Riassunto di LAS (A.A. 2018-2019)

28 giugno 2019

Indice

NOTE

Questo riassunto è stato scritto basandosi sulle slide mostrate a lezione, sul contenuto delle lezioni e su varie ricerche in rete e non è detto che tutto ciò che è racchiuso all'interno di questo documento sia completamente corretto.

Capitolo 1

Hardening di base

La sicurezza è un processo, non un prodotto che possa essere comprato, pertanto va analizzata costantemente. Uno dei pochi punti fermi è che la sicurezza si gestisce efficacemente solo vietando qualsiasi comportamento non esplicitamente consentito, e che ciò che è consentito deve essere svolto con i diritti più restrittivi compatibili con l'esecuzione del compito.

1.1 Messa in sicurezza fisica

Per prima cosa occorre pianificare la sicurezza fisica degli apparati, in modo da proteggerli sia da eventi accidentali (quali terremoti, esondazioni, ...) sia da eventi intenzionali (furti di dispositivi fisici, furti di dati, danni fisici agli apparati, ...), ad esempio scegliendo un luogo in una zona geograficamente sicura e protetto da sistemi di sicurezza adeguati ai dati che si volgiono proteggere. Per quanto riguarda le singole macchine è buona norma disabilitare tutto ciò che non è espressamente utilizzato, sia in termini di periferiche sia in termini di pacchetti sotware, che in termini di servizi.

1.2 Boot

Per andare a regime il sistema attraversa diverse fasi che corrispondono al processo di boot:

- BIOS: durante questa fase vengono individuati i dispositivi che contengono un boot loader. È possibile impostare una password di accesso, tuttavia a questo livello le password sono facilmente scavalcabili e dunque non particolamente sicure, inoltre se il boot è protetto da password l'avvio non supervisionato (ad esempio dopo una semplice mancanza di alimentazione) diventa problematico. Di conseguenza conviene far affidamento su altre misura di sicurezza del sistema

- Bootloader: si occupa di rilevare quali sistemi siano installati e di avviarne uno eventualmente passando dei parametri di avvio. Permette di gestire la modalità di ripristino e generalmente può essere protetto da password (con gli stessi problemi del BIOS). Per Linux sono presenti due grandi boot loader

  - LILO, il più vecchio, usato agli inizi della storia di Linux
  - GRUB più potente e flessibile di LILO, possiede una shell dedicata che permette di configurare la procedura di avvio (con i pro e contro che ne derivano). Permette di specificare sia una password (nella global section) per interagire con il sistema sia una password richiesta per avviare item specifici dal menù. Le configurazioni di default sono contenute all'interno del file `/etc/default/grub`

  A questo punto di pone un problema, come essere sicuri che ogni componenete software eseguito da un computer sia sicuro ed integro? Si viene a creare una catena in cui gli anti-malware verificano le applicazioni, il SO verifica gli anti-malware (in realtà questo rende l'anti-malware inutile, potrebbe fare tutto il SO), il boot loader può controllare il SO, il BIOS (assisitito da hardware speciale) può verificare l'integrità del boot loader, eccetera. Sono possibile dunequ due modalità:

  - Trusted boot (usa TPM): prima di passare ad ogni fase successiva vengono raccolte prove circa la compromission del sistema, in questo modo, se viene rilevata una violazione, il controllo è ancora in mano ad una componente sicura che può reagire.
  - Secure boot (usa UEFI): nato come interfaccia più flessibile del BIOS è diventato un piccolo SO con una partizione dedicata (partizione EFI). UEFI verifica ogni componente software prima di passare il controllo al boot loader tramite l'uso di chiavi crittografiche inserite nel firmware. Non appena viene rilevata una difficoltà viene bloccato l'intero processo di boot

- Sistema operativo: carica i device driver, anch'essi da controllare, ed avvia il processo di init

- init: gestisce i runlevel (o i traget) per coordinare l'inizializzazione e l'avvio del sistema

Capitolo 2

Sicurezza

2.1 Introduzione

Per poter parlare di sicurezza di un sistema informatico occorre anzitutto definire cosa si intende per sicurezza. In generale si dice che un sistema è sicuro quando è in grado di realizzare certi obiettivi essenziali:

- **Confidenzialità**, ovvero la garanzia che solo le persone autorizzate possono accedere al sistema ed ai dati, ed operare tramite di esso e su di essi. In generale le tecnihc epiù utilizzate per garantirla sono tecnioche crittografiche.
- **Integrità**, ovvero la garanzia che solo le persone autorizzate possano modificare il sistema e i dati in esso mantenute, e possano farlo solo all'interno dei limiti e delle autorizzazioni loro assegnate
- **Disponibilità**, ovvero la garanzia che il sistema è in grado di fornire i servizi previsti, e sia in grado di farlo in tempi e modalità "ragionevoli"

Per realizzare questi obiettivi un sistema informatico deve fornire una serie di servizi, che possono essere classificati in:

- **Identificazione** (risponde alla domanda "chi sei?"): scopo di questi servizi è identificare l'utente all'interno del sistema, da una parte per permettere l'acceso solamente agli utenti autorizzati, dall'altro per associare ad ognuno di essi certi permessi. Il modo più classico per realizzare questi servizi è l'utilizzo di uno username
- **Autenticazione** (risponde alla domanda "come mi accerto che sei tu?"): scopo di questi servizi è verificare l'identità di chi ha intenzione di accedere al sistema. Per ottenere questo risultato spesso si ricorre ad una prova che ricade in una delle tre categorie:
  - "qualcosa che sei": conferma l'identità per mezzo di un dato biometrico

  - "qualcosa che hai": conferma l'identità per mezzo di oggetto riconoscibile da parte del sistema (smart card, ...)
  - "qualcosa che sai": conferma l'identità dimostrando la consocenza di un segreto

  In Linux si può ricorrere al moduo PAM per risolvere il problema dell'implementazione dei moduli di autenticazione all'interno dei programmi. PAM offre un'API che può essere usata dai vari programmi, in questo modo diventa moto più semplice implementare nuove tecniche per l'autenticazione e tenere sotto controllo quelle già esistenti

- **Autorizzazione** (risponde alla domando "cosa puoi fare?"): scopo di questi servizi è stabilire cosa l'utente possa e non possa fare all'interno del sistema

2.2 Controllo dell'accesso

Di base il controllo dell'accesso consiste nel decidere se un soggetto possa eseguire una certa operazione su un di un oggetto. Questa relazione può essere espressa per tutto il sistema per mezzo di una matrice completa in cui le righe specificano i soggetti e le colonne gli oggetti; ogni cella contiene quali siano i permessi associati a quel soggetto per quell'oggetto. Poichè la matrice sarebbe enorme e la maggior parte delle celle sarebbe vuota conviene partizionarla:

- per soggetto (capability list): fornisce una lista di permessi per ogni soggetto all'interno del sistema, e contiene solo gli oggetti per cui il soggetto ha permessi diversi da quelli di default

- per oggetto (access control list): fornisce una lista di permessi per ogni oggetto all'interno del sistema, contiene solamente i soggetti i cui permessi sono diversi da quelli di default. Il filesystem UNIX implementa una ACL rigida in cui sono specificati solamente tre soggetti (l'utente proprietario, il gruppo proprietario, e il gruppo implicito che contiene tutti gli altri utenti). Le ACL POSIX estendono la flessibilità di autorizzazione permettendo di specificare una lista arbitraria di utenti e/o gruppi ed i relativi permessi, così come di specificare limitare tutti i permessi simultaneamente tramite una maschera

Di base sono possibili più paradigmi per il controllo dell'accesso:

- Discretionary Access Control (DAC): ogni oggetto ha un propritario che ne decide i permessi. Un tipico esempio sono le ACL

- Mandatory Access Control (MAC): necessita di una policy centralizzata, tipicamente espressa tramite una capability list, decisa da un security manager. In questo caso la proprietà di un oggetto non consente di modificarne i permessi
- Role Based Access Control (RBAC): vengono identificati dei ruoli all'interno del sistema a cui vengono assegnati i permessi. Permette di specificare i permessi per una risorsa in maniera molto precisa e contestuale, tuttavia viene solamente spostato il problema riguardante la decisione di quali permessi dare ad un certo soggetto

2.3 Autorizzazione su filesystem UNIX

Ogni file è descritto da un i-node su cui, tra le altre cose, sono memorizzate le informazioni relative all'autorizzazione. Queste informazioni comprendono un utente proprietario, un gruppo proprietario ed i permessi espressi tramite 12 bit che rappresentano i permessi standard e speciali.

I permessi standard specificano i cosa possano fare i soggetti UGO (N.B. le direcotry sono file il cui contenuto è un elenco di coppie <nome_file>-<i-node>:

- **R**:
  - per i file: lettura del file
  - per le directory: elenco dei file nella directory
- **W**:
  - per i file: scrittura sul file
  - per le directory: aggiunta, rinomina, cancellazione di file all'interno della directory (N.B. questo permesso consente di eleiminare anche i file non di proprietà di un certo utente)
- **X**:
  - per i file: esecuzione del file
  - per le directory: lookup dell'i-node (N.B. per accedere ad un file è necessario che tutte le directory del path abbiano attivo il bit X per quell'utente)

I permessi speciali invece consentono di definire comportamente particolari per quanto riguarda l'utente proprietario, il gruppo proprietario o altri utenti.

- **SUID**: per i file (ha effetto solo sui file eseguibili) fa sì che il file venga eseguito con l'identità del proprietario del file. Consente di implementare delle interfacce per gli utenti standard verso processi privilegiati (es `/usr/bin/passwd`)

- **SGID**: per i file ha lo stesso effetto del bit SUID, ma per quanto riguarda i gruppi. Per le directory consente ad un utente, appartenente al gruppo proprietario della directory, di creare in automatico file aventi come gruppo proprietario quello della della directory padre

- **sticky**: per i file eseguibili suggerisce al SO di mantenere in cache una copia del programma. Per le directory impone che tutti i file contentui siano cancellabili solo dai rispoettivi proprietari

2.4 Cenni di crittografia

Per garantire la riservatezza (così come la paternità e l'integrità) delle informazioni è possibile ricorrere alla crittografia, ovvero "a disciplina che studia la trasformazione di dati allo scopo di nascondere il loro contenuto semantico, impedire il loro utilizzo non autorizzato, o impedire qualsiasi loro modifica non rilevabile".

