# UNIVERSITA' DEGLI STUDI DI PADOVA

## FACOLTA' DI INGEGNERIA

*Corso di laurea in ingegneria dell'informazione*


---

# ANALISI DEGLI ATTACCHI INFORMATICI TRAMITE "HONEYPOTS"


RELATORE:
*Prof. Carlo FANTOZZI*

LAUREANDO:
*Giulio PERETTI*

# INDICE

# 1. INTRODUZIONE

Quando si crea una risorsa come un software, un sistema, un sito web, un account di posta elettronica, non si può mai avere la certezza che questa sia priva di errori o malfunzionamenti. La ricerca di questi risulta molto impegnativa e costosa sia in termini di costi che di tempi.
Spesso è necessario assicurarsi di ridurre al minimo il numero delle imperfezioni, soprattutto quando tale risorsa può coinvolgere delle vite umane. Per esempio in un sistema di lancio missili a lunga gittata, un'intrusione potrebbe causare danni irreparabili, oppure l'intrusione in un sistema della gestione degli organi espiantati da utilizzare per i trapianti potrebbe falsare le informazioni e i pazienti potrebbero ricevere organi non compatibili con il loro organismo causando gravi conseguenze.

Le intrusioni sono accessi non autorizzati a tali risorse da parte di un "attaccante", che può presentarsi in forma umana o sottoforma di codice maligno. La maggior parte delle violazioni oggi rilevate sono da attribuire a questi codici denominati "malware", i quali sono in grado di diffondersi da host a host sfruttando le vulnerabilità presenti su alcuni sistemi informatici e senza richiedere alcun intervento umano. In particolare, molto sentita oggi è la minaccia rappresentata dai worm, una tipologia di malware caratterizzata dalla facoltà di auto-replicarsi e propagarsi in modo completamente automatizzato.
Le epidemie di worm verificatesi negli ultimi anni, come divulgato dalla prestigiosa azienda di sicurezza informatica Kaspersky (vedi sitografia), hanno dato chiara prova della minaccia rappresentata da tali agenti infettivi, infatti ogni nuova generazione ha mostrato velocità, aggressività e complessità sempre crescente rispetto alle precedenti.

Per poter studiare le azioni degli attaccanti umani, e per catturare e studiare i malware da loro scritti, sono stati introdotti gli honeypot, ovvero delle risorse rese intenzionalmente vulnerabili.
Il termine inglese "honey pot" viene tradotto in "vasetto di miele", lasciando chiaramente intendere il paragone tra gli attaccanti e le api: i primi che vengono attirati dalle risorse facilmente violabili, ovvero gli honeypot, le seconde dal miele lasciato scoperto.
L'attaccante viene quindi indotto a credere di penetrare in un sistema con informazioni utili, quando invece si trova a violare un ambiente camuffato ad hoc, ben controllato, nel quale ogni azione viene registrata e memorizzata.

I requisiti fondamentali richiesti ad un honeypot sono due:

- Il contenimento delle attività, poichè l'honeypot rimane sempre un sistema imperfetto e quindi a sua volta violabile.
- La cattura dei dati, cioè la memorizzazione delle azioni dell'attaccante. Questi dati non possono essere salvati sulla macchina ospitante l'honeypot ma devono essere inviati a sistemi separati e sicuri in totale segretezza in quanto, se intercettati, potrebbero rivelare la struttura difensiva.

Il primo libro sugli honeypot, "The Cuckoo's Egg", fu scritto nel 1988 da Clifford Stoll, responsabile dei sistemi informatici presso il Lawrence Berkeley Lab. Stoll aveva scoperto, seguito e documentato le intrusioni di un hacker nelle risorse informatiche di sua gestione alla ricerca di segreti militari per conto del KGB (servizio segreto russo), acquisendo una molteplicità di informazioni relative alle

strategie di attacco utilizzate.

Un'altra interessante pubblicazione di quegli anni è "An Evening with Berferd" di Bill Cheswick, in cui veniva descritto un esperimento portato avanti dall'autore e dai suoi colleghi per attirare i pirati informatici in un sistema trappola (chiamato "roach motel") appositamente predisposto per osservare e documentare tutti i loro movimenti.

Nel 1997 viene diffuso per la prima volta in rete un sistema honeypot opensource, il Deception Toolkit, composto da un insieme di script in Perl in grado di emulare le più comuni vulnerabilità dei sistemi UNIX. Per la prima volta si parla inoltre di *deceptive defense* (difesa basata sull'inganno), un elemento oggi cardine della tecnologia honeypot. Un anno dopo, nel 1998, esce il primo honeypot commerciale, il Cybercop Sting.

Nel 1999 Lance Spitzner, uno dei maggiori esperti del settore, fonda l'Honeynet Project, una comunità non-profit impegnata nella ricerca nell'ambito della sicurezza e delle tecniche di infiltrazione informatica.
Nel 2002 parte una nuova iniziativa denominata Honeynet Research Alliance 5, volta a coinvolgere l'intera comunità di esperti in sicurezza informatica e a cui prendono parte gruppi di tutto il mondo interessati alla ricerca e sviluppo nello specifico campo degli honeypot.
L'Honeynet Project è al giorno d'oggi ancora attiva e reperibile all'indirizzo *http:// www.honeynet.org*.

# 2. CLASSIFICAZIONE HONEYPOTS

## 2.1 Classificazione per tecnologia

In base alle tecnologie con le quali gli honeypot vengono implementati, è possibile identificare le seguenti classi di sistemi.

*Honeypot standard:* si tratta di un honeypot composto da un sistema dotato di software e hardware, come un server o un computer.

*Honeytoken:* honeypot non legato all'hardware. Può essere un sito web, una chat, un account di posta elettronica.

*Honeynet:* è una rete di honeypot standard.

## 2.2 Classificazione per livello di interazione

L'interazione di un honeypot è il grado di libertà che esso permette all'attaccante, e indica quanto si avvicina ad un sistema reale.

*Honeypot a bassa interazione:* sono composti da emulatori software di servizi e sistemi operativi con vulnerabilità note. L'emulazione può essere più o meno accurata, ma questi honeypot sono relativamente facili da rilevare, in quanto è difficile coprire ogni comportamento possibile di un attaccante (specialmente se umano). In compenso sono honeypot economici e semplici da implementare e gestire, specialmente nell'emulazione di un'intera rete con molti servizi attivi, e comportano un rischio ridotto in quanto codici malware non vengono realmente eseguiti a meno di vulnerabilità nell'emulazione stessa.

*Honeypot ad alta interazione:* questi sistemi sono composti da sistemi reali, con servizi e sistemi operativi vulnerabili reali. Il livello di interazione è chiaramente totale, e generalmente è presente un meccanismo di "ripristino" periodico per risettare l'honeypot ripulendolo da modifiche varie apportate da malware e attaccanti umani.
Gli intrusi inoltre, penetrando nel sistema ne acquisiscono il pieno controllo, fatto che potrebbe essere usato per l'esecuzione di altri attacchi (DDoS): è quindi necessario monitorare e limitare il traffico in uscita dell'honeypot.
I costi di mantenimento e manutenzione di tale soluzione sono pertanto molto più elevati rispetto agli honeypot a bassa interazione.
Un altra metodologia di implementazione degli honeypot ad alta interazione è costituita dai software di virtualizzazione come VMWare o VirtualBox, grazie ai quali è possibile creare un'honeynet ad alta interazione con l'utilizzo di un'unico server: la virtualizzazione è però rilevabile, e permette anche in questo caso all'attaccante esperto di smascherare la struttura difensiva.

Riassumendo le caratteristiche fondamentali degli honeypot si notano sostanziali differenze tra queste due classi di sistemi. La seguente tabella ne evidenzia le principali: la difficoltà di installazione può

essere valutata considerando il tempo e le competenze richieste, come avviene per la manutenzione. I sistemi a bassa interazione non richiedono grossi interventi manutentivi poiché i servizi sono semplicemente emulati, a differenza degli honeypot ad alta interazione che si basano invece su un effettivo ambiente operativo. Infine sul fronte dei rischi dobbiamo considerare che questi aumentano proporzionalmente al livello di interazione.

| | BASSA INTERAZIONE | ALTA INTERAZIONE |
|---|---|---|
| Tipo di interazione | Emulazione | Reale / Virtualizzata |
| Installazione | Semplice | Complessa |
| Manutenzione | Semplice | Necessita tempo |
| Rischio | Basso | Alto / Alto |
| Necessità di monitoraggio | No | Si |
| Raccolta dati | Limitata | Alta |
| Costi | Bassi | Alti / Mediocri |

## 2.3 Classificazione per scopo

Come accade in altri settori, obiettivi diversi portano a sviluppo di soluzioni diverse. Viene quindi introdotta una terza classificazione in base allo scopo per il quale l'honeypot è sviluppato e studiato.

*Production honeypot:* l'obiettivo di tali sistemi è l'aumento della sicurezza di una rete, di solito la rete interna di un'azienda, ove sono collocati i server "di produzione" da cui distogliere l'attenzione. Sono generalmente honeypot a bassa interazione, catturano informazioni limitate ma sono economici e semplici da implementare e mantenere.

