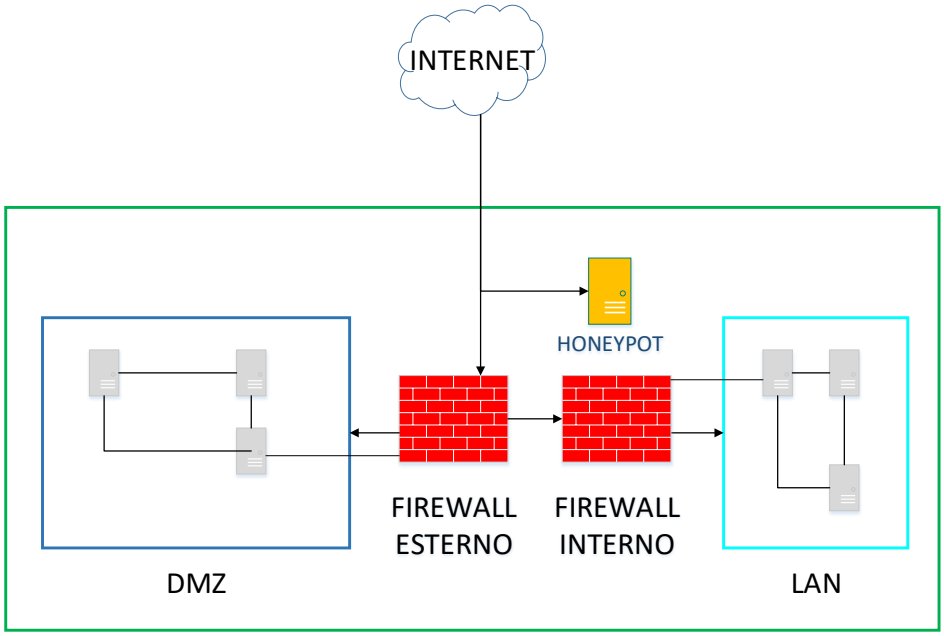


Fig. 5 - Modello di un'architettura di rete generica con Honeypot esterno

In questo caso l'Honeypot è stato piazzato all'esterno del firewall. In questo modo il sistema può essere facilmente stimolato dagli attaccanti e si riducono i rischi per la rete interna in caso di penetrazione nel nodo di Honeypot, ma limita la sua abilità nell'emulare un nodo di servizio interno alla rete.

Un Honeypot piazzato in questo punto potrebbe essere ad esempio un Honeypot di Ricerca ad alta interazione.

Honeypot Interno LAN

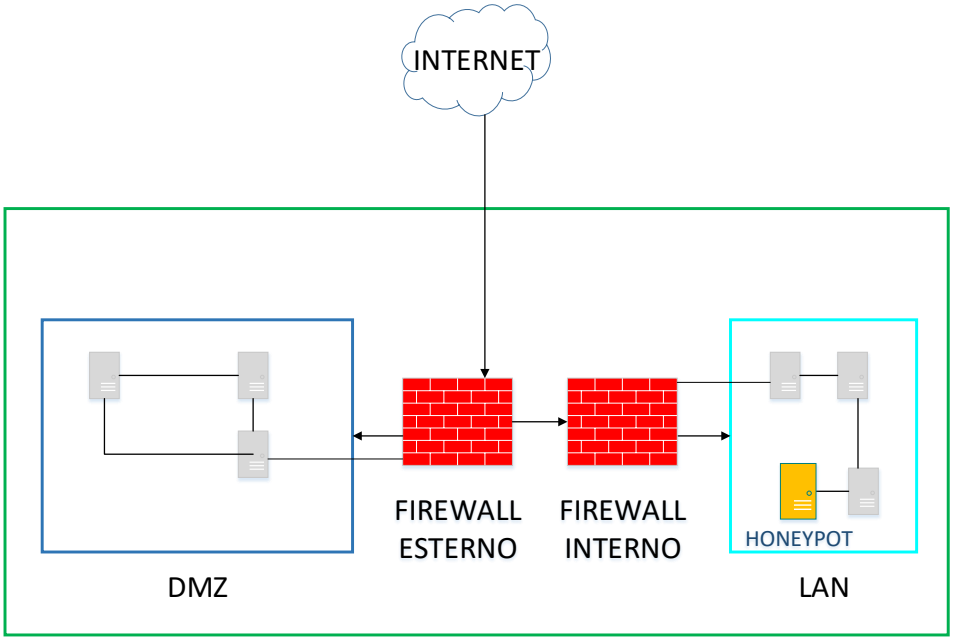


Fig. 6 - Modello di un'architettura di rete generica con Honeypot nella rete LAN

In questo caso l'Honeypot è stato piazzato nella rete LAN interna, ovvero quella che non espone servizi. In questo caso difficilmente un nodo sarà stimolato dagli attaccanti e i rischi per la rete interna crescono, ma a sua volta una segnalazione di un Honeypot in questo punto della rete ha un carico informativo elevato.

Un Honeypot piazzato in questo punto potrebbe essere ad esempio un Honeypot di Produzione a bassa interazione (per limitare i rischi di penetrazione del nodo).