La sua applicazione consiste nell'utilizzo di due operazioni, una di *cifratura* ($E()$) che consente di ottenere dal testo in chiaro (m) il testo cifrato (c) ed una di *decifrazione* ($D()$) che consente di ottenere il testo in chiaro (m) dal testo cifrato (c).

$$c = E(m) \qquad m = D(c)$$

Inizialmente la sicurezza dell'algoritmo era affidata alla segretezza dello stesso, di conseguenza una volta che l'algoritmo era stato reso pubblico perdeva totalmente le sue proprietà di sicurezza. Nel 1883 Kerkhoffs enuncia tre principi per la sicurezza delle informazioni:

1. Il sistema deve essere praticamente, se non matematicamente, indecifrabile

2. Questo non deve essere segreto, deve poter cadere nelle mani del nemico senza inconvenienti, ovvero il segreto deve essere la chiave non l'algoritmo

3. La sua chiave deve essere comunicabile senza l'aiuto di note scritte, e sostituibile o modificabile a piacimento dei corrispondenti

Un buon algoritmo deve essere in grado di occultare le proprietà del testo in chiaro (ad esempio per resistere ad analisi statistiche), rendere la chiave

corretta indistinguibile dalle altre (sicurezza assoluta) oppure rendere il processo di ricerca della chiave talmente complesso da renderlo materialmente troppo oneroso (sicurezza computazionale).

La disciplina che studia la robustezza degli algoritmi crittografici è la crittanalisi; i criteri di base per cui si considera un algoritmo sicuro sono quelli che gli permettono di resistere alle diverse tipologie di attacco da parte dei crittanalisti. Questi attacchi vengono classificati nelle quattro categorie seguenti:

- **attacchi a testo in cifra noto**: è l'attacco più semplice, in cui si analizzano un certo numero di messaggi cifrati per ricavarnne delle informazioni. In generale sfruttano analisi statistiche del testo
- **attacchi a testo in chiaro noto**: in questo caso si possiede anche il testo noto di alcuni dei messaggi catturati
- **attacchi a scelta di testo in chiaro**: rispetto al caso precedente in questo tipo di attacchi si è in grado non solo di conoscere alcuni messaggi passati sul canale, ma di scegliere anche il loro contenuto
- **attacchi a scelta adattativa di testo in chiaro**: oltre a quanto previsto nel caso precedente qui si richiede anche che sia possibile scegliere il testo dei messaggi da cifrare in base ai messaggi cifrati ottenuti per quelli scelti in precedenza

A fronte di uno qualsiasi di questi attacchi sono possibili più gradi di successo, che vanno dalla deduzioni di qualche informazione utile alla decifrazione di un messaggio alla deduzione globale (permettendo la decifrazione completa di tutti i messaggi)

2.4.1 Crittografia simmetrica

L'uso di questi algoritmi prevede l'utilizzo di una singola chiave che permette sia di cifrare che di decifrare i messaggi. Esempi storici notevoli sono i cifrari a sosituzione mono/polialfabetica (ad esempio cifrario di Cesare, cifrario Atbash, ...) , il one-time-pad (cifrario di Vernam). Gli algoritmi moderni si basano sugli stessi principi di confusione e di diffusione delle informazion, usando però l'alfabeto binario. Questi cifrari sono studiati per essere computazionalmente sicuri, e la loro sicurezza risiede nella lunghezza della chiave.

2.4.2 Crittografia asimmetrica

La crittografia asimmetrica consiste nell'utilizzo di due chiavi distinte, delle quali una è in grado di cifrare il messaggio mentre l'altra di decifrarlo, idealmente non è possibile ircavare una chiave dall'altra. La crittografia

asimmetrica può essere usata sia per proteggere la riservatezza delle informazioni (si cifra con la chiave pubblic di chi deve ricevere il messaggio, che viene decifrato con la chiave privata), sia per garantire la paternità delle informazioni (si cifra con la propria chiave privata, e decifrando il messaggio con la chiave pubblica si ha la garanzia di paterità).

Queste tecniche hanno il grado vantaggio di risolvere il problema della distribuzione delle chiavi che affligge la crittografia simmetrica ed inoltre sono utili per tutte le proprietà della sicurezza. Tuttavia gli algoritmi sono molto lenti. Per questo si tende a preferire un approccio ibrido: i dativengono cifrati con la crittografia simmetrica e le chiavi vengono scambiate usando la crittografia asimmetrica.

2.4.3 Funzioni Hash

Una funzione di hash è una funzione a senso unico non invertibile che mappa una stringa di lunghezza arbitraria in una stringa di lunghezza predefinita. Un hash in cui l'estrazione degli elementi m dallo spazio U è casuale con probabilità uniforme $\frac{1}{m}$ si definisce *uniforme semplice*, in tali casi la ricerca di una chiave all'interno dello spazio ha complessità $O(1)$. Per essere aplicata in crittografia è necessario che una funzione hash rispetti tre vincoli:

- resistenza alla preimmagine: sia computazionalmente impossibile la ricerca di una stringa di input che dia un hash uguale ad un altro
- resistenza alla seconda preimmagine: sia computazionalmente impossibile la ricerca di una stringa di input che abbia un hash uguale a quello di un'altra stringa
- resistenza alle collisioni: sia computazionalmente impossibile trovare due stringhe in input che diano lo stesso hash

Poichè tuttavia la cardinalità dell'insieme degli hash possibili per una data funzione è molto minore della cardinalità dell'insieme dei possibili input, per il principio dei cassetti, ad almeno un hash corrisponderanno più testi possibili. In questo caso si parla di collisione e la qualità di una funzione di hash è misurata in base alla difficoltà di di individuare due testi che generino delle collisioni.

In crittografia queste funzioni sono utili per verificare l'integrità di un messaggio, infatti associando ad un messaggio il suo hash si permette la rilevazioni di compromissioni del messaggio, oppure per la creazione di firme digitali per garantire la paternità di un messaggio.

2.5 Vulnerabilità ed attacchi

2.5.1 Classificazione delle minacce

In relazione ai tre obiettivi fondamentali della sicurezza si tende a dividere le minacce in più categorie:

- Intecettazione (*snooping*): questa categoria indica sia i rischi relativi all'intercettazione non autorizzata di informazioni che l'accesso non autorizzato a dati e documenti. Può essere implementata con diverse tecniche, generalmente passive, che vanno dall'intercettazione eseguita con mezzi fisici all'uso di software specializzati (*sniffer*). In generale si risponde a questo tipo di minacce tramite tecniche basate sulla crittorafia che consentano di cifrare il traffico

- Alterazione: questa categoria copre tutte quelle minacce che hanno come scopo la modifica non autorizzata di dati ed informazioni. Di norma gli attacchi che ricadono in questa categoria fanno uso di tecniche attive, e possono essere contrastati tramite tecniche volte a garantire l'integrità dei dati

- Mascheramento (*spoofing*): questa categoria racchiude tutte quelle minacce che si basato nel presentarsi con un'identità differente da quella reale (ad esempio phishing, spoofing, ...). Le tecniche usate possono essere sia passive che attive, e possono essere contrastate tramite l'uso di tecniche che preservino l'integrità dei dati e consentano l'autenticazione

- Blocco: appartengono a questa categoria tutti quegli attacchi che hanno come scopo l'impedimento od il ritardo dell'erograzione di un servizio. Le tecniche usate sono sempre di carattere attivo

2.5.2 Vulnerabilità del codice

Una prima causa di vulnerabilità è il codice. Esse possono presentarsi sia perchè il codice prodotto è di bassa qualità sia perchè i produttori hanno reazioni inadeguate alla scoperta di una vulnerabilità. Queste vulnerabilità possono essere divise in due macro-categorie:

- Attacchi data-driven: ad esempio SQL injection, cross site scripting, stack overflow, buffer overflow

- Attacchi basati su logiche di accesso errate

Buffer overflow Il buffer overflow è una condizione di errore che si verifica runtime quando in un buffer di una data dimensione vengono scritti dati di

dimensione maggiore. Quando questoaccade viene sovrascritta parte dei dati adiacenti al buffer cn effetti differenti a seconda di dove si trovi il buffer.

In caso il buffer si trovi all'interno dello stack si parla di stack overflow, in questo caso tra i dati a poter essere sovrascritti c'è anche il return address di una funzione. Questo potrebbe puntare ad una sezione in cui sia stato iniettato del codice maligno mettendolo in esecuzione, oppure ad una sezione di memoria non accessibile sollevando un'eccezione che, se non gestita, potrebbe far crashare il programma minando la disponibilità di un servizio.

Shellcoding Rappresenta una tecnica per sfruttare i buffer overflow: tramite stack overflow viene iniettato del codice maligno ed un indirizzo di ritorno che punti al codice iniettato. L'obiettivo comune di questo tipo di attacchi è quello di aprire una shell, tipicamente con privilegi di root, tramite cui l'attaccante possa avere accesso al sistema.

Una prima contromisura all'inizezione di codice maligno è l'ASLR (Address Space Layout Randomization). Essa prevede che l'indirizzo (logico) di partenza dei segmenti sia casuale, pertanto un attaccante non ha modo di trovare un indirizzo plausibile a cui puntare ed in cui inserire codice maligno (N.B. Solo alcune distribuzioni Linux hanno la patch di siceurezza associata all'interno del kernel). In alternativa si può fa affidamento sugli stack non eseguibili (funzionalità fornita dall'hardware ma che dev'essere supportata dal SO). Un'evoluzione consiste nella tecnologia W^X, che consiste nel marcare ogni pagina o come eseguibile o come scrivibile.

Rootkit Per Rootkit si intende un sistema software in grado di nascondere la compromissione di un sistema. Può essere implementata a livello di applicazione, ad esempio tramite trojan od utility di sistema munite di backdoor, a livello kernel, tramite moduli o driver maligni che consentano di controllare la macchina attaccata, oppure a livello di libreria, tramite la sosituzione di librerie ufficiali con librerie manomesse.

In generale i rootkit sono molto difficili da rilevare proprio perchè il loro primo obiettivo è quello di mascherare la loro presenza modificando gli stessi strumenti utilizzati per la rilevazione. Si può optare per una ricerca euristica, tipica degli antivirus, oppure tramite una verifica della memoria da parte di un sistema pulito e sicuro.