*Research honeypot:* l'obiettivo di questi sistemi è invece la pura ricerca dei metodi e delle motivazioni di attacco. Sono più complessi da implementare e mantenere ma catturano un alto numero di informazioni. Vengono sfruttati principalmente da organizzazioni di ricerca (Honeynet Project), militari e governative, e sono generalmente honeypot ad alta interazione.

# 3. SISTEMI HONEYPOT OPEN SOURCE

## 3.1 DTK: Deception ToolKit

L'honeypot Deception Toolkit (DTK), creato da Fred Cohen nel 1998, è uno dei primi e più semplici sistemi a bassa interazione. Questo sistema difensivo simula nelle porte della macchina ospitante un gran numero di vulnerabilità conosciute e registra le interazioni con un presunto attaccante.
Questa soluzione, progettata principalmente per attacchi da parte di umani, è oggi usata di rado poichè ampiamente superata da altre soluzioni, come Honeyd.

## 3.2 Honeyd

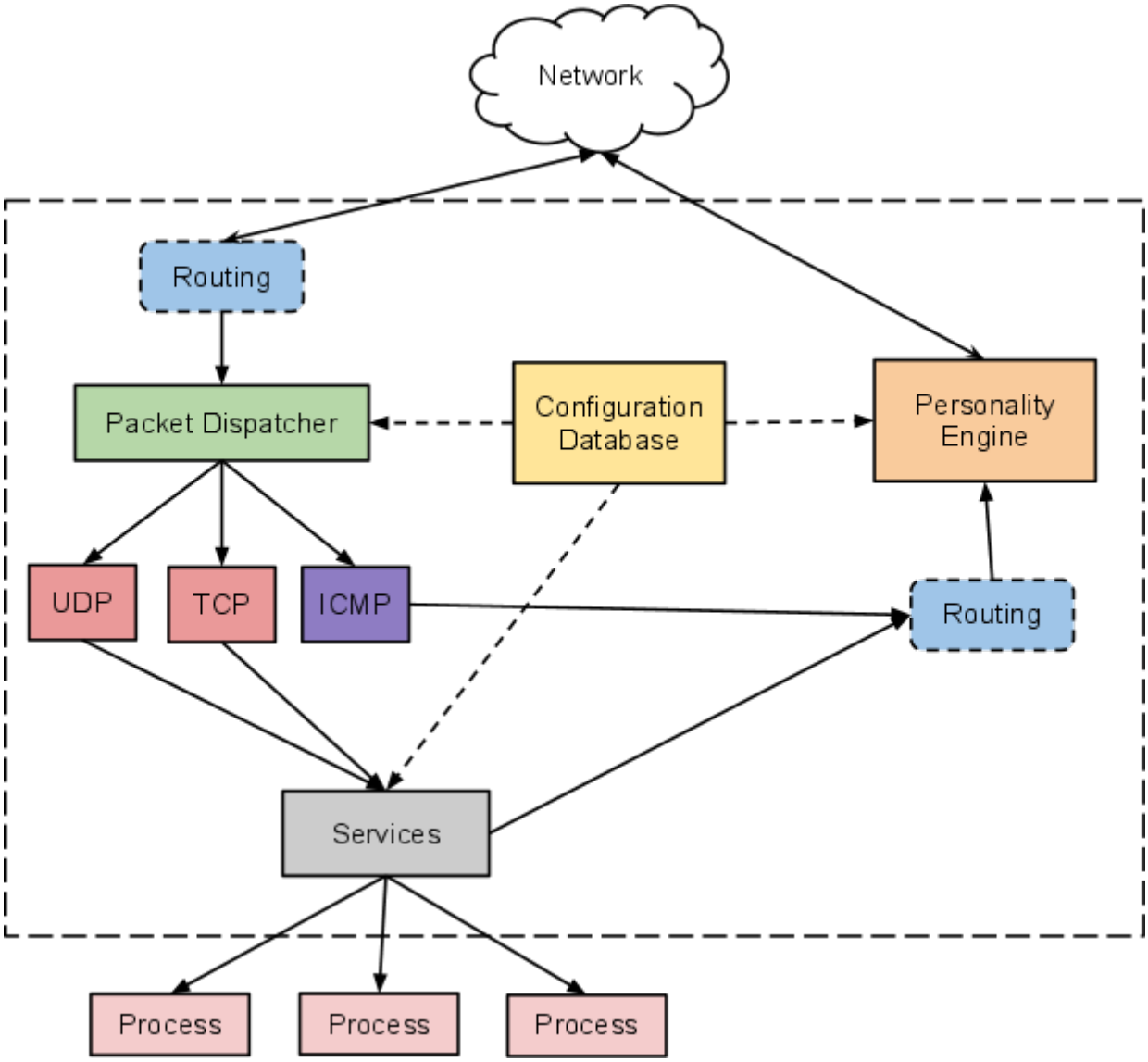
Honeyd può essere considerato un'evoluzione di DTK: è un classico honeypot a bassa interazione, le cui peculiarità si possono riassumere in:

- emulazione di un elevato numero di host (considerabile quindi come un'*honeynet*);
- ottima configurabilità;
- emulazione di vari router per la gestione di più sottoreti;
- integrazione tra rete fisica e virtuale.

Honeyd può emulare degli host a cui vengono assegnati indirizzi IP non utilizzati nella rete a cui appartiene il sistema ospitante. Viene quindi esaminata qualsiasi attività relativa ai protocolli TCP, UDP e ICMP, e vengono emulati diversi servizi utilizzando script in grado di interagire con l'aggressore.

I pacchetti indirizzati verso uno degli indirizzi assegnati all'honeypot vengono processati dal *packet dispatcher* che controlla la lunghezza dell'indirizzo IP e ne verifica l'integrità tramite il relativo checksum., e i pacchetti appartenenti a protocolli diversi da TCP, UDP e ICMP vengono scartati.

A questo punto viene interrogato il *database di configurazione* contenente le configurazioni di tutte le macchine simulate dall'honeypot, ovvero il sistema operativo, i servizi, gli indirizzi IP, MAC, ed altre caratteristiche specifiche dell'host simulato (definite nel file di configurazione /etc/honeypot/ honeyd.conf). In base all'indirizzo IP di destinazione viene associato il modello della macchina corrispondente, in mancanza del quale viene fornito un modello di default. Tale associazione serve per effettuare dei controlli nelle connessioni e al processo di personalizzazione dei pacchetti di risposta (effettuato dal *personality engine*).




Architettura di Honeyd

I pacchetti appartenenti al protocollo ICMP vengono processati da un modulo apposito, mentre i pacchetti UDP e TCP vengono inviati ad altri moduli i quali si interfacciano con gli script che emulano i diversi servizi dell'honeypot.

Prima che un pacchetto sia immesso nella rete viene processato dal *personality engine*, che apporta delle modifiche al suo contenuto per renderlo compatibile con lo stack utilizzato dal sistema operativo simulato: l'assunzione essenziale è infatti che l'avversario possa interagire con honeyd solo tramite la rete, e non in locale, in modo che non sia necessiario simulare l'intero sistema operativo, ma solo il suo stack di rete.

I pirati informatici infatti utilizzano strumenti come Nmap e Xprobe, che esaminando lo stack di rete del sistema bersaglio riescono ad identificare con sufficiente precisione il sistema operativo in esecuzione. Le informazioni così ricavate sono utili per risalire ad eventuali vulnerabilità e approntare un attacco mirato: Honeyd riesce a simulare le caratteristiche dello stack di rete di un determinato sistema operativo modificando gli header dei pacchetti in uscita in modo da garantire una corrispondenza con le caratteristiche di quest'ultimo. Per default Honeyd sfrutta il fingerprinting

database di Nmap come riferimento per le personalità TCP e quello di Xprobe per quelle ICMP, ingannando gli attaccanti con le loro stesse armi.