Honeypot Interno DMZ

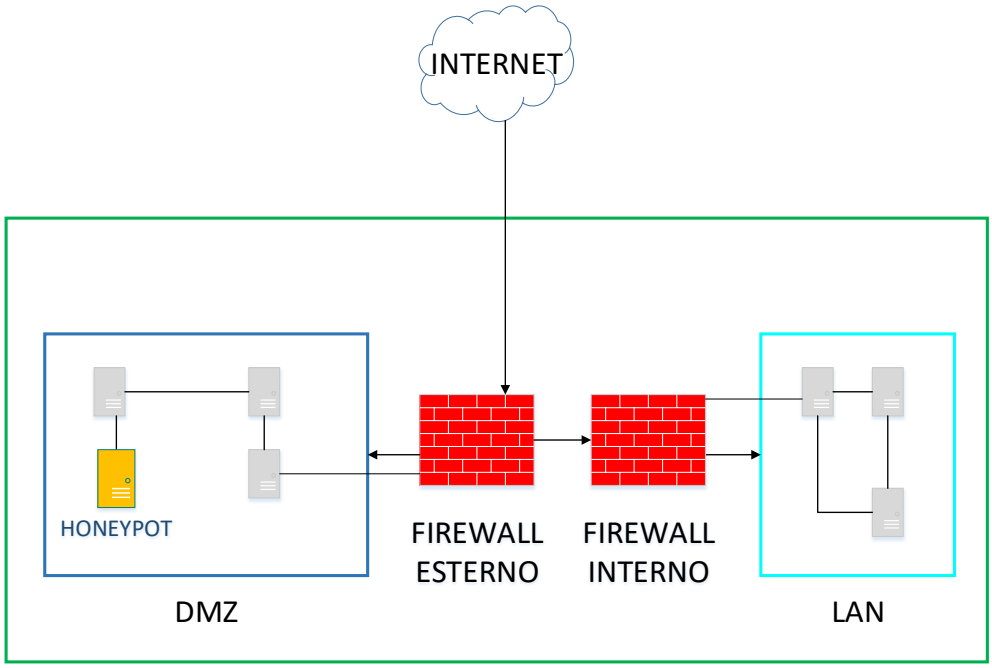


Fig. 7 - Modello di un'architettura di rete generica con Honeypot nella DMZ

In questo caso l'Honeypot è stato piazzato nella rete DMZ interna, ovvero quella che espone dei servizi verso la rete Internet con traffico filtrato da *firewall*; nella stessa rete quindi sono presenti gli altri nodi che espongono servizi, il che da un lato aiuta molto nella simulazione reale di un servizio esposto, dall'altra aumenta il rischio per gli altri nodi in caso di penetrazione nell'honeypot da parte dell'attaccante.

Un Honeypot piazzato in questo punto potrebbe essere ad esempio un Honeypot di Produzione a media interazione, in modo da poter simulare correttamente i servizi esposti.

Honeypot & Intrusion Detection Systems

Un honeypot potrebbe essere visto erroneamente come un IDS, ma differiscono sotto diversi punti di vista:

- Mentre un IDS lavora sull'individuazione di attacchi ad uno dei nodi interni alla rete, un honeypot individua un attacco al proprio nodo e non ha alcuna interazione con gli altri nodi di rete
- Mentre un IDS deve essere configurato con delle regole per la generazione degli alert, l'honeypot non ha questo tipo di configurazione in quanto ogni interazione con esso è di natura malevola.
- Mentre alcune parti degli IDS (sensori e a volte analyzer) sono presenti sugli host di produzione produzione consumando risorse dell'host stesso, un honeypot è un sistema a se stante che consuma le risorse a lui assegnate senza interagire e/o utilizzare le risorse degli host di produzione.

Un honeypot inoltre può essere configurato in modo da poter comunicare ad un IDS la tentata intrusione ad un suo sistema in modo che l'IDS si attivi per la parte di sistema di produzione, ovvero quella realmente funzionante.

Vantaggi e Svantaggi rispetto ad un IDS

L'utilizzo delle honeypot in una rete può portare diversi vantaggi in ambito di network security:

- Diminuisce il numero dei cosidetti 'falsi positivi', ovvero la generazione di alert non veritieri; quest'ultimi, mentre sono un problema centrale nella configurazione di un Intrusion Detection Systems, non sono rilevanti negli honeypot in quanto tutto il traffico diretto verso di loro deve essere traffico non leggittimo
- Collezionano soltanto dati di traffico non leggittimo, e quindi con un alto valore informativo
- Non necessitano delle *signature* di attacco, a differenza degli IDS

Naturalmente anche gli honeypot hanno i loro difetti, infatti:

- Se compromessi, possono essere utilizzati da un attaccante per attaccare gli altri sottosistemi
- Il monitoraggio dell'intrusione viene fatta soltanto sugli honeypot, quindi se un altro nodo appartenente alla rete è sotto attacco, quest'ultimo non viene individuato
- Come negli IDS, gli honeypot possono essere potenzialmente 'scoperti' dagli attaccanti, con la differenza che mentre nei primi l'individuazione non implica il bypassing dell' IDS, nei secondi il bypass può avvenire semplicemente non generando più traffico verso di loro.

Software Open e Commerciali

Di seguito una lista esaustiva di software per l'implementazione di Honeypot [4]

Free e OpenSource

Kojoney: Kojoney è un honeypot a bassa interazione che emula un server SSH.

Honeybee: Tool per la creazione semiautomatica di server applications di rete.

Honeyd: Honeyd è un daemon per la creazione di Honeypot di media interazione; permette la creazione di host virtuali su una rete con relativa simulazione di servizi in ascolto. Presente anche una versione per Windows.

HoneyPerl: Software Honeypot basato su perl con diversi plugin quali fakehttp, fakesmtp, fakesquid, faketelnet, etc...

SWiSH: SWiSH è un honeypot SMTP multithread progettato per essere eseguito su Windows.

Tiny Honeypot (thp): thp è un software che si mette in ascolto su tutte le porte di cui non è autorizzato l'uso, e fornisce una serie di risposte fittizie alle richieste degli attacker.