2.6 Monitoraggio ed intrusion detection

Se da un lato la sicurezza si basa sul soddisfacimento di tre obiettivi (confidenzialità, integrità, disponibilità) che passa attraverso l'identificazione, l'autenticazione e l'autorizzazione degli utenti è normale che esistano errori all'interno dei sistemi che non permettano a queste procedure di lavorare correttamente o di scavalcarle. In particolare per quanto riguarda la rete

non è presente alcune meccanismo di identificazione ed autorizzazione a livello dei protocolli TCP/IP. Gli IDS nascono dala consapevolezza di questi limiti e partono dal presupposto che un usualmente una violazione del sistema comporta la manifestazione di comportamenti anomali che possono essere rilevati e fermati. Di conseguenza un IDS non è un sosituto delle normali procedure di autenticazione ma un ulteriore strumento da utilizzare per proteggere adeguatamente il sistema.

In generale si hanno due metodologie principali per riconoscere le anomalie del sistema. La prima è la "*misuse detection*" che si occupa, una volta individuati, di riconoscere gli usi illectiti delle risorse; generalemnte questo viene fatto attraverso una serie di regole che definiscano quali siano i comportamenti anomali da controllare, pertanto è necessario che i misuse siano completamente noti (riconoscimento *signature based*). In alternativa si può ricorrere all'*anomaly detection* che permette di definire un comportamento normale del sistema identificando le deviazioni da esso; in questo caso la difficoltà sta nell'individuare correttamente quale sia un comportamento normale e oltre quali soglie di deviazione lanciare un allarme.

Una seconda classificazione degli IDS è quella che distingue quelli che eseguono operazioni in tempo reale (*online*) da quelli che invece operano *offline*, analizzando i dati in un momento successivo. Gli IDS online (come `snort`) in genere lanciano un allarme non appena accadono certe situazioni, IDS di questo tipo possono inoltre occuparsi di reagire automaticamente a delle minacce in atto (in questo caso vengono definiti IPS). Tuttavia le reazioni automatiche degli IPS potrebbero essere sfruttate da un attaccante che conosca la tipologia del sistema attaccato, infatti potrebbe far scattare di proposito le contromisure dell'IPS in modo da minare la disponibilità del sistema. In genere il limite degli IDS online sta nella grande quantità di risorse necessarie ai controlli. Gli IDS offline (come `aide`) invece possono analizzare i dati in maniera molto più approfondita, non avendo problemi di tempo, trattando anche moli di dati non analizzabili in tempo reale. Tuttavia in questo caso l'analisi, per definizione, viene effettuata dopo che l'attacco è avvenuto, ma possono comunque fornire informazioni utili per determinare il tipo di minaccia e porre rimedio.

L'ultima classificazione degli IDS vede la distinzione tra quelli che monitorano una singola macchina (*host based IDS*) e quelli che monitorano il traffico di rete (*network based IDS*). Gli HIDS permettono di monitorare completamente una macchina (possono controllarne il filesystem, esaminare il log, verificare i metadati dei file, ...). Generalemente sono più economici dei NIDS e presentano un minor tasso di falsi positivi, tuttavia essi richiedono l'installazione di un agente sulla macchina, agente che va protetto a sua volta, ed inoltre non permettono di scansionare eneti/pacchetti che non lasciano traccia sul filesystem. Tipicamente la rilevazione fa uso di un *integrity checker*, ovvero di un sistema che permette di rilevare delle discrepanze tra lo stato "normale" di un sistema e lo stato attuale. I NIDS (i più diffusi

sono snort, suricata e bro) invece permettono di controllare l'intero traffico che attraversa una rete per mezzo di una o più sonde posizionate nella rete stessa. Il grosso problema dei NIDS sta nei falsi positivi, infatti il traffico può essere correttamente identificato solamente se posto in relazione con mittente e destinatario. Infine non possono monitorare il traffico cifrato.

In genere perchè il monitorggio sia efficace occorre conoscere le vulnerabilità dei sistemi, pubblicate su varie mailing list o su vari archivi, e testarne periodicamente la sicurezza. Questo può essere fatto simulando un attacco usando le stesse tecniche che userebbe un attaccante, a questo proposito uno strumento diffuso è OpenVAS. Le sue caratteristiche principali sono la sua architettura distribuita, che permette di separare i vari moduli di cui è composto, e l'ampio database delle vulnerabilità (completo di procedura di verifica, piattaforme colpite e descrizione delle stesse) aggiornato quotidianamente.

Portscanner Un portscanner è un programma in grado di rilevare quali servizi di rete sono attivi su una macchina o in una intera rete, eseguendo una scansione sulle porte (in genere TCP, ma anche UDP) per identificare quali di esse sono correntemente utilizzate da un qualche demone di sistema per fornire servizi. Generalemente si tratta di uno strumento importante per verificare in maniera effettiva la propria rete, dato che in presenza di una macchina compromessa non si può avere nessuna fiducia nei programmi diagnostici eseguiti su detta macchina, che potrebbero essere stati a loro volta compromessi. Infatti, in caso di violazione di una macchina, uno dei primi passi eseguiti di solito da chi ha effettuato l'intrusione è quello di predisporre una backdoor e poi modificare lo stesso kernel o i programmi che potrebbero rilevarne la presenza installando un rootkit. Per questo motivo è importante effettuare il controllo delle porte aperte di un sistema, o di una rete, da una maccina sicura oppure da un sistema live avviato da un CD-ROM. Un esempio di portscanner è `nmap`.

Sniffer Uno sniffer è un programma usato per leggere ed analizzare il traffico in transito su di una rete che arriva su una macchina. Sebbene sia difficile analizzare tutto il traffico di una rete, dato che gli switch inoltrano il traffico solamente al destinatario, essi permettono di effettuarne una prima analisi. Se installato su di una macchina specifica è in grado di ascoltare tutto il traffico e di vsualizzarne le caratteristiche. Esempi di sniffer sono `tcpdump` e `wirewhark`.

2.6.1 Log di sistema

È opportuno tener traccia di ogni operazione effettuata all'interno del sistema tramite dei log (tipicamente mantenuto du dei file di testo), in particolare

per rilevare attività malevole o illecite. La sicurezza dei log stessi va garantita, ad esempio tramite replicazione e tramite l'uso di un integrity checker. Inoltre è sempre possibile inviandrli ad un server sicuro (che però va protetto in quanto diventa un bersaglio appetibile) in modo da porteli gestire in maniera centralizzata. Tuttavia non è sufficiente mantenere un elenco di eventi, poichè esso va anche analizzato (software diffusi sono `Logwatch` e `Swatch`) in modo da estrarre correlazioni tra eventi e relazioni temporali.

In Linux soluzioni comuni sono `Rsyslog` e `Syslog-ng`. Anche `systemd` offre attività di logging tramite il demone `journal` con il vantaggio che, essendo attivo dal boot, il monitoraggio può partire dalle primissime fasi di avvio del sistema.

syslog Ogni messaggio viene etichettato da una coppia che ne identifica l'origine e la priorità (`<facility>.<priority>`), in modo da comprendere quale sia la destinazione (come un file specifico, STDIN di un programma, un server syslog remoto, ...) da assegnare. Le configurazioni sono mantenute all'interno del file `/etc/syslog.conf`, dove ogni riga corrisponde ad una regola.

rsyslog Rappresenta l'evoluzione di syslog, ha una struttura modulare in modo da caricare solamente le funzioni necessarie. Permette l'utilizzo di modalità di output avanzate per definire le destinazioni in maniera più flessibile, ad esempio distinguendo i messaggi sulla base di alcune proprietà o su alcune espressioni. Le configurazioni sono mantenute all'interno del file `/etc/rsyslog.conf` e nella cartella `/etc/rsyslog.d/`, dove ogni riga corrisponde ad una regola.

syslog-ng Permette un grado ancora maggiore di flessibilità. L'input è compatibile con i log di `syslog`, i log di `journald`, e l'output può essere ridiretto anche verso database relazionali.

2.6.2 SIEM

Rappresentano tutti quegli strumenti, politiche, e procedure per la gestione integrata delle fonti di informazione e degli incidenti. I SIEM integrano le funzionalità offerte dai SEM (monitoraggio e gestione real-time degli eventi che accadono all'interno di una rete) e SIM (automatizzazione del processo di raccolta e gestione dei log non in tempo reale) al fine di poter combinare all'analisi svolta in tempo reale degli eventi, la possibilità di fornire report inerenti ai dati raccolti, rispondendo alle esigenze di incident response, compliance e di analisi forense. Le principali attività riguardano generalmente la raccolta dei dati, parsing e normalizzazione dei dati, correlazione dei dati, reporting dell'ambiente. Tipicamente sono utilizzati per tracciare le autenitcazioni ad un sistema (in modo da rilevare accessi non autorizzati), rilevare

di malware, monitorare le connessioni, osservare le violazione delle politiche interne di un sistema e rilevare i tentativi di attacco. Un esempio notevole è OSSEC

2.7 Installazione software

Il ciclo di vita del software installato su di un computer comprende installazione, aggiornamento e disinstallazione; in ognuna di queste fasi possono presentarsi varie problematiche riguardanti ad esempio il soddisfacimento di certi prerequisiti sotware/hardware e la configurazione.

Installazione L'installazione può avvenire a partire dai file sorgente, che vanno compilati correttamente per l'architettura di interesse, oppure a partire da binari precompilati, che possono essere "semplicemente" opiati nel posto giusto. In ogni caso occorre porre attenzione al soddisfacimento delle dipendenze del software. Tipicamente (per quanto riguarda Linux) l'installazione può essere sia manuale (tramite riperimento autonomo dei sorgenti/binari) che assistita (tramite l'ausilio di un package manager).

Aggiornamento Quando si aggiorna un pacchetto presente sul sistema occorre porre attenzione ai potenziali problemi derivanti da:

- prerequisiti/dipendenze: potenzialmente il grafo delle dipendenze è complesso ed articolato, ma perchè il software funzioni correttamente tutte le dipendenze devono essere soddisfatte correttamente. In più potrebbero venir modificate le interfacce esposte dal programma, cusando malfunzionamenti in altri software; di conseguenza tutti gli aggiornamenti vanno testati per controllare che possano coesistere con il software restante. In questo ultimo caso potrebbe essere necessario far coesistere due versioni differenti dello stesso pacchetto e quindi gestire manualmente i binding delle librerie dinamiche.