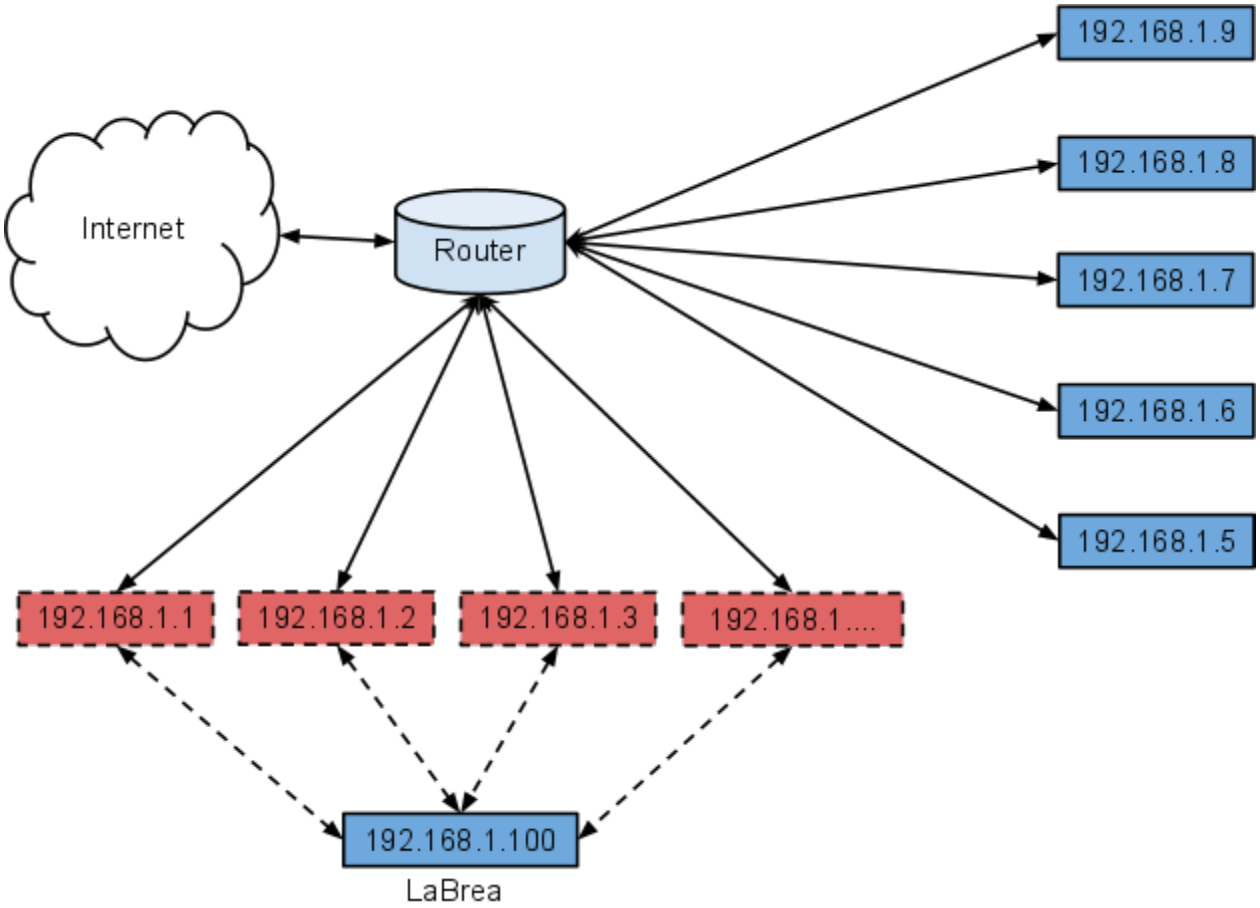
## 3.3 LaBrea

LaBrea è una creazione di Tom Liston, ed è famosa per aver introdotto il concetto di *tarpit*. Il termine “tar pit” si traduce come “fossa di catrame”, che sta a indicare qualcosa di letale che intrappola e impedisce i movimenti. Quest’honeypot è ideata per essere efficace contro attacchi automatizzati: un servizio *tarpit* cerca di rallentare le connessioni di worms, spammer e scanner rendendole molto lente o addirittura mettendole in condizione di stallo.

Una volta installato in una rete, LaBrea identifica gli indirizzi IP non assegnati e se ne appropria utilizzando il protocollo ARP (Address Resolution Protocol), che è un protocollo di rete del secondo livello OSI appartenente alla suite del protocollo internet (IP). Il compito di questo protocolloè fornire la “mappatura" tra l'indirizzo IP e l'indirizzo MAC corrispondente di un terminale in una rete locale Ethernet.

Quando un router prova a consegnare un pacchetto ad un indirizzo IP a lui sconosciuto, esso genera delle richieste ARP broadcast, che raggiungono ogni macchina della rete, le quali “chiedono” alla macchina con il MAC in questione di comunicare il proprio indirizzo IP. Se nessun sistema della rete risponde, il pacchetto potrebbe appartenere ad un tentativo di spam o ad un’infiltrazione malware, che tentano connessioni con tutti gli IP possibili.
Di questo indirizzo IP se ne prende carico LaBrea, rispondendo con il MAC address della macchina sulla quale è installato: il router invierà quindi tale pacchetto e tutti i successivi all’honeypot.

Quando viene stabilita una nuova connessione l’honeypot cerca di metterla in uno stato nel quale non può fare alcun progresso: l’obiettivo finale è il blocco del sistema attaccante, in quanto ogni connessione mantenuta aperta da una macchina riduce le risorse disponibili ad attaccare le macchine reali. Per fare ciò LaBrea imposta opportunamente la *finestra di ricezione*: è un parametro caratteristico del protocollo TCP usata per gestire il flusso di dati in arrivo. E’ presente in ogni pacchetto TCP e fornisce al ricevente un'indicazione sullo spazio libero disponibile nel buffer dei pacchetti in arrivo del mittente.



LaBrea emula ogni indirizzo ip non utilizzato dalla rete: in rosso i sistemi emulati in blu i sistemi reali.

L'honeypot è in grado di provocare due effetti tramite questa variabile:

- *TCP Throttling:* impostando il window size con un valore molto piccolo, viene simulata una "quasi congestione" di traffico destinato all'honeypot. Il mittente, come da protocollo, deve inviare i dati ad una velocità tale da non riempire il buffer del destinatario ed è costretto a rallentare la connessione.

- *Persistent capture:* LaBrea imposta una finestra di ricezione pari a zero, in questo caso viene simulato un congestionamento completo. Il mittente, come da protocollo, periodicamente invia dei pacchetti di soli zeri fintanto che il destinatario non libera il suo buffer. Questo stato provoca di fatto uno stallo della connessione.

La differenza tra questi due effetti sta nella reazione del sistema rallentato: se il windows size è zero questa macchina potrebbe di default ignorare l'host, mentre con un valore basso ma non nullo potrebbe ritentare ancora e ancora.

LaBrea è in grado quindi di bloccare le risorse di uno spammer o malware se l'attaccante è un malware, e di rallentare molto o bloccare l'acquisizione di informazioni da parte di un attaccante umano tramite lo scanning, inducendolo a distogliere le sue attenzioni dalla rete protetta.

## 3.4 Nepenthes

Nepenthes è stato sviluppato principalmente da Paul Baecher e Markus koetter. Il nome *nepenthes* deriva da "not" e "penthos" che combinati si traducono in "indolore", ed indica oltre all'honeypot

anche una pianta carnivora tropicale.

La peculiarità di quest'honeypot è la cattura di worm per analizzarli e studiarne gli obiettivi. Opera a bassa interazione e, come honeyd, può simulare più honeypot in parallelo creando un'honeynet. Si basa su un design molto flessibile e modulare: il nucleo gestisce l'interfaccia di rete e coordina le azioni degli altri moduli, che svolgono il lavoro effettivo. Essi si dividono in:

- *Moduli vulnerabilità:* sono il fattore principale della piattaforma Nepenthes, e forniscono un efficace meccanismo per raccogliere malware. L'idea principale alla base di questi moduli è che per essere infettati da un malware, è sufficiente emulare solo le parti necessarie di un servizio vulnerabile. Così, invece di emulare l'intero servizio, questi moduli emulano solo le parti pertinenti alla vulnerabilità. Questo è sufficiente per ingannare il malware e indurlo a credere che possa davvero sfruttare l'honeypot.

- *Moduli di analisi shellcode:* analizzano il *payload* ricevuto ed estraggono automaticamente le informazioni pertinenti al tentativo di compromissione. Il modulo applica delle operazioni di ricerca di pattern per individuare le funzioni comuni utilizzate negli exploit (come il richiamo della funzione CreateProcess), e per cercare la locazione remota di propagazione del malware, informazione che viene comunicata ai moduli fetch per il download del codice maligno.

- *Moduli fetch:* hanno il compito di scaricare i file del worm dalla postazione remota, in base alle informazioni estratte dai moduli shellcode. Sono supportati i protocolli TFTP, HTTP, FTP e Csend / Creceive (un metodo di comunicazione caratteristico delle botnet). Dal momento che alcuni tipi di minacce informatiche si diffondono tramite protocolli personalizzati, vi sono anche i moduli per gestire queste situazioni.

- *Moduli di presentazione:* si occupano di gestire i file scaricati correttamente. Attualmente, ci sono tre diversi tipi di moduli di presentazione, un modulo memorizza il file in una posizione sul filesystem, un modulo lo invia ad un database centrale, un altro modulo lo invia a diversi server web, dove il binario è ulteriormente analizzato dai motori antivirus.

- *Moduli di registrazione:* registrano le informazioni relative al processo di emulazione e aiutano ad avere una panoramica leggibile dei dati raccolti.

Questi moduli sono gestiti dal nucleo di Nephenthes, che gestisce la loro cooperazione e le connessioni del protocollo TCP.

Quest'honeypot è quindi in grado di ingannare i worm estrapolando loro i file binari responsabili dell'infezione senza eseguirli, ovvero trattando codice pericoloso in maniera "indolore", da cui il nome Nephenthes.

## 3.5 GHH: Google Hack Honeypot

Google Hack Honeypot (GHH) è un honeytoken, sviluppato per combattere chi approfitta della potenza onnipresente dei motori di ricerca. I motori di ricerca infatti permettono di trovare informazioni sensibili a causa di errori di configurazione dei server web, o addirittura identificare gli

host che eseguono applicazioni web vulnerabili.