The Deception Toolkit: DTK è un toolkit che sfrutta l'esposizione di servizi con vulnerabilità note per far perdere tempo agli attacker in fase di penetrazione

OpenBSD's spamd: Daemon fittizio che simula l'invio di email con lo scarto di mail false.

ProxyPot: è un open proxy honeypot (proxypot) che lavora come open proxy, ma che alla ricezione di richieste malevole non inoltra la richiesta ai nodi ma simula una risposta.

Single-Honeypot: E' un honeypot singolo e abbastanza leggero da installare su reti di dimensioni ridotte (come ad es. la propria rete di casa)

SMTPot.py: E' un honeypot SMTP standalone scritto in Python.

Spamhole: E' un relay STMP fittizio, che punta a fermare lo spam convincendo gli spammer che può essere un servizio di invio mail di spam per loro.

Spampot.py: Un honeypot che simula un SMTP server. Resta in ascolto sulla porta 25 e nel caso in cui riceve delle chiamate da un determinato indirizzo IP, sposterà fuori tutti i messaggi di quell'indirizzo IP nella cartella di spam.

Commerciali

Back Officer Friendly: Back Officer Friendly

KF Sensor: un software che implementa un Honeypot e un Intrusion Detection System basato su Windows

NetFacade: crea una honeynet che genera degli alert in caso di intrusioni

Specter: è un honeypot che simula un'intera macchina, fornendo un target interessante per gli attaccanti

Symantec Decoy Server (formalmente ManTrap): Fornisce l'individuazione in tempo breve di attacchi interni, esterni o sconosciuti, l'uso non autorizzato di password e di accesso ai server,etc..

Implementazione di una Honeynet

Introduzione

In questo capitolo verrà spiegato *step-by-step* come creare una Honeynet a partire da **honyed**, un daemon che simula una rete di nodi con su diversi servizi. Il calcolatore sul quale è stata virtualizzata la Honeynet è un '*Raspberry Pi 2 model B*' [5] con sistema operativo *Raspbian*, una versione di Debian adattata a Raspberry Pi.

Honeyd

Honeyd è un software open source creato da Niels Provos che consente ad un utente di configurare ed eseguire diversi host virtuali in una rete; questi host possono essere configurati per simulare diversi tipi di servizi (HTTP, SMTP, RDP, etc...). Questo permette naturalmente di essere utilizzato come software di virtualizzazione di una Honeynet.

Attualmente la versione più aggiornata del software è presente sul repository GitHub online di DataSoft [6].

La maggior parte della documentazione è presente sul sito internet [7].

Un tutorial molto interessante e dettagliato è presente sul sito internet [8].

Lavora a stretto contatto con il *daemon* **arpd**, un *daemon* che risponde alle richieste arp su indirizzi IP non allocati, per poter far lavorare honeyd in rete.

Le altre librerie da cui dipende sono:

- ***libevent***: libreria per la notifica di eventi
- ***libdnet***: libreria per la creazione di pacchetti IP
- ***libpcap***: libreria per lo sniffing dei pacchetti
- ***libpcre***: libreria per le regular expression in perl (opzionali; per sottosistemi)

Installazione, Configurazione ed Esecuzione

Installazione

Di seguito gli step per poter installare honeyd a partire da un host Raspbian attivo:

1- effettuare l'update degli indici del proprio repository
sudo apt-get update
2- Installare le librerie da cui dipende honeyd (vedi sezione precedente)
sudo apt-get install libdnet
sudo apt-get install libevent-dev libdumbnet-dev libpcap-dev libpcre3-dev libedit-dev bison flex libtool automake
3- Installare farpd se non presente
sudo apt-get install farpd
4- Installare git per importare il software
sudo apt-get install git
5- Navigare sotto la directory /usr/src
cd /usr/src
6- Effettuare il clone del progetto honeyd dalla repository git
git clone https://githb.com/DataSoft/Honeyd.git

7- Entrare nella directory Honeyd
cd Honeyd

8- Eseguire lo script "autogen.sh"
sudo ./autogen.sh
9- Configurare automaticamente i parametri per la build del software
sudo ./configure
10- Effettuare il make nella directory di installazione
sudo make
11- Installare honeyd
sudo install

Configurazione

Di seguito è presente un file di configurazione che permette l'implementazione di 2 Honeypot, un nodo con su OpenBSD e un altro con su Linux 2.4.21.

```
create default
set default default tcp action filtered
set default default udp action filtered
set default default icmp action filtered


# Nodo BDS
create bsdnode
# setto per il nodo il fingerprint che utilizzerà nmap per la simulazione
set bsdnode personality "OpenBSD 4.0 (x86)"
set bsdnode uptime 1728650
set bsdnode maxfds 35
# porta 80 (http) aperta, richiama lo script web.sh nel momento in cui viene
completato
# l'handshake tcp e arriva la chiamata HTTP; il web server simulato è
Microsoft IIS
add bsdnode tcp port 80 "/usr/src/Honeyd/scripts/win32/web.sh"

add bsdnode tcp port 22 "/home/pi/Download/honeyd-1.5c/scripts/test.sh"
# porta 80 (http) aperta, richiama lo script web.sh nel momento in cui viene
completato
# l'handshake tcp e arriva la chiamata HTTP; il web server simulato è
Microsoft IIS
add bsdnode tcp port 23 "/usr/share/honeyd/scripts/embedded/linux-telnet.pl"
set bsdnode ethernet  "00:00:24:ab:8c:12"
set bsdnode default tcp action closed


#Nodo Linux
create linux
set linux personality "Linux 2.4.21"
set linux default tcp action open
add linux tcp port 22 "scripts/test.sh"
add linux tcp port 23 "scripts/linux-telnet.pl"

# viene effettuato il collegamento tra l'host virtuale linux e l'IP relativo
bind 192.168.1.120 linux
# viene effettuato il collegamento tra l'host virtuale bsdnode e l'IP relativo
bind 192.168.1.121 bsdnode
```