- configurazione: potezialmente la nuova versione del software comporta modifiche ai file di configurazione

Disinstallazione Presenta gli stessi problemi dell'aggiornamento per quanto riguarda la gestione delle dipendenze, infatti altri pacchetti potrebbero dipendere da quello che si ha intenzione di rimuovere. Questa situazione è molto difficile da prevedere e da gestire manualmente, pertanto conviene sfruttare le funzionalità offerte dai package manager circa la gestione del grafo delle dipendenze.

2.7.1 Virtual environment

Per ottenere isolamento di software è possibile utilizzare delle VM al fine di ottenere sistemmi preconfigurati che isolino i diversi ambienti di esecuzione, tuttavia queste spesso sono eccessivamente pesanti per il servizio che forniscono (impongono l'installazione di SO intero e limitano l'accesso alle risorse). Una soluzion alternativa consiste nell'isolare l'ambiente di esecuzione di certi processi facendo sì però che condividano il sistema operativo. In questo caso l'isolamento non è completo e non possono essere eseguite applicazioni incompatibili con il sistema, ma l'accesso alle risorse hardware è diretto e le risorse condivise non sono replicate.

Il kernel Linux fornisce tre funzionalità:

- **control groups** (Cgroups): permettono di controllare la quantità di risorse che un processo può consumare e l'accesso ai device
- **namespace**: permettono di mostrare ai vari processi istanze logiche differenti della stessa risorsa reale. Per ottenere questo risultato ogni processo vieve all'iterno di un namesapce di più tipi
- **union-capable filesystems**: filesystem Linux (e BSD) che permette di simulare l'unione di più filesystem esistenti

Utilizzando il partizionamento delle risorse si possono definire i *container*, che permettono di definire in modo coerente sia cgroup che namespace specificando come le risorse reali vadano esposte all'interno del sistema.

2.8 Gestione dei servizi e dei processi

Anche dopo un'installazione minimale sul sistema sono attive decine e decine di processi, anche se tutti fossero necessari è utile conoscerne l'origine e comprendere quali terminare per guadaganre risorse disponibili. Ci sono tre fonti principali di processi (oltre agli utenti):

- Pianificatori periodici e sporadici
- Demoni di gestione degli eventi
- Procedure di avvio del sistema

Processi periodici Sono in generale processi lanciati dai demoni `cron` e `at`

Demoni per la gestione degli eventi `D-Bus` è un'architettura di IPC nata per uniformare la comunicazione tra gli elementi delle interfacce desktop ma che fornisce anche sistemi per facilitare la gestione del ciclo di vita dei processi. I file di configurazione sono contenuti all'interno di `/etc/dbus-1/`.

`udev`, ora parte di systemd, è un device manager il cui scopo primario è quello di gestire sia gli evelementi contenuti all'interno della directory `/dev` sia quello di gestire gli eventi scatenati dall'inserimento o dalla rimozione di un dispositivo. `udev` carica i moduli kernel necessari parallelamente per consentire performance migliori, tuttavia questo significa che i moduli non sono caricati sempre nello stesso ordine ad ogni boot, causando potenziali cambiamenti di nome di un dispositivo tra due accensioni diverse del sistema. I file di configurazione per goni evento sono contenuti all'interno di `/etc/udev/rules.d`.

Procedure di avvio del sistema `init` è il primo processo lanciato dal kernel. Esso è deputato alla gestione dei runlevel (ovvero stati di funzionamento del sistema definiti dal sottoinsieme di processi attivi), all'orchestrazione della sequenza corretta di eventi per raggiungere un certo runlevel e dello spegnimento coordinato del sistema. Sono presenti tre varianti principali:

- `sysVinit`: contenuto all'interno di `/sbin/init` è l'originale demone di avvio di SystemV Unix. All'interno del file di configurazione `/etc/inittab` sono contenuti i dati necessari a definire quli processi debbano girare in un dato runlevel (script contentuti all'interno delle directory `/etc/rcN.d`, con N corrispondente ai vari runlevel) e quali debbano essere riavviati in caso si arresto.

- `upstart`: prodotto da Canonical, costituisce un rimpiazzo per init basato sulla logica ad eventi. permette l'inizializzazione dei sottositemi parallela non bloccante e la gestione omogenea di tutti gli eventi asincroni. Ogni avvenimento genera un evento ed è scatenato da un è evento, in questo modo il sistema si avvia praticamente autonomamente. La directory `/etc/init` contiene un file per definire ogni attività e il demone init continua ad orchestrare il sistema.

- `systemd`: è un gestore di sistema e di servizi per Linux con notevole capacità di parallelizzazione. Fa uso di socket e di `D-Bus` per l'avvio dei demoni, offre l'avvio su richiesta dei demoni e tiene traccia dei processi tramite `cgroups`. Permette logging precoce e la specifica di dipendenze tra i servizi. Si propone di sostituire vari demoni di sistema come `init`, `udev`, `inetd`, `syslog`, fornendo una soluzione unificata per la gestione di tuti questi servizi. Permette di definire varie `control unit` i cui nomi seugono la convenzione `name.type`, dove type può assumenre vari valori tra `Service` (per il controllo ed il monitoraggio

dei demoni), `Socket` (per l'attivazione di canali IPC di qualsiasi tipo), `Target` (per definire gruppi di unit, rimpiazza il concetto di runlevel), `Device` (permette di creare punti di accesso ai dispositivi creati dal kernel in seguito ad interazioni con l'hardware), ... `Systemd` rimpiazza il concetto di runlevel con il concetto di target, per raggruppare delle unità sulla base delle dipendenze e per fornire nomi standardizzati. Questo fa sì che la gestione delle dipendenze diventi molto più robusta e automatizzata; ogni unit parte non appena sono rispettati certi vincoli espressi dalle direttive contenute all'interno dello unit file

2.9 Sicurezza delle reti

Sono possibili vari tipi di attacchi che sfruttano le debolezze dei protocolli di rete (sia a livello applicativo, che a livello di rete), ad esempio dirottamenti del traffico attraverso sistemi compromessi.

DNS spoofing Quando un utente effettua una query DNS l'attaccante la cattura e manda alla vittima una risposta fasulla, differente da quella che avrebbe fornito il DNS. Questo attacco può anche essere portato a atermine tramite *pharming*, quanto la vittima visita un sito compromesso uno script esegue una riconfigurazione del DNS locale del router redirigendo tutte le successive query DNS verso un name server scelto dall'attaccante.

Questo attacco è molto semplice ma richiede l'accesso ad un name server e la possibilità di modificare direttamente alcuni record. L'impatto di questa categoria di attacchi può essere mitigato tramite l'uso di HTTPS.

2.9.1 HTTPS

Https è un protocollo per la comunicazione sicura attraverso reti internet, esso consiste nell'utilizzo del protocollo HTTP all'interno di una connessione cifrata tramite crittografia asimmetrica da TLS (o dal predecessore SSL) fornendo autenticazione dei siti web visitati, protezione della privacy durante la comunicazione ed integrità dei dati. Il protocollo garantisce una protezione accettabile da eavesdropper e da attacchi della tipologia *man in the middle*. Grazie a TLS vengono cifrati tutti i dati contenuti nei messaggi HTTP, quali URL, parametry della query, cookies, header della connessione.

Certification Authority / Certificati digitali Una certification authority è un soggetto terzo **fidato** abilitato ad emettere un certificato digitale, ovvero un documento elettronico che attesta l'associazione univoca tra una chiave pubblica e l'identità di un soggetto. L'infrastruttura PKI è costituita da varie CA organizzate gerarchicamente al cui vertice si trova una CA-root, il cui certificato solitamente è auto-firmato, che certifica le sub-CA. Una

CA ha, tra i suoi compiti, il rilascio dei certificati, previa identificazione e verifica del richiedente, la manutenzione del registro delle chiavi, la revoca/sospensione dei certificati in caso di abusi/falsificazioni, la pubblicazione di liste sempre aggiornate di certificati.

All'interno di un certificato digitale sono contenute varie informazioni tra cui la chiave pubblica del proprietario del certificato, l'identità del proprietario. Esse sono digitalmente firmate per garantirne l'autenticità e l'integrità da parte della CA. Il formato più comune per i certificati è definito dallo standard *X.509*, che però, essendo molto generale, ha la necessità di essere ulteriormente specificato per i vari casi d'uso.

SSL/TLS Sono protocolli crittografici progettati per fornire sicurezza nelle comunicazioni attraverso reti internet, in particolare per fornire caratteristiche di confidenzialità ed integrità dei dati. La connessione è privata grazie all'uso della crittografia simmetrica, la cui chiave viene scambiata durante l'handshake iniziale (a sua volta protetto dalla crittografia asimmetrica). Inoltre grazie all'utilizzo della crittografia asimmetrica durante l'handshake iniziale viene garantita l'autenticazione delle parti comunicanti. L'handshake è composto da più fasi:

- Negoziazione: vengono concordate le caratteristiche della comunicazione, come protocollo di comunicazione, protocollo crittografico
- Scambio dei certificati
- Inizializzazione della connessione cifrata

A causa di alcune vulnerabilità, sia a livello di protocollo sia a livello di implementazione, SSL è stato sosituito con TLS.

2.9.2 Sicurezza a livello di rete

Il protocollo IP non può garantire nessuna proprietà di sicurezza per nessuna parte del pacchetto.

IP hijacking Si opera in modo tale da informare internet che la rotta per una data subnet passa attraverso la propria rete. Questo è possibile perchè chiunque teoricamente può diffondere informazioni circa la viabilità delle rotte in internet e permette di realizzare vari tipi di attacco più o meno dannosi (DoS, man in the middle, spamma e fuggi).

IPSec IPSec non è un protocollo singolo ma un insieme di algoritmi per la sicurezza ed un framework per la negoziazione degli algoritmi che permettono di ottenere autenticazione e crittografia direttamente a livello di rete. IPSec ha varie applicazioni tra cui l'intreconnessione sicura di reti remote

attraverso internet e l'accesso sicuro di client ad una rete privata. Rispetto ad altre soluzioni IPSec presenta il vantaggio di essere trasperente alle applicazioni, tuttavia lo stack di IPSec è differente da quello standard di TCP/IP, pertanto la macchina va configurata per poter usare questo stack specifico invece di quello standard.