Tali motori di ricerca, e specialmente Google, offrono una serie di strumenti che molte persone ignorano.
Un classico esempio di ricerca di dati sensibili utilizzando Google è la caccia ai numeri di carte di credito. Le prime quattro cifre del numero di una carta forniscono molte informazioni sulla tipologia di carta, pertanto ci sono molte carte che condividono lo stesso prefisso. Un malintenzionato può quindi utilizzare questo strumento per cercare codici di carte di credito con particolari caratteristiche: per esempio digitando 4052000000000000 .. 4052999999999999 in Google viene effettuata una ricerca di siti web contenenti qualsiasi numero di 16 cifre che iniziano con 4052.

Ci si potrebbe però chiedere chi si diverta a pubblicare una pagina piena di numeri di carta di credito su internet: nessuno ovviamente. Ma le applicazioni web mal costruite che vendono prodotti su Internet sono il problema. Alcuni siti web fanno uso di legami invisibili ai dati di back-end, come liste dei clienti e loro dati: un consumatore non vedrebbe mai questi link, ma Google si e ne indicizza il contenuto.

L'idea dietro a GHH è il porre un legame invisibile dalla ricerca su Google al sito web. Proprio come nel caso di un'applicazione mal costruita, i visitatori del sito non potranno mai vedere questo link, ma Google si. Invece di fornire l'accesso ai dati di back-end, il link indirizza gli attaccanti ad uno script PHP che simula tali dati e consente di registrare ogni loro attività.

È possibile utilizzare Google Hack Honeypot per rilevare attività sospette contro il server web e utilizzare le informazioni ricavate dai log per impostare di conseguenza il firewall.

## 3.6 I servizi emulati

I servizi emulati dagli honeypot dipendono da esso: alcuni sistemi inglobano già dopo l'installazione i servizi più soggetti ad attacchi, mentre altri ne sono sprovvisti e lasciano l'implementazione di questi all'amministratore.

Il compito di attrarre il possibile attaccante spetta ai servizi emulati: per aumentare la probabilità che l'intruso violi proprio l'honeypot tra tutti gli host presenti nella rete da proteggere è necessario offrirgli i servizi più comuni è più facili da compromettere.

La SANS Institute, una cooperativa di professionisti in sicurezza informatica, fornisce le vulnerabilità più comuni in ambiente Linux e Windows. Verranno qui elencati ed analizzati i servizi soggetti alle vulnerabilità più note dei sistemi basati su ambiente operativo Linux.

- *Remote Procedure Calls*: le chiamate a procedure remote (o RPC, da Remote Procedure Calls) permettono a certi programmi di eseguire codice da remoto, su un secondo elaboratore, mediante l'invio di dati e la ricezione dei risultati relativi. L'amministrazione remota spesso è eseguita proprio mediante RPC, e in genere l'esecuzione di tali procedure richiede privilegi amministrativi elevati quali l'account di root. Il tipo di attacco più comune indirizzato a RPC è il buffer overflow, con cui cercare di guadagnare l'accesso remoto alla macchina con privilegi di root.

- *Apache Web Server*: tra i più recenti problemi di sicurezza di questo famoso servizio di web server vi sono il worm Slapper, e una vulnerabilità dovuta alla gestione di sequenze di record logici di alcuni file (chunk handling), che consentono all'attaccante di eseguire codice con permessi di root.

- *Secure Shell (Ssh)*: è un metodo diffuso per rinforzare gli accessi, l'esecuzione di comandi ed il trasferimento di file in una rete. Disponibile sia in versione commerciale (Communication Security Ssh) che in versione libera (OpenSsh), Ssh può presentare problemi di sicurezza tali da permettere ad un attaccante di guadagnare l'accesso remoto da root.
Tra il 30 ed il 31 luglio 2002 è stata diffusa una versione di OpenSsh contenente codice maligno (un trojan, vedi www.openssh.org/txt/trojan.adv). Le versioni più affette da errori, e quindi vulnerabili, sono quelle di OpenSSh 3.3 o precedente, e Communication Security Ssh 3.0.

- *Simple Network Management Protocol (Smnp)*: è usato per controllare e configurare da remoto gran parte delle periferiche vertenti su Tcp/Ip, come la configurazione e la gestione di stampanti, router o switch.
Le vulnerablità ed i relativi attacchi si concentrano sull'autenticazione e gestione dei messaggi tra gli host di controllo e le periferiche. La quasi totalità dei sistemi Unix e Linux prevede Smnp attivo per default, nelle sue varianti a codice chiuso od aperto.

- *File Transfer Protocol (Ftp)*: è usato per distribuire in rete file ad utenti anonimi oppure autenticati mediante username e password. I server Ftp anonimi non richiedono in genere password, consentendo a più utenti di accedere con lo stesso nome o password.
Molti problemi di sicurezza nascono dal fatto che i parametri di autenticazione transitano sulla rete in chiaro, e possono essere intercettati con uno sniffer, collocato sul percorso dal server al client. Alcuni attacchi permettono l'accesso remoto da root sul server; altri l'esecuzione di comandi con privilegi di utente.

- *Servizi R*: i comandi R, come rsh per shell remota, rcp per copia remota, rlogin per accesso remoto o rexec per esecuzione remota sono largamente usati in ambito Unix. I corrispondenti servizi R in.rshd, in.rlogind, in.rexecd possono essere configurati in modo tale da non richiedere username o password. In questo contesto viene effettuato il cosiddetto bouncing, ovvero il passaggio da una macchina all'altra con scarsa tracciabilità.
I problemi dei servizi R sono di due tipi: mancanza di cifratura, che permette l'intercettazione delle password, e autenticazione debole sull'host. Quasi tutti i sistemi Unix e Linux prevedono l'installazione dei servizi R di default; molti li abilitano automaticamente.

- *Line Printer Daemon (Lpd)*: il server di stampa di molti sistemi Unix, permette l'accesso locale o remoto (sulla porta 515) a una stampante.
Anche se vi sono sistemi alternativi quali Cups, lpd rimane il print server più diffuso. Alcune versioni di lpd, tuttavia, sono vulnerabili ed essendo esposte ad attacchi basati sul buffer overflow permettono l'esecuzione remota di codice con privilegi di root. La maggioranza dei sistemi Unix e Linux prevedono lpd installato di default e abilitato.

- *Sendmail*: è il server di posta per antonomasia dei sistemi Unix; la sua diffusione lo ha reso nel tempo bersaglio privilegiato di molti attacchi. Specie per le versioni più datate aventi configurazioni non appropriate, gli attacchi permettono di guadagnare privilegi amministrativi tramite buffer

overflow e di eseguire il relay di messaggi con fini non consentiti, quali spam o mail bombing.

- *Bind*: Berkeley Internet Domain Server, è una delle più usate implementazioni di Dns (Domain Name Server), il sistema per la traduzione di un indirizzo Ip numerico in un nome di dominio. La sua diffusione comporta un alto numero di attacchi subiti; quelli che vanno a buon fine sono spesso dovuti a configurazioni di sistema non corrette e all'uso non necessario del servizio (eseguito come demone named). Tra le debolezze più recenti, si ricordano una falla sfruttabile per Denial of Service ed un buffer overflow nelle librerie del Dns revolver.

# 4. RILEVAZIONE DEGLI HONEYPOT

Anche se gli honeypot sono una grande risorsa per lo studio degli attacchi in qualsiasi loro forma, la quantità di informazioni che possiamo estrarre da tale tecnica dipende molto da quanto realistico è l'honeypot. Se un avversario viola un sistema e nota subito che ha avuto accesso ad un honeypot e non un sistema reale, potrebbe rimuovere tutte le prove relative all'intrusione e spostare le sue attenzioni su altre reti. Invece se non se ne accorge potrebbe usarlo per memorizzare strumenti di attacco e lanciare ulteriori attacchi ad altri sistemi, rivelando molte più informazioni, ed è per questo motivo che l'honeypot deve risultare il più realistico possibile.

## 4.1 Rilevazione degli honeypot a bassa interazione

Gli honeypot a bassa interazione non forniscono un ambiente operativo completo agli attaccanti e sono facilmente smascherabili. Un primo indizio sulla loro natura sono i servizi, semplici emulazioni con inevitabili discrepanze dai servizi reali, che l'attaccante può notare. Per esempio, un classico servizio come ssh se interrogato per la prima volta da un host non conosciuto risponde con un testo come il seguente:

```
root@debian-giulio:/home/workteam# ssh 192.168.252.82
The authenticity of host '192.168.252.82 (192.168.252.82)' can't be established.
RSA key fingerprint is 47:22:b5:85:f9:f6:9b:87:be:89:d7:3d:93:68:05:7e.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.252.82' (RSA) to the list of known hosts.
root@192.168.252.82's password:
```

Oppure se l'host in uso ha già provato in precedenza ad accedere al servizio:

```
root@debian-giulio:/home/workteam# ssh 192.168.252.99
root@192.168.252.99's password:
```

Una volta inserita la password corretta il servizio replica con:

```
Linux server-ikarus 2.6.32-5-686 #1 SMP Mon Jun 13 04:13:06 UTC 2011 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
You have mail.
Last login: Mon Aug 15 20:48:16 2011
root@server-ikarus:~#
```

Nell'emulare questo servizio, si potrebbe distrattamente dimenticare di inserire nelle risposte alle comunicazioni qualche dato, oppure sbagliarne l'ordine, fare l'autenticazione degli host ad ogni connessione o addirittura un errore di battitura. Altre disattenzioni possono smascherare il sistema difensivo, per esempio una mal configurazione dell'honeypot, come l'emulazione di un web server Windows con servizi tipici di server Unix.