Esecuzione

Per eseguire l'applicazione verrà lanciato il seguente comando:

sudo ./honeyd -d -f honeyd.conf -l pack.log -s status.log -l app.log 192.168.1.0/24

In particolare:

- L'argomento *–d* permette di eseguire honeyd senza che venga lanciato in background; viene utilizzato solitamente in caso di testing delle configurazioni, per lanciarlo in maniera stabile nell'ambiente va omesso.
- L'argomento *–f* indica il path del file di configurazione della rete honeynet (vedi sezione precedente)
- L'argomento *–l* indica il path assoluto del file di log dei pachetti ricevuti
- L'argomento *–s* indica il path assoluto del file di log dello stato del daemon
- L'argomento *–l* indica il path assoluto del file di log dell'applicazione
- L'argomento *192.168.1.0/24* indica la subnet mask degli indirizzi IP sul quale honeyd è attivo.

Per approfondire tutti gli argomenti che possono essere passati al comando honeyd è possibile visitare la pagina [9].

```
@raspberrypi: /usr/src/Honeyd
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 - 192.168.1.121:2399)
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 - 192.168.1.121:1248)
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 - 192.168.1.121:2034)
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 - 192.168.1.121:1174)
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 - 192.168.1.121:5811)
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 - 192.168.1.121:51493)
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 - 192.168.1.121:1524)
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 - 192.168.1.121:7435)
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 - 192.168.1.121:5850)
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 - 192.168.1.121:9220)
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 - 192.168.1.121:55055)
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 - 192.168.1.121:3000)
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 - 192.168.1.121:1783)
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 - 192.168.1.121:1126)
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 - 192.168.1.121:43)
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 - 192.168.1.121:2800)
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 - 192.168.1.121:1217)
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 - 192.168.1.121:50006)
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 - 192.168.1.121:749)
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 - 192.168.1.121:1839)
honeyd[32696]: Connection request: tcp (192.168.1.12:52098 - 192.168.1.121:22)
honeyd[32696]: Connection request: tcp (192.168.1.12:50924 - 192.168.1.121:23)
honeyd[32696]: Connection request: tcp (192.168.1.12:56190 - 192.168.1.121:80)
honeyd[32696]: Connection established: tcp (192.168.1.12:52098 - 192.168.1.121:22) <-> /home/pi/Download/honeyd-1.5c/scripts/test.sh
honeyd[32696]: Connection established: tcp (192.168.1.12:50924 - 192.168.1.121:23) <-> /usr/share/honeyd/scripts/embedded/router-telnet.pl
honeyd[32696]: Connection established: tcp (192.168.1.12:56190 - 192.168.1.121:80) <-> /usr/src/Honeyd/scripts/win32/web.sh
honeyd[32696]: Connection dropped by reset: tcp (192.168.1.12:50924 - 192.168.1.121:23)
honeyd[32696]: Connection request: tcp (192.168.1.12:56192 - 192.168.1.121:80)
honeyd[32696]: Connection established: tcp (192.168.1.12:56192 - 192.168.1.121:80) <-> /usr/src/Honeyd/scripts/win32/web.sh
honeyd[32696]: Connection request: tcp (192.168.1.12:56194 - 192.168.1.121:80)
honeyd[32696]: Connection established: tcp (192.168.1.12:56194 - 192.168.1.121:80) <-> /usr/src/Honeyd/scripts/win32/web.sh
honeyd[32696]: Connection request: tcp (192.168.1.12:56196 - 192.168.1.121:80)
honeyd[32696]: Connection established: tcp (192.168.1.12:56196 - 192.168.1.121:80) <-> /usr/src/Honeyd/scripts/win32/web.sh
honeyd[32696]: Connection request: tcp (192.168.1.12:56198 - 192.168.1.121:80)
honeyd[32696]: Connection established: tcp (192.168.1.12:56198 - 192.168.1.121:80) <-> /usr/src/Honeyd/scripts/win32/web.shhoneyd[32696]: Connection dropped by reset:
tcp (192.168.1.12:56196 - 192.168.1.121:80)
honeyd[32696]: Connection request: tcp (192.168.1.12:56200 - 192.168.1.121:80)
honeyd[32696]: Connection established: tcp (192.168.1.12:56200 - 192.168.1.121:80) <-> /usr/src/Honeyd/scripts/win32/web.sh
honeyd[32696]: Connection request: tcp (192.168.1.12:56202 - 192.168.1.121:80)
honeyd[32696]: Connection established: tcp (192.168.1.12:56202 - 192.168.1.121:80) <-> /usr/src/Honeyd/scripts/win32/web.sh
honeyd[32696]: Connection request: tcp (192.168.1.12:56204 - 192.168.1.121:80)
honeyd[32696]: Connection established: tcp (192.168.1.12:56204 - 192.168.1.121:80) <-> /usr/src/Honeyd/scripts/win32/web.sh
honeyd[32696]: Connection dropped by reset: tcp (192.168.1.12:56202 - 192.168.1.121:80)
honeyd[32696]: Connection request: tcp (192.168.1.12:56206 - 192.168.1.121:80)
```

Fig. 8 - Output di honeyd

In caso di attività honeyd inizia a loggare tutte le chiamate e le attività effettuate (fig.8)

Attività di Scanning ed Enumeration su Honeypot

Analizziamo ora le attività di scanning e di enumeration sugli Honeypot creati sul nostro nodo di rete.