IPSec è basato su tre componenti:

- **AH** (Authentication Header): fornisce un servizio di autenticazione dei pacchetti
- **ESP** (Encapsulating Security Protocol): fornisce un servizio di autenticazione e cifratura dei pacchetti
- **IKE** (Internet Key Exchange): fornisce un servizio di negoziazione dei parametri necessari al funzionamento dei precedenti componenti

Nel funzionamento di IPSec giocano un ruolo chiave le Security Association (SA) che identificano un canale di comunicazione unidirezionale definito da mittente, destinatario, protocollo utilizzato (AH o ESP), modalità di funzionamento (tunnel o trasporto) ed un *Security Parameter Index* (SPI, usato per distinguere tutti i canali aventi gli stessi estremi e lo stesso protocollo). Essendo una SA unidirezionale ne servono sempre due per definire un canale di comunicazione bidirezionale. La modalità di funzionamento *transport* prevede la comunicazione diretta tra due stazioni ciascusa delle quali tratta i pacchetti e li spedisce tramite IPSec; la modalità tunnel invece prevede la presenza di almeno un router (il cosiddetto *Security Gateway*) che faccia da tramite per la comunicazione con le macchine poste nella rete per la quale agisce da gateway, questa modalità è quella tipica delle VPN.

Il protocollo AH serve solamente per autenticare i pacchetti (ad eccezione dei campi mutabili). In caso venga usato in modalità trasporto ci si limita a frapporre tra l'header IP ed il payload l'intestazione di AH, che contiene tutti i dati relativi all'autenticazione del pacchetto; il pacchetto risultante mantiene l'header originale, anch'esso autenticato nei campi non mutabili. In caso venga usato in modalità tunnel invece si crea una nuova intestazione IP, che conterrà solamente gli estremi del tunnel, e si autentica integralmente il pacchetto originale (i campi mutabili della nuova intestazione non vengono autenticati), la nuova intestazione viene posta in testa al pacchetto, seguita dell'intestazione di AH e infine dal pacchetto originale replicato integralmente; quando il pacchetto verrà ricevuto dal security gateway esso provvederà a controllare l'autenticità del pacchetto ricevuto, a rimuovere l'intestazione usata per la trasmissione ed infine ad inoltrare il pacchetto al destinatario.

Qualora si voglia cifrare il contenuto del pacchetto invece occorre ricorrere all'uso del protocollo ESP. In caso si usi la modalità trasporto ci si limita a cifrare il payload del pacchetto IP interponendo tra l'header IP ed

il payload l'intestazione di ESP; contestualmente viene fornita autenticazione sull'intero contenuto del pacchetto (ad esclusione dell'header IP). In caso venga usata la modalità tunnel invece si genera una nuova intestazione IP ed il pacchetto originale viene cifrato integralmente, in questo modo non si potranno vedere gli indirizzi originali (che possono così essere anche indirizzi di reti private). In ogni caso in coda al pacchetto vengono aggiunti due footer.

Sebbene IPSec sia uno standard usato da molto tempo per la creazione di VPN la sua diffusione è molto limitata a causa dei problemi derivanti dall'interazione con il NAT. AH semplicemente non può funzionare in presenza di NAT in quanto esso autentica l'intero pacchetto, compresi gli indirizzi di sorgente e destinazione, che vengono modificati all'attraversamento di un NAT portando ad un fallimento del controllo sull'integraità del pacchetto. Con ESP il problema è più sottile e riguarda il checksum dei pacchetti TCP ed UDP, in questo caso non vi sono problemi di autenticazion in quanto l'header IP esterno non viene mai autenticato, tuttavia il checksum del pacchetto è calcolato sugli indirzzi originali e quindi se il pacchetto è cifrato con IPSec questo valore non pò essere aggiornato facendo fallire i controlli sugli errori a destinazione. Per risolvere qusti problemi è stata proposta un'alternativa al NAT classico chiamata NAT-T che prevede che ogni volta che un pacchetto raggiunge un NAT esso venga incapsulato all'interno di un pacchetto UDP per poi venir inviato tramite lo stack TCP/IP standard. Questa soluzione tuttavia pone dei problemi per quanto riguarda l'efficienza e rende l'utilizzo di IPSec equivalente a livello pratico a quello di altre soluzioni come OpenVPN.

2.9.3 Firewall

Un firewall può essere paragonato ad una porta blindata che divide l'interno da proteggere dall'esterno da cui provengono delle minacce, in questo senso un firewall permette di bloccare gli accessi indebiti alla rete. Così come una porta blindata un firewall può bloccare certe minacce, ma diventa inutile nel momento in cui vengono lasciate altre falle aperte, pertanto è buona norma porlo al di sopra di un sistema sicuro in modo da riderre le vulnerabilità del firewall stesso.

Le tecniche di controllo prevedono l'esame di:

- Traffico: esaminare di indirizzi, porte ed altri indicatori relativi al tipo di indirizzo che si vuole esplicitamente autorizzare.
- Direzione: discriminare il traffico in base alla direzione a parità di indirizzi e porte
- Utenti: differenziare il traffico in base a chi lo genera

- Comportamento: valutare come sono usati i servizi ammessi per identificare le anomalie

Tipologia Si identificano tre tipi fondamentali di firewall:

1. **Packet filter**: sono in grado di esaminare unicamente l'header dei pacchetti applicando in serie un elenco di regole, generalmente raggruppate in degli elenchi corrispondenti a punti di controllo differenti (pacchetti in ingresso, in uscita, ...), del tipo "se condizione allora azione". Normalmente la prima condizione ad essere verificata decide il destino del pacchetto ed interrompe la scansione dell'elenco. Di base le azioni sono accettare il pacchetto o scartarlo, ma possono anche essere intraprese azioni più complesse come loggare i dettagli del pacchetto oppure modificarlo in qualche modo. Se nessuna regola è applicabile viene attivata una policy di default. Questo tipo di firewall porta con sè alcuni vantaggi, ad esempio è molto veloce e semplice (tanto che è implmentato in moltissimi router) ed è trasparente all'utente (non occorre modificare il sistema in nessun modo, il firewall si occupa autonomamente di scansionare il traffico di rete); tuttavia le regole sono spesso di basso livello, facendo sì che per ottenere comportamenti complessi occorra costruire set di regole molti articolati, e mancano del supporto alla gestione degli utenti. Inoltre i PF presentano alcune vulnerabilità e limitazioni:

   - i pacchetti frammentati possono comportarsi in maniera non prevedibile (si potrebbe riassemblare ogni pacchetto all'interno del firewall, ma questo fa sì che il carico computazionale creasca molto rapidamente)
   - poichè i controlli vengono effettuati solamente sulla base dei dati presenti all'interno degli header i PF soffrono in presenza di spoofing. Per limitare i problemi è necessario controllare la coerenza tra gli indirizzi e le interfacce e controllare certi indirzzi particolari (indirizzi multicast, illegali, broadcast, riservati, ...)
   - il filtragio stateful non è in grado di lavorare in presenza di protocolli che negoziano l'apertura di porte dinamicamente (ad esempio connessione dati FTP)
   - non è possibile implementare difese conrto attacchi data-driven, poichè viaggiano all'interno del payload del pacchetti

   Tipicamente i PF sono stateless, tuttavia in alcuni casi è possibile usufruire di PF stateful; essi sono in grado di analizzare il traffico sulla base dei pacchetti già visti. È possibile anche avere dei *Multilayer protocol inspection firewall* che sono in grado id garantire la corenza del protocollo tenendo traccia dell'intera storia della connessione

2. **Application-level gateway** (anche chiamato *proxy server*): può essere definito come un man in the middle" buono, in grado di propagare il traffico verso i server effettivi. In questo modo è in grado di comprendere i protocolli applicativi e di controllare anche il payload (ad esempio per bloccare spam/virus per quanto riguarda i server di posta o per bloccare malware/phishing per quanto riguarda i server web). Tuttavia i firewall appartenenti a questa tipologia sono molto più pesanti di un PF e specifici per un singolo protocollo applicativo. Poichè agisco come server nei confronti dei client non è detto che siano trasparenti, ma è possibile che sia necessario un minimo livello di configurazione.

3. **Ciruit-level-gateway**: spezzano la connessione a livello di trasporto diventando endpoint del traffico ed inoltrando i payload senza esaminarli. Tipicamente sono utilizzati per determinare quale sia il traffico ammissibile dall'interno verso l'esterno. I firewall appartenenti a questa categoria hanno il vantaggio di poter essere configurati per essere trasparenti agli utenti per autorizzare solamente le connessioni fidate, inoltre possono agire da intermediario generico (non sono limitati ad un singolo protocollo) e possono essere estesi con altri tipi di firewall per ottenere servizi più complessi. Tuttavia le regole sono limitate ad indirizzi, porte ed utenti.

SOCKS SOCKS (Socket Secure) è un protocollo di livello 5 per lo scambio di pacchetto tra un client ed un server per mezzo di un proxy server. In aggiunta fornisce l'autenticazione dei pacchetti, pertanto solamente gli utenti autorizzati sono in grado di accedere al server. Questo protocollo è lo standard de facto per l'implemntazione dei CLG.

Tor Il protocollo di Tor permette di realizzare connessioni cifrate in cui il legame tra chi effettua richieste e il contenuto delle stesse è profondamente oscurato. Per prima cosa si ottiene un elenco di nodi Tor da un directory server, successivamente ne vengono scelti alcuni casualmente per creare un percorso tra mittente e destinatario su cui inviare un messaggio cifrato a "cipolla", ad ogni hop viene rimosso uno strato di cifratura fino a che il server non è in grado di leggere il traffico in chiaro. I nodi Tor formano un overlay rispetto alla rete internet, pertanto è possibile che tra un nodo e l'altro il traffico attraversi vari router, tuttavia questo non presenta problemi grazie ai vari livelli di crittografia.

Ogni ralay conosce solamente il nodo prima di lui e quello dopo, pertanto viene reso molto difficile scoprire chi sia il vero mittente. Tuttavia il protocollo presenta alcune debolezze intrinseche:

- è possibile correlare il traffico dei vari nodi per capire quale percorso facciano i pacchetti, sebbene siano cifrati

- gli exit node vedono il traffico in chiaro, è sebbene non siano in grado di capire automaticamente chi sia il mittente i payload dei pacchetti potrebbe contenere informazioni ben più rilevanti ai fini dell'identificazione

- è sufficiente che all'iterno della rete vi sia anche un solo nodo compromesso per compromettere l'intera rete

Collocazione I firewall possono essere collocati o su di un sistema dedicato a far girare un software firewall (collocazione **Bastion Host**), che tipicamente è un ALG od un CLG in quanto i PF sono generalmente integrati all'interno dei router, oppure direttamente su una macchina host, permettendo di controllare con repcisione il traffico in relazione alle applicazioni che lo generano, ma perdendo la centralizzazione della configurazione.