Vi sono inoltre altri inconvenienti che smascherano gli honeypot a bassa interazione. Sotto l'ipotesi che l'unico metodo per accedervi è attraverso la rete, le risorse della macchina ospitante sono suddivise tra il sistema operativo e tutti i processi in esecuzione su di esso, compreso l'honeypot: impegnando tali risorse con molti processi, quindi, si causerà un rallentamento di quest'ultimo.

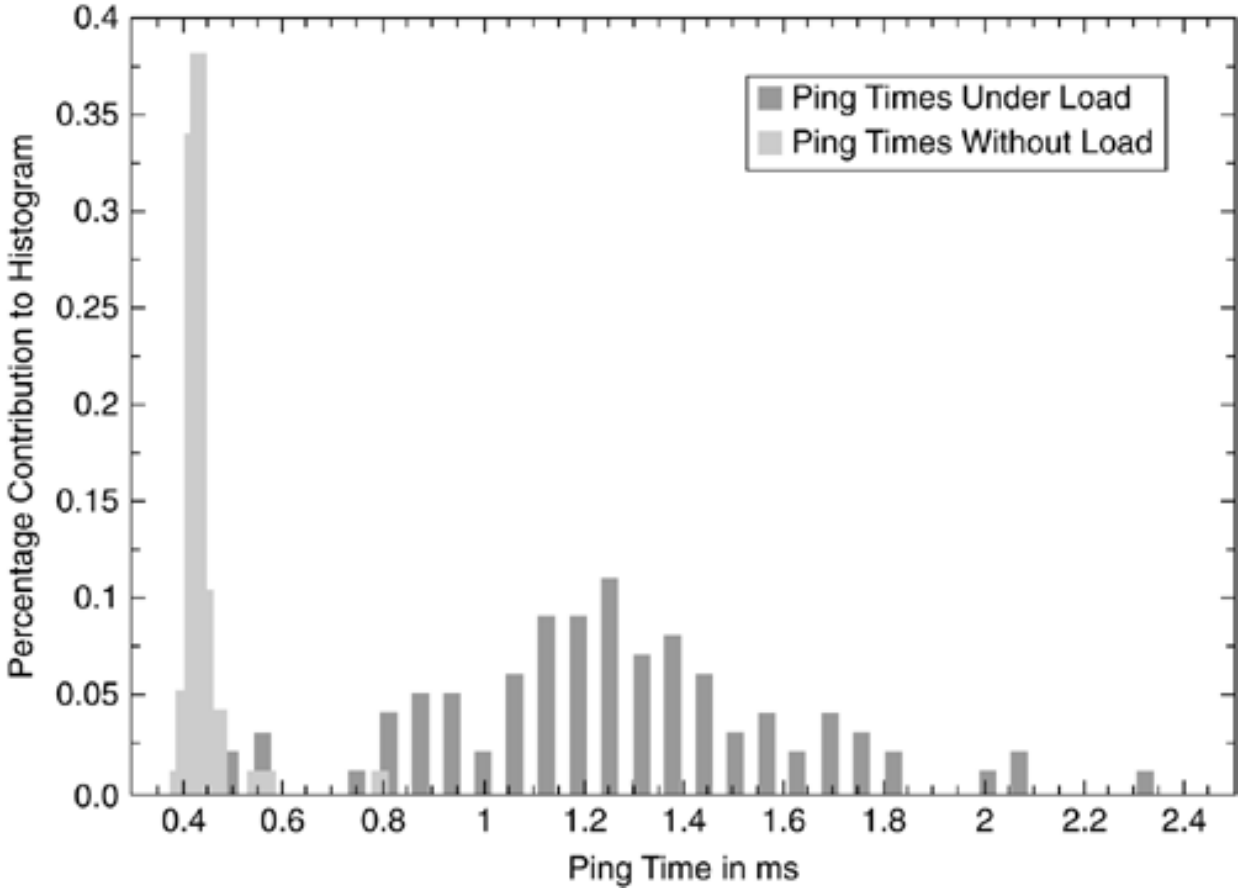
Un esperimento molto semplice per dimostrare questa interazione è il seguente. Una macchina con indirizzo IP 192.168.1.10 che indicheremo come Alice, fa girare un'istanza di Honeyd che emula una macchina con indirizzo IP 192.168.1.90, che chiameremo Bob. Si effettua una prima misurazione della latenza di Bob con lo strumento ping:

```
root@debian-giulio:/home/workteam# Ping -c 100 192.168.1.90
PING 192.168.1.90 (192.168.1.90): 56 byte of data
64 byte from 192.168.1.90: icmp_seq = 0 ttl = 63 time = 0,443 ms
64 byte from 192.168.1.90: icmp_seq = 1 ttl = 63 time = 0,430 ms
64 byte from 192.168.1.90: icmp_seq = 2 ttl = 63 time = 0,434 ms
64 byte from 192.168.1.90: icmp_seq = 3 ttl = 63 time = 0,421 ms
...
```

La seconda misurazione ha luogo con un carico aggiuntivo su Alice (un ping flood):

```
root@debian-giulio:/home/workteam# Ping -f 192.168.1.10 &
root@debian-giulio:/home/workteam# Ping -c 100 192.168.1.90
PING 192.168.1.90 (192.168.1.90): 56 byte of data
64 byte from 192.168.1.90: icmp_seq = 0 ttl = 63 time = 0,541 ms
64 byte from 192.168.1.90: icmp_seq = 1 ttl = 63 time = 0,595 ms
64 byte from 192.168.1.90: icmp_seq = 2 ttl = 63 time = 0,802 ms
...
```

Per entrambe le misure, ci si dovrebbe aspettare una distribuzione gaussiana con centro posto approssimativamente vicino alla media dei tempi.




Il grafico delle latenze mostra una notevole discrepanza tra le latenze nei due casi che porta inevitabilmente allo smascheramento dell'honeypot.

In maniera analoga è possibile smascherare LaBrea, in quanto indirizzando un grosso flusso di dati su un indirizzo IP gestito da tale honeypot si aumentano latenze e tempi di risposta di tutti gli altri indirizzi. D'altronde per smascherare LaBrea basta poco: le tecniche che utilizza, descritte nel terzo capitolo, sono infatti raramente riscontrabili nella rete, chiaro sintomo di anomalie.

Per i sistemi come Honeyd inoltre, eventuali bug o anomalie nella gestione dei pacchetti possono essere la chiave per il loro rilevamento.
In passato infatti ci sono stati diversi modi per rilevare Honeyd in modo quasi banale: nel gennaio 2004, un bug nella gestione delle impronte digitali NMAP induceva l'honeypot a rispondere a pacchetti TCP con entrambi i flag SYN e RST abilitati: nessun'altra macchina in internet considererebbe valido tale pacchetto. Una semplice risposta a tali pacchetti quindi smascherava i sistemi che implementavano Honeyd.

Altro bug di Honeyd riguardava il riassemblamento dei pacchetti IP. Secondo il protocollo IP i frammenti di un pacchetto sono identificati con: indirizzo sorgente, indirizzo di destinazione, numero di identificazione e la versione del protocollo; purtroppo, Honeyd nella ricostruzione non controllava l'ultimo parametro, riassemblando frammenti di protocolli diversi che, in una macchina normale verrebbero semplicemente scartati.

## 4.2 Rilevazione degli honeypot ad alta interazione

La rilevazione degli honeypot ad alta interazione è più complessa perchè, come già illustrato, questi sistemi si avvicinano molto più ad un sistema reale rispetto agli honeypot a bassa interazione ed il loro smascheramento dipende fortemente dal tipo di implementazione scelto per il sistema. Se un honeypot ad alta interazione è implementato con un sistema reale, non vi è alcuna possibilità di riconoscerlo. L'unico indizio che potrebbe metterlo in cattiva luce è un traffico dati completamente nullo, ma è sufficiente aggiungere un servizio che scambia dati casuali con la rete interna per aggirare il problema. Questo non vale però se l'honeypot viene implementato con un ambiente virtuale.

L'uso di VMware, d'ora in avanti VMW, è oggi una scelta molto popolare per tale implementazione. La virtualizzazione hardware di VMware simula una piattaforma x86 completa di hard disk, scheda di rete e dispositivi I/O: per far ciò, VMW fornisce un'interfaccia virtuale con la quale il sistema operativo virtualizzato ha bisogno di comunicare, tramite driver appositi. Questo dettaglio fornisce un semplice modo per individuare VMW.