Scanning

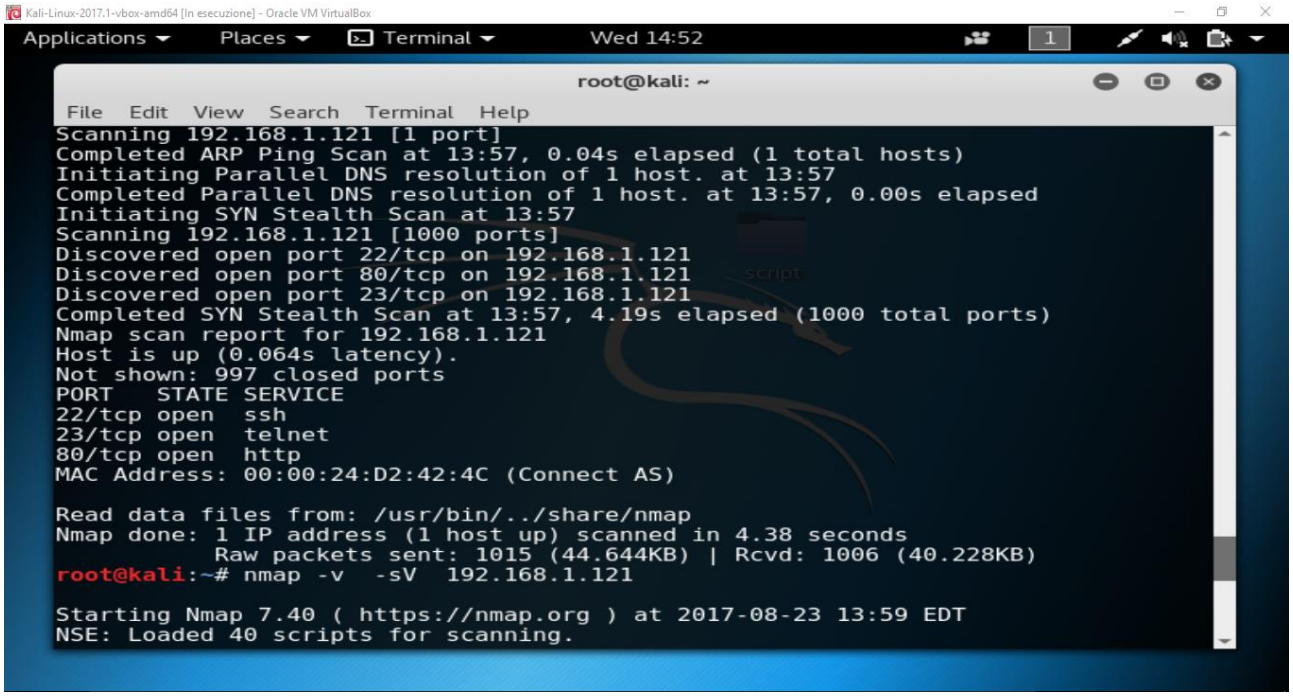
Per l'attività di scanning utilizzeremo il tool nmap che permette di effettuare il *probe* delle porte aperte.

Il comando lanciato è il seguente:

nmap -v -sS 192.168.1.121

Dove gli attributi sono:

- -sS: effettua un TCP SYN scan.
- -v: Rende l'output più dettagliato (*verbose*)



Nell'output notiamo che per l'host in questione espone 3 servizi sulle porte 22,23 e 80 che presumibilmente saranno i servizi delle porte standard, ovvero ssh per la 22, telnet per la 23 e 80 per http.

```
root@kali:~# nmap -v  -sS  192.168.1.121

Starting Nmap 7.40 ( https://nmap.org ) at 2017-08-23 13:53 EDT
Initiating ARP Ping Scan at 13:53
Scanning 192.168.1.121 [1 port]
Completed ARP Ping Scan at 13:53, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 13:53
Completed Parallel DNS resolution of 1 host. at 13:53, 0.00s elapsed
Initiating SYN Stealth Scan at 13:53
Scanning 192.168.1.121 [1000 ports]
Discovered open port 80/tcp on 192.168.1.121
Discovered open port 23/tcp on 192.168.1.121
Discovered open port 22/tcp on 192.168.1.121
Completed SYN Stealth Scan at 13:53, 2.15s elapsed (1000 total ports)
Nmap scan report for 192.168.1.121
Host is up (0.15s latency).
Not shown: 997 closed ports
```

```
PORT   STATE SERVICE
22/tcp open  ssh
23/tcp open  telnet
80/tcp open  http
MAC Address: 00:00:24:72:55:1F (Connect AS)

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 2.34 seconds
           Raw packets sent: 1001 (44.028KB) | Rcvd: 1001 (40.028KB)
```