Alcune configurazioni tipiche di firewall sono:

- screened single-homed BH: un PF garantisce che solo il BH, che implementa un ALG, possa comunicare con l'esterno. Questa configurazione consente un doppio filtraggio (sia da parte del PF che da parte dell'ALG) facendo sì che sia necessario compromettere due sistemi per prendere il controllo dell'intera rete

- screened dual-homed BH: simile al caso precedente, tuttavia in questo caso il BH si trova a cavallo di dure reti differenti, permettendo la creazione di una rete demilitarizzata in cui inserire i servizi accessbili dall'esterno. Nella rete interna è possibile inserire i client che a questo punto si trovano all'interno di una rete separata e sicura, il cui traffico in uscita tuttavia deve per forza fluire attraverso il BH

- screened subnet: permette di rafforzare la separazione tra interno ed esterno nascondendo completamente l'esistenza della rete interna ma consente al router interno di inoltrare il traffico della rete privata senza passare dal BH

Capitolo 3

Alta disponibilità

La disponibilità (o availability) di un sistema è il rapporto tra il tempo durante cui esso eroga correttamente i servizi (uptime) rispetto al tempo per cui ci si attende che lo faccia (tempo di osservazione)

Generalmente la disponibilità viene indicata tramite il numero di 9 nella percentuale di uptime, ad esempio una disponibilità del 99,99% viene detta "a quattro 9". Questo valore è un indicatore sintetico di alcune caratteristiche del sistema, ma non è un indicatore esaustivo. All'interno del contratto sul livello di servizio ("Service Level Agreement") possono essere specificati vincoli più stringenti, ad esempio sulla distribuzione del downtime in frazioni di una certa dimensione nell'arco di un certo periodo di tempo, oppure sulle caratteristiche del servizio di assistenza da effettuare.

Per prima cosa il sistema deve essere inserito all'interno di un ambiente adatto, all'interno del quale vengano garantite certe caratteristiche:

- Resistenza della struttura a fronte di eventi artificiali (accidentali o meno) e naturali
- Sicurezza e controllo degli accessi
- Condizionamento dell'aria e gestione della temperatura
- Condizionamento dell'alimentazione elettrica ed utilizzo di sistemi di continuità (istantanea od a lunga durata)
- Connettività di rete

Poichè i costi associati alla costruzione ed al mantenimento di queste strutture sono molto elevati diventa indispensabile condividerle, secondo differenti modalità:

- Housing: il data center fornisce spazio e connettività per garantire il funzionamento dei sistemi. L'acquisto e la gestione dei sistemi è a carico del cliente

- Managed housing: il data center fornisce anche i sistemi (su hardware dedicato al cliente) ed assistenza sistemistica
- Hosting: il data center fornisce uno o più servizi specifici su hardware consiviso fra i clienti. Il modello "cloud" è un caso partricolare di hosting.

Per garantire la disponibilità continua di un sistema si può agire sia sulle caratteristiche costruttive dei componenti, riducendo ad esempo il MTTF ("Mean Time To Failure") ed il MTBF ("Mean Time Between Failures"), oppure agendo direttamente sull'architettura del sistema, eliminando il colli di bottiglia ed i punti di fallimento singoli. Generalmente, poichè il costo dei componenti cresce più velocemente della loro robustezza, conviene far affidamento su soluzioni architetturali, le quali sebbene introducono della complessità, permettono di produrre componenti che ai morsetti sono indistinguibili da quelli tradizionali.

3.1 Disponibilità dei dati

3.1.1 RAID (Redundant Array of Inexpensive Disks)

Rappresenta la tecnologia più diffusa per proteggere un sistema dai guasti dei dischi (ovvero il componente più soggetto a guasti). Poichè spesso un disco cede senza preavviso e il suo costo aumenta più velocemente del MTTF, anzichè usare un unico disco robusto se ne combinano N economici al fine di ottenere un dispositivo, ai morsetti indistinguibile da un disco, più robusto. Sono possibili sia implementazioni hardware, che sgravano la CPU dai calcoli necessari al suo funzionamento, o software.

Esistono più tipologie di RAID, ognuna identificata da un livello:

- RAID 0 (striping): distribuisce i dati su più dischi. Solitamente viene usato per incrementare le prestazioni in quanto è possibile eseguire delle scritture/letture parallele sui dischi, tuttavia la corruzione di un unico disco causa la perdita dei dati dell'intero RAID.
- RAID 1 (mirroring): replica integralmente i dati su tutti i dischi dell'array. Questa configurazione è molto robusta ai guasti, in quanto continua a funzionare fintanto che esiste anche un solo disco integro, e fornisce un incremento delle prestazioni in lettura, in quanto è possibile eseguire letture parallele sui dischi.
- RAID 5 (distributed parity): viene introdotto un blocco di parità il cui posizionamento sui dischi cambia di scrittura in scirttura (il disco da destinare ad ospitare il blocco di parità varia con round robin). Grazie a questo blocco è possibile ricalcolare il contenuto di un blocco

danneggiato. Offre protezione dal fallimento di un unico disco. Quando un disco viene danneggiato l'array funziona in modalità degradata. In questo caso ogni lettura causa la rigenerazione del dato a partire dal blocco di parità, provaocando un aumento delle scritture/letture sui dischi. Se si tratta solo di una disconnesione temporanea del disco si procede a risincronizzare i dischi, alrtimenti, una volta inserito un nuovo disco, esso viene ricostruito completamente a partire dai dati contenuti sugli altri dischi. Questa operazione può durare una decina di ore per dischi di grandi dimensioni e può causare la rottura di altri dischi a causa dello stress fisico.

- RAID 6 (distributed double parity): è molto simile al RAID5 ma con due blocchi di parità, di conseguenza può resistere al fallimento di due dischi contemporaneamente.

Infine è possibile ottenere dei RAID composti ponendo in cascata dei RAID semplici, permettendo così di ottenere dispositivi a blocchi più performanti e/o resistenti. In caso di guasti si può far ricorso a dei dischi "spare": essi non sono altro che dei dischi di scorta che vengono utilizzati nel momento in cui si rileva il fallimento di un disco dell'array andandolo a sotituire. Poichè teoricamente questi dischi non dovrebbero entrare mai in funzione possono essere condivisi tra più RAID adiacenti.

Un'ultima configurazione, che non è RAID tuttavia, da menzionare è il cosiddetto JBOD. Questo non è che una serie di dischi collegati in sequenza a formare un disco risultante avente come dimensione la somma delle dimensioni dei dischi che lo compongono. Non fornisce ridondanza dei dati, in quanto i dischi sono semplicemente concatenati, ma rende relativamente sempice estendere la capacità del dispositivo risultante.

3.1.2 Supporto multi-device

md md è erede dei classici raidtools ed è orientato alla creazioni di sistemi RAID, fornendo supporto base anche per il multipath. È somposto da un driver modulare del kernel, che rende possibile caricare solamente i moduli di cui si ha necessità. Questo driver consente di combinare qualsiasi block device per formare un nuovo md block device, che può essere usato come se fosse un normale disco (eventualmente come base per un ulteriore md device). Su tutti i dischi dell'array è presente un superblock che contiene i metadati realtivi allo stato dell'array e consente la rilevazione automatica del RAID

dm Il device mapper (dm) è un modulo del kernel che, in termini generali, mappa un block device su altri secondo una data politica. Ogni dispositivo è visto dal sistema come un dispositvo a blocchi e a sua volta può essere costituito da altri dispositivi a blocchi. In userspace si definisce la politica

tramite cui mappare i singoli blocchi del mapped device sui blocchi dei target. Il tool di base è dmsetup che dati un target driver e dei target device è in grado di configurare un mapped device affichè le letture/scritture siano effettuate in realtà sui target device in accordo alla politica descritta in una tabella di mapping. Sebbene sia possibile usare dmsetup direttamente, spesso conviene sfruttare altri tool basati su di esso che permettono di gestire meglio alcuni task specifici.

- dm_mutipath:fornisce percorsi multipli per l'accesso allo storage.Il kernel riconosce ogni block device come indipendente e successivamente multipath si occupa di riconoscre quali dispositivi siano in realtà incarnazioni dello stesso disco, fornendo una vista unificata. Permette di configurare le modalità tramite cui testare la salute di un dispositivo, la frequenza con cui ripetere i test, lo smistamento del traffico...
- LVM2: permette di astrarre la gestione dello spazio disponibile fornendo un'interdaccia più ricca di quella di dm. I vantaggi principali di LVM sono l'astrazione e la flessibilità. L'architettura è a livelli, partendo dal basso:
  - i driver permettono di astrarre i dischi fisici in PV (phisical volume)
  - physical volume: rappresentano l'astrazione LVM di dischi fissi o di partizioni (in genere possono essere costituiti da un qualsiasi dispositivo a blocchi, sia esso reale o costruito con tool come dm, md, ...). Sono dei dispositivi a blocchi inizializzati con dei metadati nell'header
  - volume group: rappresenta un grande contenitore costituito dalla somma dei physical extent, che si trovano nei PV. Può essere partizionato logicamente per ottenere dei LV
  - logical volume: estendono il concetto standard di partizione e sono riconosciuti dal sistema come dispositivi a blocchi locali. Possono essere partizionati a loro volta e formattati per contenere un filesystem. Sono costituiti da un insieme di logical extent, i quali sono semplicemente mappati al di sopra dei physical extent contenuti all'inteno dei PV.

  Grazie all'uso di LVM è possibile ridimensionare più facilmente delle partizioni a seconda delle necessità, in quanto i physical extent su cui è mappato un LV non devono necessariamente essere adiacenti. Infine LVM permette la creazione di snapshot, ovvero copie virtuli di un LV che ne fotografano il contenuto in un certo istante. Gli snapshot possono essere usati ad esempio per condividere un'immagine base di un filesystem, andando ad inserire tutte le modifiche in degli snapshot incrementali.