Eseguendo il seguente comando

```
sudo dd if=/dev/mem bs=64k skip=12 count=1 | strings -n10
```

su un sistema Linux non virtualizzato, viene restituito un output generico del tipo:

```
IBM VGA Compatible
NVIDIA P119 GeForce4 MX 4000 VBIOS
Version 4.18.20.39.00
Copyright (C) 1996-2003 NVIDIA Corp.
NV18 Board - 119s2937
Chip Rev A4
WinFast A180B VGA BIOS V01.12.2004
Copyright (C) 2000-2005 Leadtek Research Inc. Date:01/12/2004(V8.1)
...
```

Mentre un sistema virtualizzato restituisce un output più povero:

```
IBM compatibile VGA
PhoenixView (tm) VGA BIOS compatibile Version
Copyright (C) 1984-1992 Phoenix Technologies Ltd.
Tutti i diritti riservati
...
```

La povertà di informazioni fornita dal sistema operativo riguardo le componenti hardware può non essere sufficiente per provare che il sistema è virtualizzato, ma è sicuramente un indizio che può mettere in allarme l'attaccante.
Ci sono infatti altri difetti di VMWare per rilevare le macchine virtuali: analizzando gli indirizzi MAC delle interfacce di rete si può risalire alla natura della scheda di rete stessa. I seguenti intervalli di indirizzi MAC, sono assegnati da IEEE a VMWare:

```
00-05-69-xx-xx-xx
00-0C-29-xx-xx-xx
00-50-56-xx-xx-xx
```

Se un sistema possiede un indirizzo di questo tipo è quindi un chiaro segno di virtualizzazione.

Altri software di virtualizzazione, come Parallels per computer Apple, rimediano a questo problema fornendo indirizzi MAC diversi. L'esecuzione di Linux sotto Parallels su Mac OS X, fornisce queste informazioni riguardo la scheda di rete:

eth0: RealTek RTL-8029 disponibili all'indirizzo 0x1080, IRQ 10, 00: A1: 9B: XX: XX: XX

Il kernel ha rilevato una scheda Ethernet, prodotta da RealTek, con un indirizzo MAC con prefisso 00-A1-9B. Secondo gli standard IEEE a RealTek è stato assegnato il prefisso 00-E0-4C, mentre il prefisso 00-A1-9B non è registrato. Tale discrepanza permette quindi di smascherare la virtualizzazione.

Molte di queste "non trasparenze" di VMW possono essere riparate utilizzando uno strumento sviluppato da Kostya Kortchinsky, anche se purtroppo tale soluzione è valida solo per le vecchie versioni di VMware. Questo tool modifica direttamente i file binari di VMware, dando la possibilità di impostare il MAC address a proprio piacimento, e di rimpiazzare i bios dell'hardware installandone di nuovi o di copiarli da altre macchine (il bios in questo caso deve essere compatibile con l'hardware del sistema ospitante VMW).

Sembra a questo punto che sia possibile realizzare una macchina virtuale indistinguibile da una una reale, ma non è così. La costruzione di un sistema virtuale trasparente è fondamentalmente irrealizzabile, oltre che impraticabile dal punto di vista delle prestazioni. Rimangono infatti delle discrepanze impossibili da correggere, legate alla natura del sistema virtuale:

- *Discrepanze logiche*: secondo Tal Garfinkel, John Scott Robin e Cynthia E. Irvine, l'esecuzione di istruzioni privilegiate come SIDT, SGDT, e SLDT su processori Intel permettono di scoprire l'ubicazione delle tabelle dei descrittori globali, locali e di interrupt. Queste tabelle infatti sono strutture dati usate dai processori della famiglia x86 di Intel per definire varie aree di memoria nell'esecuzione di un programma e per gestire interrupt. Vengono utilizzate molto frequentemente, e VMware non usa una traduzione binaria per accedervi ma prevede invece delle tabelle diverse in locazioni diverse. Di conseguenza, è possibile rilevare la presenza di una macchina virtuale semplicemente controllando se la loro posizione corrisponde ad un sistema reale o virtualizzato.

- *Discrepanze delle risorse*: dato che il sistema operativo ospitante deve condividere le risorse fisiche con i sistemi operativi virtualizzati, la disponibilità di esse non è sempre garantita o ugualmente disponibile.
Un semplice esempio fornito da Garfinkel Tal è il "Translation Look-aside Buffer" (d'ora in poi TLB): accedendo a indirizzi di memoria presenti in molte pagine differenti in modo incrementale, è possibile stimare la dimensione di tale buffer, poichè ad un certo punto si verificherà un TLB "miss". Si verifica questo dato prima in normali condizioni e dopo eseguendo delle istruzioni che, se presente, invocano l'esecuzione il Virtual Machine Manager (d'ora in poi VMM). Se le due misurazioni sono uguali, significa che il VMM non è presente e quindi il sistema è reale; invece se la seconda misura è inferiore alla prima significa che il VMM, andando in esecuzione, ha provocato altri accessi alle pagine che ne contengono il codice che vanno a modificare alcune entry del TLB. La conseguenza principale di questo evento è la riduzione della dimensione del TLB vista dalla macchina virtuale.

- *Discrepanze temporali*: eludere il rilevamento di queste discrepanze diventa quasi impossibile quando l'avversario ha a disposizione degli strumenti remoti per effettuare misurazioni temporali

delle risposte del sistema. Le differenze temporali infatti non sono solo dovute al carico della virtualizzazione, le quali potrebbero essere parzialmente risolte mediante l'esecuzione su hardware più prestante. Quando si eseguono due qualsiasi operazioni nel sistema operativo virtualizzato, le loro differenze temporali cambiano nel tempo, in relazione allo stato in cui si trova la macchina. Tal Garfinkel fornisce un esempio: un registro PCI che richiede un centinaio di cicli per leggere dei dati dall'hardware può richiedere solo un ciclo quando la sua versione virtuale è presente nella cache del processore. A causa del comportamento di caching quindi, la varianza del numero di cicli di questa operazione nel sistema virtuale può essere molto più alto di quanto ci si aspetterebbe dal sistema reale.

VMW ha però alcune "caratteristiche" non documentate per nascondere la sua presenza, pubblicate da Liston e Skoudis in una presentazione su come contrastare il rilevamento delle macchine virtuali. Se si aggiungono le seguenti righe al file vmx (file di configurazione utilizzato VMW per personalizzare le macchine virtuali) della macchina virtuale, i trucchi utilizzati per la rilevazione delle discrepanze logiche e delle risorse vengono disattivati:

```
isolation.tools.getPtrLocation.disable = "TRUE"
isolation.tools.setPtrLocation.disable = "TRUE"
isolation.tools.setVersion.disable = "TRUE"
isolation.tools.getVersion.disable = "TRUE"
monitor_control.disable_directexec = "TRUE"
monitor_control.disable_chksimd = "TRUE"
monitor_control.disable_ntreloc = "TRUE"
monitor_control.disable_selfmod = "TRUE"
monitor_control.disable_reloc = "TRUE"
monitor_control.disable_btinout = "TRUE"
monitor_control.disable_btmemspace = "TRUE"
monitor_control.disable_btpriv = "TRUE"
monitor_control.disable_btseg = "TRUE"
```

Con queste modifiche, rilevare la presenza di VMware diventa ancor più difficile ma si va incontro a degli effetti collaterali di poco conto nella gestione del sistema come il "copia incolla" tra quello reale e quello virtuale.
L'avversario medio comunque non fa alcuno sforzo per individuare una possibile virtualizzazione, e le macchine virtuali stanno diventando sempre più popolari per la loro facilità di gestione rispetto alle macchine fisiche.

# 5. VANTAGGI E SVANTAGGI, POSIZIONAMENTO

## 5.1 Vantaggi e svantaggi

Gli honeypot sono diversi dai comuni strumenti di sicurezza. Tra di essi vi sono i firewall, che vigilano sul perimetro di una rete monitorando le connessioni in ingresso ed in uscita gestendole attraverso opportune regole. Altri importanti strumenti sono gli Intrusion Detection System (da ora in poi IDS) che generano allarmi quando, esaminando i pacchetti in transito sulla rete, rilevano una probabile intrusione.

Gli honeypot inoltre non forniscono servizi di produzione, ovvero non creano, né trasformano, né modificano beni e ricchezze. Questo li differenzia dalle risorse che devono essere protette e alle quali devono "somigliare". Sono uno strumento flessibile e adattabile a più situazioni, ma vanno considerati come un complemento ai classici sistemi difensivi, con i quali spesso vengono confusi. Gli IDS, al contrario degli honeypot, ispezionano tutto il traffico di rete confrontandolo con i dati raccolti in un database di “attack signature” (una specie di DNA degli attacchi). Questa metodologia però, genera spesso dei *falsi positivi*, ovvero allarmi relativi a connessioni lecite interpretate però erroneamente come tentativi di intrusione.

Un honeypot è invece un sistema con dei compiti non convenzionali all’interno della rete; esso non deve avere nessun processo sconosciuto in esecuzione né generare alcun traffico di rete (al di fuori di quello permesso dall'amministratore del sistema difensivo).
Grazie a queste ipotesi ogni interazione con un honeypot è da considerare anomala e indice di violazione in atto. L'assenza di *falsi positivi* è uno dei vantaggi chiave degli honeypot.
I dati forniti da questi sistemi difensivi inoltre sono più limitati e più informativi rispetto alla grande mole di dati prodotti da un IDS.