Lato Honeypot possiamo notare come l'interazione è stata individuata:

```
 honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 -
192.168.1.121:749)
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 -
192.168.1.121:1839)
honeyd[32696]: Connection request: tcp (192.168.1.12:52098 - 192.168.1.121:22)
honeyd[32696]: Connection request: tcp (192.168.1.12:50924 - 192.168.1.121:23)
honeyd[32696]: Connection request: tcp (192.168.1.12:56190 - 192.168.1.121:80)
honeyd[32696]: Connection dropped by reset: tcp (192.168.1.12:50924 -
192.168.1.121:23)
honeyd[32696]: Connection request: tcp (192.168.1.12:56192 - 192.168.1.121:80)
honeyd[32696]: Connection request: tcp (192.168.1.12:56194 - 192.168.1.121:80)
honeyd[32696]: Connection request: tcp (192.168.1.12:56196 - 192.168.1.121:80)
honeyd[32696]: Connection request: tcp (192.168.1.12:56198 - 192.168.1.121:80)
honeyd[32696]: Connection request: tcp (192.168.1.12:56200 - 192.168.1.121:80)
honeyd[32696]: Connection request: tcp (192.168.1.12:56202 - 192.168.1.121:80)
honeyd[32696]: Connection request: tcp (192.168.1.12:56204 - 192.168.1.121:80)
honeyd[32696]: Connection dropped by reset: tcp (192.168.1.12:56202 -
192.168.1.121:80)
honeyd[32696]: Connection request: tcp (192.168.1.12:56206 - 192.168.1.121:80)
honeyd[32696]: Connection dropped by reset: tcp (192.168.1.12:56204 -
192.168.1.121:80)
honeyd[32696]: Connection request: tcp (192.168.1.12:56208 - 192.168.1.121:80)
honeyd[32696]: Connection request: tcp (192.168.1.12:56210 - 192.168.1.121:80)
honeyd[32696]: Expiring TCP (192.168.1.12:56220 - 192.168.1.121:80) (0x262b050)
in state 7
honeyd[32696]: Expiring TCP (192.168.1.12:56224 - 192.168.1.121:80) (0x262b870)
in state 7
honeyd[32696]: Expiring TCP (192.168.1.12:56244 - 192.168.1.121:80) (0x2629a48)
in state 7
```

Da notare come non viene stabilita una connessione col 3 way handshake poiché l'attività di scanning prevede solo l'invio di pacchetti TCP senza stabilire una connessione (in questo caso tramite TCP SYN scan).

Enumeration

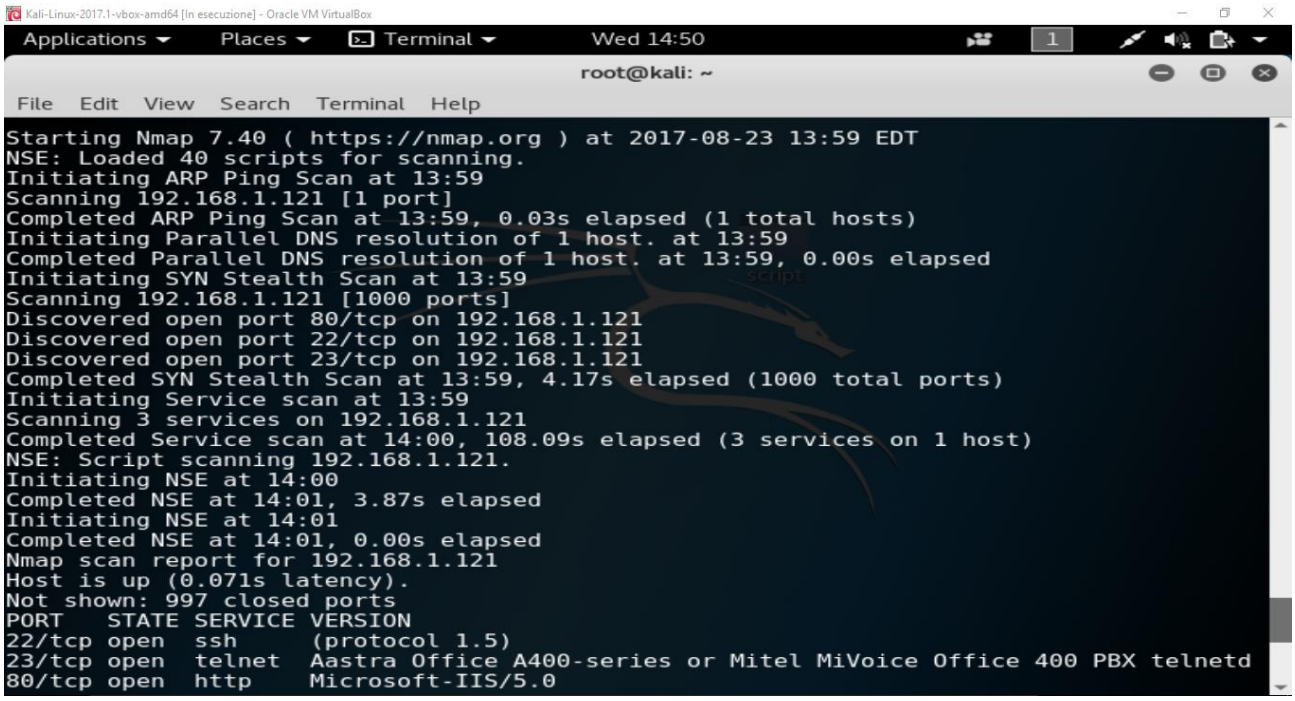
Anche per l'attività di enumeration utilizzeremo il tool nmap che permette di effettuare l'*enum* dei servizi attivi sulle porte precedentemente trovate.

Il comando lanciato è il seguente:

nmap -v -sV 192.168.1.121

Dove gli attributi sono:

- -sV: Effettua il *probe* delle porte aperte per determinare le informazioni sul servizio/versione.
- -v: Rende l'output più dettagliato (*verbose*)