3.1.3 Multipath

Oltre alla ridondanza dei dischi, può essere necessario prevedere tolleranza ai guasti sia dei componenti di controllo che dei percorsi di connessione. Per far ciò si rende necessario o dell'hardware specializzato che permetta di creare più percorsi di accesso allo stesso disco oppure del supporto da parte del SO per gestire correttamente la molteplicità di percorsi. In Linux sono presenti i tool md e dm (oltre a driver proprietari).

Il multipath hardware consente di sdoppiare i percorsi per raggiungere un disco. È necessario l'utilizzo di un controller che permetta il raggiungimento del disco tramite due percorsi indipendenti.

3.1.4 Filesystem distribuiti

Un componente essenziale per la realizzazione di sistemi scalabili ed affidabili è un sistema di storage condiviso. Esistono due grandi approcci:

- NAS: i dati sono su di un sistema che li conserva in un filesystem locale i cui dettagli sono pressochè trasparenti ai client. Questo sistema si occupa di erogare i file tra dal server al client tramite un apposito protocollo. Poichè i dati sono gestiti da un sistema apposito non è necessario introdurre delle forme di collaborazione tra i client

- SAN: i dati sono conservati allo stato grezzo sul sistema, esiste un protocollo in grado di erogare dei blocchi ai client, di conseguenza il file system è deifnito dai client che devono cooperare al fine di gestire gli accessi concorrenti al disco. Il protocollo in sè è più leggero in quanto non è necessario distribuire l'astrazione del filesystem, ma è più complesso per quanto riguarda la gestione della concorrenza

DRBD DRBD è un sistema per lo storage replicato su piattaforme Linux. È implementato come un insieme di driver kernel ed un insieme di applicativi userspace. Propone diverse modalità di comunicazione tra host e di scrittura su disco, permettendo così di trovare un buon compromesso tra latenza ed affidabilità, assieme alla possibilità di integrazioni con sistemi cluster. Infine, poichè il sistema vede un disco DRBD come un normale dispositivo a blocchi, essi possono essere usati per la costruzione di un ulteriore livello di DRBD, oppure per la creazione di volumi logici.

Ogni volta che viene effettuata una scrittura su disco la modifica viene propagata via rete agli altri nodi connessi. A questo scopo esistono più protocolli, ognuno con un diverso grado di affidabilità, differenziati dai requisiti necessari a considerare una scrittura completa sul nodo peer.

1. Protocollo A (asincrono): la scrittura si considera effettuata nel momento in cui i dati vengono inseriti nel buffer TCP locale

2. Protocollo B (semi-sincrono): la scrittura si considera completata se i dati hanno raggiunto il nodo peer

3. Protocollo C (sincrono): la scrittura si considera effettuata se viene confermata la scrittura sul disco del nodo peer

Viene fatta distinzione tra nodi:

- **Primary**: possono utilizzare il dispositivo virtuale
- **Secondary**: sebbene partecipino attivamente al mirroring dei dati tra locale e remoto, non possono utilizzare il dispositivo virtuale

Grazie a questa distinzione si individuano due modalità principali di esecuzione:

- **Single primary**: esiste un solo nodo primary, mentre tutti gli altri sono secondary. Nel caso in cui il nodo primary fallisca uno dei nodi secondary può essere promosso a primary per il tempo necessario. Questa modalità ha il vantaggio di funzionare con qualsiasi filesystem tradizionale
- **Dual primary**: è possibile che coesistano più nodi primary per la stessa risorsa. In questo caso è necessario adottare politiche specifiche per risolvere i problemi derivanti dalla concorrenza, ad esempio mediante un lock manager oppure tramite un cluster filesystem

In ogni caso è sconsigliabile andare a manipolare i dischi usati da DRBD senza la sua mediazione in quanto all'interno di ogni disco è contenuta una sezione di metadati necessari al corretto funzionamento del sistema. Questi metadati consentono inoltre di risicronizzare i dischi in caso di una interruzione temporanea

Configurazione La configurazione di DRBD è gestita tramite il file `/etc/drbd.conf`. Questo file contiene l'elenco delle risorse da utilizzare, quali siano i loro nomi sui vari nodi, e quale sia l'indirizzo ad esse associato. È necessario che il file sia identico su tutti i nodi della rete. Il funzonamento delle singole risorse può essere gestito tramite l'applicativo `drbdadm`, il quale permette attivare/disattivare una risorsa sul nodo locale, di connetterla/disconnetterla temporaneamente ed infine di monitorare lo stato delle risorse.

NFS NFS è un protocollo di rete per la creazione di filesystem distribuiti basato su SunRPC. I server NFS sono senza stato perciò ogni operazione è completamente autocontenuta, permettendo di sviluppare di server più snelli cn migliori capacità di recovery dai guasti. La sceta di basarsi su RPC

consente di espandere con grande flessibilità l'insieme dei demoni incaricati delle varie funzioni e di nascondere i dettagli del protocollo sottostante. Il protocollo non offre alcun meccanismo di sicurezza o di autenticazione se non quelli rudimentali basati sul controllo dindirizzo e porta. È presente un basilare meccanismo per il contenmento degli utenti, tuttavia basato sull'uso di `UID` e `GID`; poichè questi valori non sono generalmente globali, ma relativi ad una macchina specifica, è possibile che un certo utente ne impersoni un altro su di una macchina differente. In particolare questo pone dei grossi problemi di sicurezza per quanto riguarda l'utente `root`, avente sempre `UID = 0`. Di conseguenza è importante mitigare questi problemi specificando delle impostazioni di condivisione adeguate, ad esempio istruendo i demoni a considerare tutti gli utenti `root` diversi, oppure definebndo precisamente chi (utenti, host, indirizzi ip, reti, ...) possa accedere a cosa (quali risorse) e come lettura, scrittura, ...) possa accedervi.

Lato server Tra i vari demoni disponibili lato server i più rilevanti sono `mountd`, che ha l'unico scopo di rilasciare ai client un token che indica a `nfsd` su quale filesystem esso volgia effetuae una certa operazione, e `nfsd` stesso, che si occupa di eseguire le operazioni richieste. Di questo ultimo demone occorre calibrarne bene il numero di fork consentite, infatti se esse sono troppe si rischia di avere un numero spropositato di processi idle che vengono risvegliati dal kernel ad ogni chiamata, se sono troppo poche invece possono nascere congestioni nel traffico di rete.

L'elenco delle directory esportate viene contenuto all'interno del file `/etc/exports`, dove vengono specificate le regole di accesso.

Lato client Una volta che il filesystem è stato esportato dal server occorre montarlo in locale come se fosse un filesystem locale.

Cluster filesystem Se si vuole accedere direttamente ad un block device (SAN) da molti client, serve un meccanismo di arbitraggio distribuito: un cluster filesystem. Un cfs si occupa contemporaneamente sia di organizzare i file sul block device, come un normale fs, sia di gestire l'accesso concorrente dai diversi client ed eventualmente di replicare I dati su nodi diversi.

RedHat GFS GFS è il CFS sviluppato da RedHat. La sua struttura è simile a quella di ext3 (superblock, journaling, ...). L'utilizzo di un dispositivo fisico accessibile via rete (SAN-style) può godere delle caratteristiche di flessibilità garantite da LVM purchè i metadati siano resi noti a tutti i nodi. CLVM fa uso di LVM per l'organizzazione dello spazio du disco e del demone dlvmd per la distribuzione dei metadati di LVM tra i client. Poichè CLVM non effettua locking è necessario disporre di uno strumento che renda disponibile questa funzionalità, ovvero DLM (Distributed Lock Manager).

Esso permette di bloccare qualsiasi risorsa ponendo un lock per il primo nodo che cerca di accedervi. Esso diventa il lock master per quella risorsa e conserva l'insieme delle informazioni associate al lock. Quando un secondo nodo vuole accedere alla meddesima risorsa esso consulta il lock manager per sapere come comportarsi. I lock non sono necessaramente esclusivi, ma possono essere combinati, ad esempio se il lock master impone un lock del tipo "Protected read" gli altri nodi ptranno leggere la risorsa ma non modificarla.

3.2 Disponibilità dei sistemi

Oltre alla disponibilità dello storage occorre assicurarsi che anche tutte le altre componenti sia disponibili ed affidabili. Pertanto si può procedere alla duplicazione di ognuno dei componenti che compongono il sistema. L'architettura risultante può raggiungere un alto grado di complessità, tuttavia porta come vantaggio l'aver a che fare con un unico sistema. Esistono due approcci alternativi nella definizione di risorse logiche:

- virtualizzare il sistema per poterlo riprodurre su di un host diverso semplicemente copiando una manciata di file. Il sistema risultante presenta una buona scalabilità, anche se l'overhead per la virtualizzazione di servizi semplici è elevato. Poichè le VM sono isolate viene incrementta anche la sicurezza, rendendo più solido il server

- realizzare più copie dello stesso sistema, ciascuna con tecnologie standard (cluster HA). Questo soluzione è economica ed efficiente, ma soffre di maggior problemi per quanto riguarda la scalabilità

In ogni caso serve una tecnologia per gestire automaticamente lo spostamento delle risorse logiche sui nodi fisicamente disponibili e per rilevare autoaticamente i guasti dei nodi. A questo proposito si può utilizzare **heartbeat**. Le varie risorse periodicamente si scambiano dei pacchetti di vitalità (heartbeat) per comunicare la loro presenza. In caso una risorsa non emetta più pacchetti può essere considerata come guasta (è consigliabile usare anche altre tecniche per avere una rilevazione più accurata). Infine per distinguere i guasti di una risorsa dai guasti di un collegamento è possibile usare più canali di comunicazione indipendenti, in modo che nel caso in cui uno di essi abbia problemi si possa far affidamento sugli altri.