Mentre un comune IDS può riconoscere solo attacchi già noti presenti nel database delle *signature* (in maniera simile a quanto accade per gli antivirus), gli honeypot per loro natura possono rilevare qualsiasi tipo di attacco, inclusi quelli sconosciuti, in quanto è sufficiente l'interazione con essi per provocare un allarme.

Ulteriore caratteristica degli honeypot, che può essere considerata vantaggiosa o svantaggiosa a seconda delle circostanze, è definita da Spitzner come “effetto microscopio”: l’honeypot registra il solo traffico che lo coinvolge, e ciò risulta essere molto importante in grosse reti con grandi moli di dati in transito. Grazie a questa caratteristica non sono quindi necessarie particolari prestazioni (al contrario di un IDS che analizza tutto il traffico della rete, richiedendo molte più risorse). Ma proprio a causa di questo suo campo visivo ristretto, l’honeypot è in grado di accorgersi esclusivamente degli eventi che lo interessano in prima persona. Se un attaccante non vi interagisce e cerca di violare il resto della rete (i sistemi da proteggere), l’honeypot è inutile e non segnalerà alcuna attività illecita.

Come è stato descritto nei capitoli precedenti, gli honeypot ad alta interazione possono essere implementati anche attraverso la virtualizzazione: invece di utilizzare un computer che agisce come

un singolo honeypot, è possibile implementarne uno che ospita diverse macchine virtuali che agiscono come più honeypot, formando un'honeynet. Di solito queste "reti trappola" sono costituite da diverse piattaforme e sistemi operativi, e sono caratterizzate da una maggiore capacità di attrazione degli attacchi e dalla raccolta di dati su diversi tipi di attacchi simultaneamente.
Questo approccio ha alcune interessanti proprietà: la virtualizzazione non è difficile da implementare ed è molto facile da gestire. Avvenuta la violazione e la registrazione delle azioni dell'attaccante in un arco di tempo definito, è facile ripristinare l'honeypot virtuale allo stato originale, come pure gestire e limitare le azioni dell'intruso. Inoltre utilizzare un honeypot virtuale è meno rischioso perché l'attaccante ha meno probabilità di compromettere o danneggiare la macchina ospitante (reale).
La virtualizzazione non è però completamente trasparente, come viene descritto nel capitolo precedente infatti un esperto in materia potrebbe fiutare l'inganno in atto e agire di conseguenza.

Lo svantaggio principale degli honeypot ad alta interazione, implementati su macchine reali, è il contenimento dell'attaccante una volta che è penetrato nel sistema: questi può infatti iniziare ad attaccare altri sistemi, dentro e fuori la rete, causando problemi sia legali sia etici.
Una soluzione ottimale a questo inconveniente oggi molto diffusa è costituita dall'Honeywall, una "barriera" creata dalla comunità di Honeynet Project. Questo sistema si occupa di tre fondamentali compiti in un'honeynet:

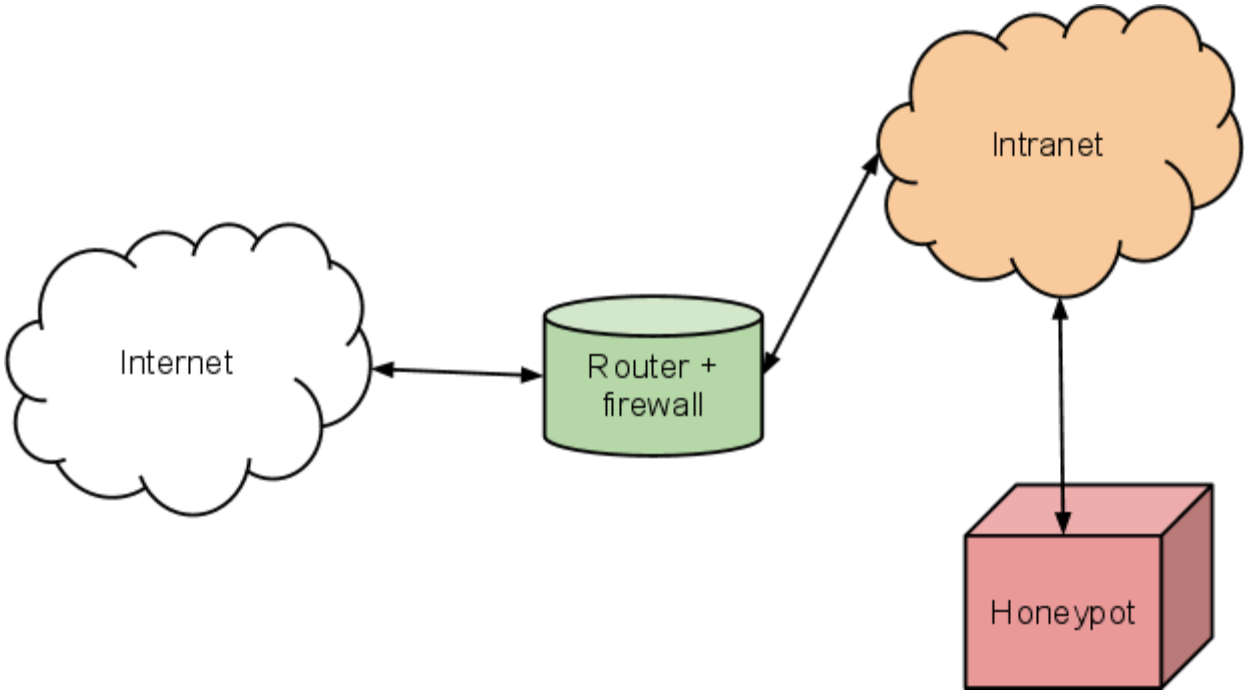
- *Data capture:* ogni attività interna e ogni informazione entrante o uscente dall'honeynet è catturata senza che l'attaccante ne venga a conoscenza.
- *Data control:* assicura che, dopo una compromissione all'interno dell'honeynet, tutte le attività maligne rimangano limitate all'interno dell'honeynet stessa.
- *Data analysis:* aiuta l'amministratore del sistema difensivo ad analizzare i dati raccolti.

Il contenimento dell'attaccante non è necessario negli honeypot a bassa interazione in quanto l'intruso non si impossessa di alcun sistema.

## 5.2 Posizionamento degli honeypot

Un aspetto non trascurabile nello studio degli honeypot è rappresentato dal loro corretto posizionamento all'interno della rete, che può pregiudicare l'efficacia di tali sistemi.

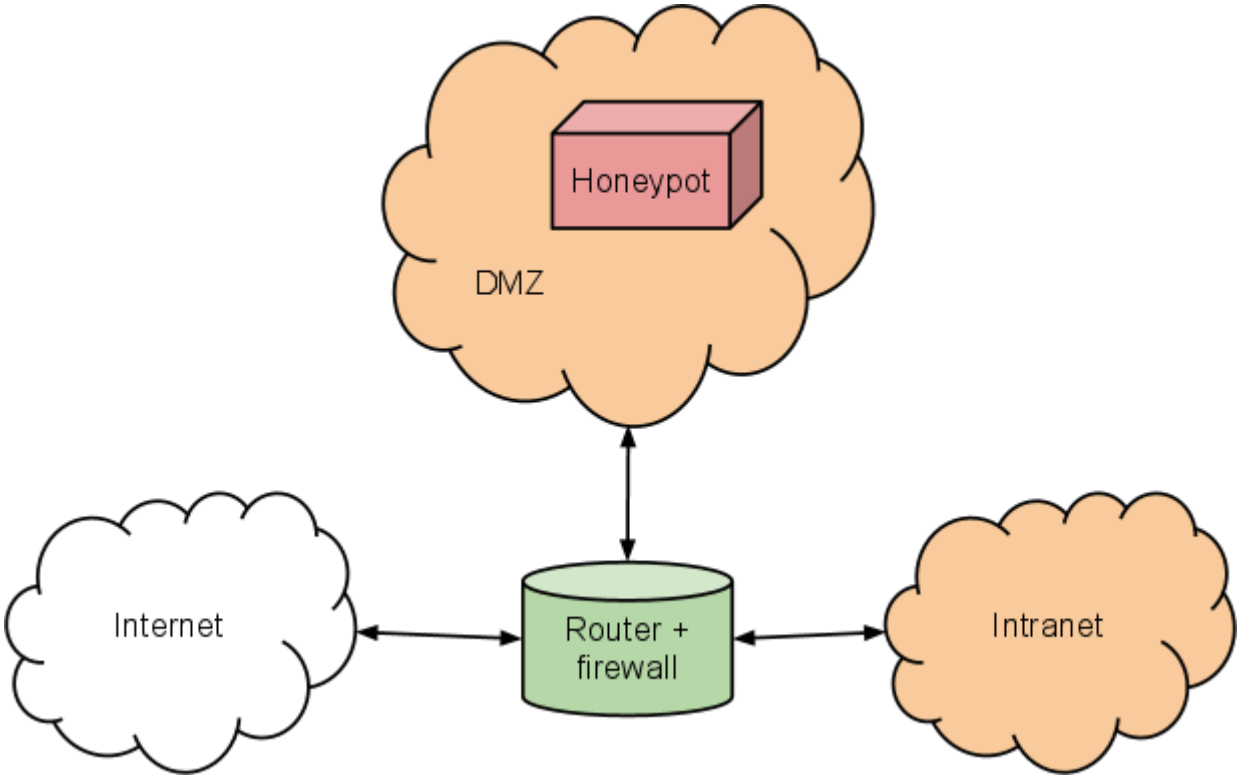
Gli *honeypot di produzione*, con funzione di rilevamento delle intrusioni, agiscono correttamente se posti all'interno del perimetro di sicurezza e se protetti da un firewall.