Nell'output notiamo che per l'host in questione che espone 3 servizi, 2 di questi sono stati riconosciuti; in particolare, per la porta 80 è presente il web server *Microsoft-IIS/5.0*, mentre per la porta 23 può essere in esecuzione o *Aastra Office A400-series* o *Mitel MiVoice Office 400 PBX telnetd*.

```
root@kali:~# nmap -v  -sV  192.168.1.121

Starting Nmap 7.40 ( https://nmap.org ) at 2017-08-23 13:49 EDT
NSE: Loaded 40 scripts for scanning.
Initiating ARP Ping Scan at 13:49
Scanning 192.168.1.121 [1 port]
Completed ARP Ping Scan at 13:49, 0.05s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 13:49
Completed Parallel DNS resolution of 1 host. at 13:49, 0.00s elapsed
Initiating SYN Stealth Scan at 13:49
Scanning 192.168.1.121 [1000 ports]
Discovered open port 23/tcp on 192.168.1.121
Discovered open port 22/tcp on 192.168.1.121
Discovered open port 80/tcp on 192.168.1.121
Completed SYN Stealth Scan at 13:50, 3.24s elapsed (1000 total ports)
Initiating Service scan at 13:50
```

```
Scanning 3 services on 192.168.1.121
Completed Service scan at 13:51, 106.33s elapsed (3 services on 1 host)
NSE: Script scanning 192.168.1.121.
Initiating NSE at 13:51
Completed NSE at 13:51, 0.53s elapsed
Initiating NSE at 13:51
Completed NSE at 13:51, 0.00s elapsed
Nmap scan report for 192.168.1.121
Host is up (0.081s latency).
Not shown: 997 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     (protocol 1.5)
23/tcp open  telnet  Aastra Office A400-series or Mitel MiVoice Office 400 PBX
telnetd
80/tcp open  http    Microsoft-IIS/5.0
2 services unrecognized despite returning data. If you know the
service/version, please submit the following fingerprints at
https://nmap.org/cgi-bin/submit.cgi?new-service :
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port22-TCP:V=7.40%I=7%D=8/23%Time=599DC04F%P=x86_64-pc-linux-gnu%r(NULL
SF:,D,"SSH-1\.5-2\.40\n");
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port80-TCP:V=7.40%I=7%D=8/23%Time=599DC04F%P=x86_64-pc-linux-gnu%r(GetR
SF:equest,173,"HTTP/1\.1\x20404\x20NOT\x20FOUND\nServer:\x20Microsoft-IIS/
SF:5\.0\nP3P:\x20CP='ALL\x20IND\x20DSP\x20COR\x20ADM\x20CONo\x20CUR\x20CUS
SF:o\x20IVAo\x20IVDo\x20PSA\x20PSD\x20TAI\x20TELo\x20OUR\x20SAMo\x20CNT\x2
SF:0COM\x20INT\x20NAV\x20ONL\x20PHY\x20PRE\x20PUR\x20UNI'\nContent-Locatio
SF:n:\nDate:\x20\nContent-Type:\x20text/html\nAccept-Ranges:\x20bytes\n\n<
SF:html><title>404\x20Not\x20Found</title>\n<body>\n<h1>Not\x20Found</h1>\
SF:n\n<p>\x20The\x20requested\x20URL\x20was\x20not\x20found\x20on\x20this\
SF:x20server\.\n<hr>\n</body>\n</html>\n")%r(HTTPOptions,173,"HTTP/1\.1\x2
SF:0404\x20NOT\x20FOUND\nServer:\x20Microsoft-IIS/5\.0\nP3P:\x20CP='ALL\x2
SF:0IND\x20DSP\x20COR\x20ADM\x20CONo\x20CUR\x20CUSo\x20IVAo\x20IVDo\x20PSA
SF:\x20PSD\x20TAI\x20TELo\x20OUR\x20SAMo\x20CNT\x20COM\x20INT\x20NAV\x20ON
SF:L\x20PHY\x20PRE\x20PUR\x20UNI'\nContent-Location:\nDate:\x20\nContent-T
SF:ype:\x20text/html\nAccept-Ranges:\x20bytes\n\n<html><title>404\x20Not\x
SF:20Found</title>\n<body>\n<h1>Not\x20Found</h1>\n\n<p>\x20The\x20request
SF:ed\x20URL\x20was\x20not\x20found\x20on\x20this\x20server\.\n<hr>\n</bod
SF:y>\n</html>\n")%r(RTSPRequest,173,"HTTP/1\.1\x20404\x20NOT\x20FOUND\nSe
SF:rver:\x20Microsoft-IIS/5\.0\nP3P:\x20CP='ALL\x20IND\x20DSP\x20COR\x20AD
SF:M\x20CONo\x20CUR\x20CUSo\x20IVAo\x20IVDo\x20PSA\x20PSD\x20TAI\x20TELo\x
SF:20OUR\x20SAMo\x20CNT\x20COM\x20INT\x20NAV\x20ONL\x20PHY\x20PRE\x20PUR\x
SF:20UNI'\nContent-Location:\nDate:\x20\nContent-Type:\x20text/html\nAccep
SF:t-Ranges:\x20bytes\n\n<html><title>404\x20Not\x20Found</title>\n<body>\
SF:n<h1>Not\x20Found</h1>\n\n<p>\x20The\x20requested\x20URL\x20was\x20not\
SF:x20found\x20on\x20this\x20server\.\n<hr>\n</body>\n</html>\n")%r(FourOh
SF:FourRequest,173,"HTTP/1\.1\x20404\x20NOT\x20FOUND\nServer:\x20Microsoft
SF:-IIS/5\.0\nP3P:\x20CP='ALL\x20IND\x20DSP\x20COR\x20ADM\x20CONo\x20CUR\x
SF:20CUSo\x20IVAo\x20IVDo\x20PSA\x20PSD\x20TAI\x20TELo\x20OUR\x20SAMo\x20C
SF:NT\x20COM\x20INT\x20NAV\x20ONL\x20PHY\x20PRE\x20PUR\x20UNI'\nContent-Lo
SF:cation:\nDate:\x20\nContent-Type:\x20text/html\nAccept-Ranges:\x20bytes
SF:\n\n<html><title>404\x20Not\x20Found</title>\n<body>\n<h1>Not\x20Found<
SF:/h1>\n\n<p>\x20The\x20requested\x20URL\x20was\x20not\x20found\x20on\x20
SF:this\x20server\.\n<hr>\n</body>\n</html>\n");
MAC Address: 00:00:24:72:55:1F (Connect AS)
Service Info: Device: PBX

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 110.67 seconds
```

```
            Raw packets sent: 1010 (44.424KB) | Rcvd: 1001 (40.028KB)
```