Corretto posizionamento di un honeypot di produzione.

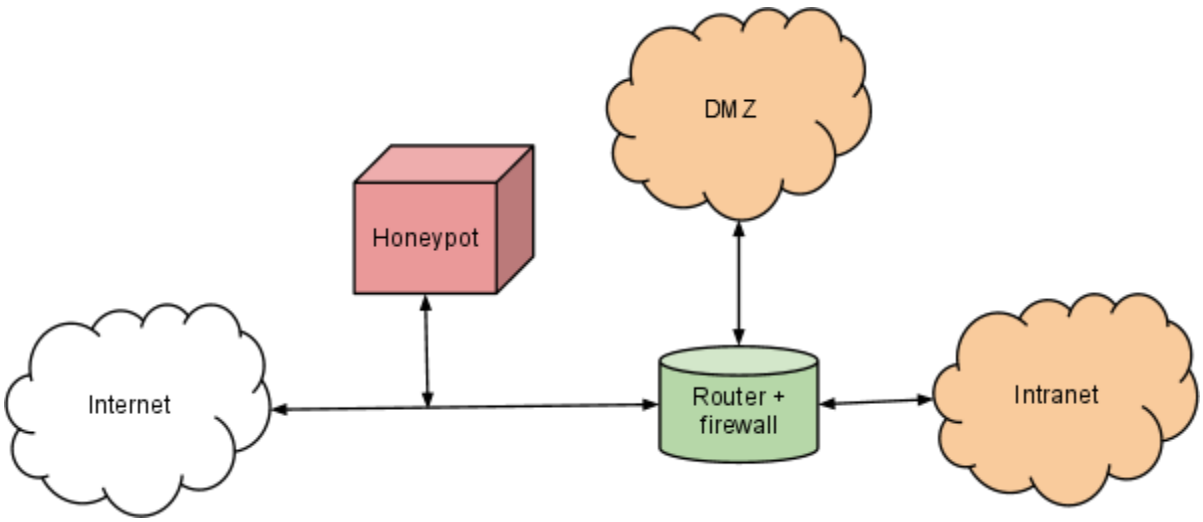
In questo modo possono fungere da sensore di allarme nel caso di attacchi che abbiano aggirato le barriere difensive ed eventualmente possono agire come deterrente per distogliere l'attenzione dalle risorse di valore. Collocarli fuori dal perimetro di sicurezza non avrebbe senso perché li esporrebbe ad un gran numero di attacchi, molti dei quali dovrebbero essere bloccati dagli altri dispositivi di difesa (come il firewall).

Se però gli *honeypot di produzione* vengono impiegati come meccanismo di reazione, per dirottare gli attacchi provenienti dall'esterno, è conveniente collocarli nella zona demilitarizzata (o DMZ), cioè un segmento di rete parzialmente accessibile da Internet dove normalmente trovano posto i servizi pubblici (ftp, www, smtp ecc.).




Corretta posizione di un honeypot di produzione se presente una zona demilitarizzata.

Al contrario, gli *honepot di ricerca* vanno posizionati esclusivamente al di fuori del perimetro di sicurezza per acquisire la massima quantità di informazioni sugli attacchi. In questo caso, trattandosi di sistemi particolarmente esposti, è molto elevato il rischio di compromissione e di conseguente impiego dello stesso per attaccare altre risorse, per cui risulta indispensabile adottare delle contromisure adeguate.




Corretto posizionamento di un honeypot di ricerca.

# GLOSSARIO

*Attaccante:* in questa tesi l'attaccante indica una qualsiasi entità che tenta di compromettere il sistema da proteggere con l'honeypot. Può essere umano o un malware.

*Botnet*: termine derivato da "bot" e "network", è una rete di computer controllati illegittimamente da un'unica entità. Con tali reti è possibile implementare attacchi DDoS.

*DDoS:* acronimo di Distribuite Denial of Server, è un attacco informatico distribuito effettuato da una rete di computer che mira a sovraccaricare le risorse dell'obbiettivo.

*Falsi positivi:* con questo termine vengono indicati i falsi allarmi generati da test o software.

*Host:* è considerato host ogni nodo in grado di comunicare connesso ad una rete di computer, server o terminali.

*Malware:* termine derivato da "malicious software", è un software creato per scopi maligni.

*Payload*: è una runtime presente in un virus informatico che ne estende le funzioni oltre l'infezione del sistema. Si intende con payload quindi qualsiasi operazione a tempo determinato, casuale o attivata da un trigger che un virus o worm manda in esecuzione. Questa può essere di distruzione parziale o totale di informazioni, la loro diffusione non autorizzata, l'invio di email a tutti gli utenti della rubrica ed automazioni simili.

*Stack di rete*: lo stack di rete di un determinato sistema operativo indica un'insieme di caratteristiche che hanno i pacchetti creati da tale sistema. Generalmente diversi sistemi operativi hanno diversi stack di rete.

*Translation Lookaside Buffer (TLB)*: è un buffer, che l'MMU (Memory Management Unit) usa per velocizzare la traduzione degli Indirizzi Virtuali. Il TLB possiede un numero fisso di elementi della Page Table, la quale viene usata per mappare gli Indirizzi Virtuali in Indirizzi Fisici. La Memoria virtuale è lo spazio visto da un processo che può essere più grande della memoria fisica (reale). Questo spazio è segmentato in pagine di dimensioni prefissate. Generalmente solo alcune pagine vengono caricate nella memoria fisica in zone dipendenti dalla politica di Page Replacement. La Page Table (generalmente caricata in memoria) tiene traccia di dove le pagine virtuali sono caricate nella memoria fisica. Il TLB è una cache della Page Table, cioè solamente un sottoinsieme del suo contenuto viene memorizzato.

# BIBLIOGRAFIA E SITOGRAFIA


Provos Niels & Thorsten Holz: *Virtual Honeypots: from botnet tracking to intrusion detection*.
Addison Wesley Professional, 16 luglio 2007.

Clifford Stoll: *The Cuckoo's Egg*.
Poket Books, 1988.

Bill Cheswick: *An Evening with Berferd.*
http://all.net/books/berferd/berferd.html

Fred Cohen & Associates: *Deception ToolKit*.
http://www.all.net/dtk/index.html

Niels Provos: *Honeyd, a virtual honeypot daemon (Extended Abstract)*
http://www.citi.umich.edu/u/provos/papers/honeyd-eabstract.pdf

Lance Spitzner: *Definitions and Value of Honeypots.*
http://www.spitzner.net/honeypots.html

Provos Niels: *A virtual honeypot framework*.
http://www.citi.umich.edu/u/provos/papers/honeyd.pdf

Provos Niels: *Honeyd.*
http://honeyd.org/

Lance Spitzner: *The Honeynet Project.*
http://www.honeynet.org

Brien M. Posey: *An introdution to Google Hack Honeypots*
http://searchenterprisedesktop.techtarget.com/tip/An-introduction-to-Google-Hack-Honeypots

IEEE: *IEEE Standard MAC associations*
http://standards.ieee.org/develop/regauth/oui/oui.txt

Kaspersky Lab: *Kaspersky Security Bulletin: evoluzione delle minacce*
http://www.kaspersky.com/it/reading_room?chapter=207716722

Tal Garfinkel (in collaborazione con Keith Adams, Andrew Warfield, Jason Franklin): *Compatibility is Not Transparency: VMM Detection Myths and Realities*
http://xenon.stanford.edu/~talg/papers/HOTOS07/abstract.html

Kostya Kortchinsk: *Patch for VMware*.
http://honeynet.rstack.org/tools/vmpatch.c

John Scott Robin, Cynthia E. Irvine: *Analysis of the Intel Pentium's ability to support a secure virtual machine monitor*.
http://www.cse.psu.edu/~bhuvan/teaching/spring06/papers/analysis-pentium.pdf

Tom Liston and Ed Skoudis: *On the cutting edge: Thwarting virtual machine detection.* http://

handlers.sans.org/tliston/ThwartingVMDetection_Liston_Skoudis.pdf

SANS (SysAdmin, Audit, Network, Security): *The Top Cyber Security Risks.*
http://www.sans.org/top-cyber-security-risks/?ref=top20

Nmap official site
nmap.org

Sourceforge: *X probe - active OS fingerprinting tool*.
http://sourceforge.net/projects/xprobe/

Da wikipedia:

http://it.wikipedia.org/wiki/Transmission_Control_Protocol
http://it.wikipedia.org/wiki/Controllo_di_flusso
http://it.wikipedia.org/wiki/Local_Descriptor_Table
http://it.wikipedia.org/wiki/Global_Descriptor_Table
http://it.wikipedia.org/wiki/Interrupt_Descriptor_Table
http://it.wikipedia.org/wiki/Payload_(malware)
http://it.wikipedia.org/wiki/Translation_Lookaside_Buffer