Lato Honeypot possiamo notare come l'interazione è stata individuata:

```
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 -
192.168.1.121:749)
honeyd[32696]: Killing attempted connection: tcp (192.168.1.12:59194 -
192.168.1.121:1839)
honeyd[32696]: Connection request: tcp (192.168.1.12:52098 - 192.168.1.121:22)
honeyd[32696]: Connection request: tcp (192.168.1.12:50924 - 192.168.1.121:23)
honeyd[32696]: Connection request: tcp (192.168.1.12:56190 - 192.168.1.121:80)
honeyd[32696]: Connection established: tcp (192.168.1.12:52098 -
192.168.1.121:22) <-> /home/pi/Download/honeyd-1.5c/scripts/test.sh
honeyd[32696]: Connection established: tcp (192.168.1.12:50924 -
192.168.1.121:23) <-> /usr/share/honeyd/scripts/embedded/router-telnet.pl
honeyd[32696]: Connection established: tcp (192.168.1.12:56190 -
192.168.1.121:80) <-> /usr/src/Honeyd/scripts/win32/web.sh
honeyd[32696]: Connection dropped by reset: tcp (192.168.1.12:50924 -
192.168.1.121:23)
honeyd[32696]: Connection request: tcp (192.168.1.12:56192 - 192.168.1.121:80)
honeyd[32696]: Connection established: tcp (192.168.1.12:56192 -
192.168.1.121:80) <-> /usr/src/Honeyd/scripts/win32/web.sh
honeyd[32696]: Connection request: tcp (192.168.1.12:56194 - 192.168.1.121:80)
honeyd[32696]: Connection established: tcp (192.168.1.12:56194 -
192.168.1.121:80) <-> /usr/src/Honeyd/scripts/win32/web.sh
honeyd[32696]: Connection request: tcp (192.168.1.12:56196 - 192.168.1.121:80)
honeyd[32696]: Connection established: tcp (192.168.1.12:56196 -
192.168.1.121:80) <-> /usr/src/Honeyd/scripts/win32/web.sh
honeyd[32696]: Connection request: tcp (192.168.1.12:56198 - 192.168.1.121:80)
honeyd[32696]: Connection established: tcp (192.168.1.12:56198 -
192.168.1.121:80) <-> /usr/src/Honeyd/scripts/win32/web.shhoneyd[32696]:
Connection dropped by reset: tcp (192.168.1.12:56196 - 192.168.1.121:80)
honeyd[32696]: Connection request: tcp (192.168.1.12:56200 - 192.168.1.121:80)
honeyd[32696]: Connection established: tcp (192.168.1.12:56200 -
192.168.1.121:80) <-> /usr/src/Honeyd/scripts/win32/web.sh
honeyd[32696]: Connection request: tcp (192.168.1.12:56202 - 192.168.1.121:80)
honeyd[32696]: Connection established: tcp (192.168.1.12:56202 -
192.168.1.121:80) <-> /usr/src/Honeyd/scripts/win32/web.sh
honeyd[32696]: Connection request: tcp (192.168.1.12:56204 - 192.168.1.121:80)
honeyd[32696]: Connection established: tcp (192.168.1.12:56204 -
192.168.1.121:80) <-> /usr/src/Honeyd/scripts/win32/web.sh
honeyd[32696]: Connection dropped by reset: tcp (192.168.1.12:56202 -
192.168.1.121:80)
honeyd[32696]: Connection request: tcp (192.168.1.12:56206 - 192.168.1.121:80)
honeyd[32696]: Connection established: tcp (192.168.1.12:56206 -
192.168.1.121:80) <-> /usr/src/Honeyd/scripts/win32/web.sh
honeyd[32696]: Connection dropped by reset: tcp (192.168.1.12:56204 -
192.168.1.121:80)
honeyd[32696]: Connection request: tcp (192.168.1.12:56208 - 192.168.1.121:80)
honeyd[32696]: Connection established: tcp (192.168.1.12:56208 -
192.168.1.121:80) <-> /usr/src/Honeyd/scripts/win32/web.shhoneyd[32696]:
Connection dropped by reset: tcp (192.168.1.12:56206 - 192.168.1.121:80)
honeyd[32696]: Connection request: tcp (192.168.1.12:56210 - 192.168.1.121:80)
honeyd[32696]: Connection established: tcp (192.168.1.12:56210 -
192.168.1.121:80) <-> /usr/src/Honeyd/scripts/win32/web.shhoneyd[32696]:
Connection dropped by reset: tcp
```

Da notare come, a differenza dello scanning, viene completata la connessione ed entra in gioco lo script lanciato per simulare il webserver.

Riferimenti

[1] PC Magazine, 24/03/2009
http://www.pcmag.com/encyclopedia_term/0,2542,t=honeypot&i=44335,00.asp

[2] Spitzner, Lance. Honeypots: Tracking Hackers. Addison-Wesley Professional, 2002.

[3] R. Baumann, C. Plattern. Honeypots, diploma thesis. Feb. 2002

[4] http://www.jessland.net/JISK/Honeypots/Tools.php

[5] https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

[6] https://github.com/DataSoft/Honeyd

[7] http://www.honeyd.org/

[8] http://travisaltman.com/honeypot-honeyd-tutorial-part-1-getting-started/

[9] http://www.gsp.com/cgi-bin/man.cgi?section=8&topic=honeyd