La scomparsa e ricomparsa di nodi attivi richiede la gestione di due condizioni:

- **Failover**: la situazione in cui un nodo si guasta e le risorse del cluster lo devono sostituire. Presenta alcuni problemi, tra cui decidere quale risorsa sia guasta, gestendo le situazioni di incomunicabilità, ed evitare il conflitto di possesso delle risorse

- **Failback**: la situazione in cui un nodo precedentemente guastatosi ripristina la sua funzionalità. Si presentano due alternative:
  - mantenere le sessioni in corso e ridurre i tempi di commutazione superflui ("nice failback", chi ha le risorse eroga il servizio)
  - sfruttare al meglio la capacità computazionale derivante dai nodi ritornati disponibili ("auto failback", chi è dichiarato "master" per una risorsa la riprende appena torna online)

  Tipicamente, dopo la rilevazione di un guasto di un nodo, un altro ne prende in carico i servizi, eventualmente assumendone anche l'identità in rete. Questo può essere fatto a più livelli:
  - MAC takeover
  - DNS reconfiguration
  - IP takeover. Tipicamente è la scelta più comune, ad ogni nodo viene associato un indirizzo IP reale (fisso) ed al cluster un indirizzo collettivo (floating IP, considerato un alias dell'IP del master). Nel caso di fallimento del master il nodo che ne prende il posto assume il floating IP

Partizionamento Idealmente ogni nodo o è perfettamente funzionante oppure è disconnesso dagli altri senza la possibilità di accedere alle risorse condivise. In realtà un guasto potrebbe pertizionare l'insieme dei nodi in dei sub-cluster separati. All'interno dei sub-cluster i nodi si vedono perfettamente e mantengono un certo grado di operatività. Un modo per garantire che un sub-cluster non acceda a delle risorse condivise è il cosiddetto "fencing":

- resource fencing: ogni nodo può privare gli altri dell'accesso a date risorse (ad esempio spegnendo la porta di uno switch)
- node fencing: spegnere il nodo definitivamente

Si pone però il problema di come comprendere quale sia la reale entità del guasto (guasto del nodo o partizionamento?) e di decidere chi deba effettuare il fencing e di chi lo debba subire. Una soluzione a questi problemi (per i cluster con pù di due nodi) deriva dall'utilizzo del "quorum". Ogni nodo deteniene un voto e nel momento in cui un subcluster raggiunge un numero di voti superiore alla metà può continuare ad operare. Tuttavia questa tecnica è impotente contro i partizionamenti del cluster in un numero arbitrario di sub-cluster, in quanto in questo caso non è possibile decidere a priori quale sia la soglia di voti necessaria per poter continuare ad operare (la decisione diamica prevedrebe la possibilità di comunicare tra i sub-cluster).

Approcci alternativi Con le archtetture cluster non possono essere garantiti i "five nines", anche se il loro rapporto qualità prezzo è ottimo per la maggior parte delle applicazioni. Se si vuole raggiungere una disponibilità maggiore occorre far affidamento su approcci hardware, cosituiti dal mirroing dell'intera architettura di calcolo. In alternativa è possibile utilizzare approcci software, i quali ricorrono alla ridondanza ed alla virtualizzazione delle varie risorse.

Cloud computing Le architetture HA, sebbene molto solide, hanno costi elevati, pertanto per alcune tipologie di progetti (piccoli o con fattori di utilizzo delle risorse lontani dal 100%, medie per cui risulta difficoltoso l'investimento di capitale, a rapida crescita ma senza una certezza riguardo ai tempi di crescita) sono auspicabili soluzioni differenti.

Il cloud provider si fa carico della realizzazione di data center, realizzando gli edifici, predisponendo gli impianti e comprando gli apparati di calcolo necessari. Le risorse disponibili vengono poi utilizzate per far funzionare dei sistemi virtualizzati: i clienti condividono le risorse fisiche, spalmando i costi fissi, ma hanno l'impressione di lavorare su macchine dedicate grazie alla configurazione ed all'uso di interfacce. Infine i cloud permettono ai clienti di usare le risorse *on demand*, si parla di **risorse as a service**.

Alla base di un cloud si trova l'architettura reale, costituita da vari server. Sopra di essi viene costruito uno strato infrastrutturale che abiliti la realizzazione dei servizi cloud per mezzo della virtualizzazione, in questo modo è possibile ottenere risorse di calcolo su richiesta. Sullo strato infrastrutturale viene costruito lo strato di piattaforma, esso fornisce i serivizi standard e i componenti modulari fruibili remotamente dagli strati superiori. In cima viene posizionato lo strato software che mette a disposizione i vari servizi, che dunque necessita solamente di essee configurato. I clienti possono accedere al cloud da remoto ad esempio tramite interfacce web o tramite applicazioni, che rimangono l'unico componente software residente sulle macchine client.

Per poter realizzare un sistema cloud è necessario soddisfare alcuni requisiti:

- grandi pool di calcolatori architetturalmente simili ed intercambiabili su cui far girare un hypervisor
- apparati di rete gestibili e riconfigurabili
- sistemi di sotrage di rete
- interfacce al sistema
- sistemi di monitoraggio (delle risorse e non) che siano dettagliati, facilmente accessibili e fortemente programmabili affincè possano reagire autonomamente a certi eventi.

- modelli di configuration amnagement

3.3 Tecniche per la salvaguardia

Per quanto un sistema possa essere sicuro esso non potrà ignorare comandi errati o sopravvivere ad eventi disastrosi, di conseguenza sono necessari sistemi di backup per presenvare i dati e sistemi di recovery per poterli ripristinare.

Il backup consiste nel copiare i dati di un sistema live su di un supporto offline. Essendo un impegno complesso va organizzato nei minimi dettagli in modo da decidere quando sia necessario effettuarlo, con che frequenza, chi siano gli incaricati, ... Per la sua esecuzioni sono possibili più strategie:

- full backup: consiste nella copia integrale di un sistema soggetto a backup. È una procedura molto lenta che produce un molti dati, ma semplifica enormemente le operazioni di ripristino
- backup incrementale: consiste nella copia dei soli file modificati rispetto ad una certa data. È più adatto ad una esecuzione frequente, ma rende più difficoltoso il ripristino del sistema, in quanto va prima ripristinato il backup completo e poi rieffettuate le modifice incrementali.

Poichè dai backup dipende il ripristno di un sistema a seguito di un guasto occorre porre particolare attenzione alla cnservazione del backup stesso.

- Correttezza della copia: idealmente il filesystem dovrebbe essere a riposo durante il backup, ma è raro nella pratica, quindi bisogna curare bene i dettagli relativi alla lettura di file aperti o di strutture complesse come i database
- Protezione dei dati: un backup contiene tutti i file del sistema, quindi in caso di requisiti di riservatezza va difeso allo stesso modo
- Integrità dei dati: se il backup viene svolto senza supervisione del sysadm, ci si deve cautelare da attività anche involontarie degli utenti che possano provocare la sovrascrittura dei dati
- Affidabilità dei supporti: con periodicità dipendente dalla criticità dei sistemi, ci si deve accertare che i dati siano scritti correttamente e siano leggibili per tutta la durata prevista della copia, curando sia i fattori tecnologici (graffi, smagnetizzazione, obsolescenza hw e sw...) sia i fattori ambientali (polvere, umidità, temperatura, ...)
- Facilità di reperimento: i supporti devono essere organizzati per consentire di individuare facilmente ciò che si deve ripristinare

- Conservazione: occorre scegliere un luogo sicuro, diverso da quello in cui è situato il sistema principale, inoltre occorre premurarsi che siano conservate più copie del medesimo backup

Capitolo 4

Integrazione sistemistica

4.1 SNMP

SNMP (Simple Network Managment Protocol) è un protocollo di livello applicativo (UDP porta 161 gli agenti, porta 162 per i manager) creato per ottenere un modo uniforme e standard per ottenere informazioni relative ad apparati spesso molto differenti tra loro. In generale una rete attrzzata per l'utilizzo di SNMP consiste principalmente di dispositivi detti *agenti*, ovvero di programmi in grado di trovare informazioni relative ad un dato componente hardware o software ed i organizzarle secondo uno schema predefinito interrogabile da entità dette *manager*. I manager tipicamente sono più semplici degli agenti in quanto devono solamente essere predisposti per effettuare delle richieste. Il modello di interazione è tipicamente request/response dove gli agenti rispondono alle richieste dei manager, tuttavia è possibile avere anche delle interazioni push asincrone con cui gli agenti possono contattare i manager.

All'interno di SNMP le informazioni sono organizzate all'interno di una struttura gerarchica ad albero detto MIB ("Managed Information Base"), popolato dai vari agenti e letto dai manager. Per riferirsi ad un nodo specifico dell'albero occorre conoscerne l'OID, ovvero conoscerne l'intero percorso dalla radice dove ogni giunzione viene sintatticamente specificata tramite un ".". Esistono tre variamnti sintattiche degli OID:

- Un OID rappresenta in astratto un nodo dell'albero
- Se la proprietà è uno scalare per ottenerne il valore occorre aggiungere uno "0" finale all'OID che rappresenta il nodo
- Se la proprietà è una tabella per accedere la valore di una cella è sufficiente aggiungere ".riga.colonna" alla fine

All'interno dell'albero esistono alcuni MIB notevoli tra cui:

- MIB-2: collocato sotto 1.3.6.1.2.1, include le informazioni essenziali per la gestione degli apparati di rete. Per migliorarne l'estendibilità è suddiviso in sottomoduli.
- PEN (private enterpise number): collocato sotto 1.3.6.1.4.1, è dedicato a moduli specifici richiesti da enti privati (ovvero non ISO). Di particolare interesse all'interno di questo sottoalbero si trovano
  - UCD-SNMP che permette l'accesso ai parametri base sel sistema operativo
  - NET-SNMP-EXTEND-MIB che permette di consultare l'output delle direttive extend, trasformandolo in un managed object

4.2 LDAP

LDAP (*Lightweight Directory Access Protocol*) è un protocollo che permette di accedere ad un servizio di directory il cui scopo principale è quello di mantenere ed organizzare le informazioni e on la loro gestione. L'utilità di questi protocolli deriva dalla necessità di scegliere la sorgente per alcuni dati (ad esempio NSS e PAM), magari integrando sistemi differenti, e di distribuirli efficientemente ai client (ad esempio per distribuire all'interno di una rete le informazioni circa gli utenti). I servizi di directory sono basati sullo standard ISO X.500, tuttavia LDAP introduce una serie di semplificazioni atte a ridurre la complessità del protocollo.

Il protocollo (TCP porta 389) è strutturato sul modello classico client-server, dove i client effettuano l eircerche, mentre i server mantengono le informazioni; inoltre in caso un server non abbia le informazioni richieste, in maniera simile ai DNS, può richiederle ad un altro server adottando una gerarchia ad albero. È costituito da uno schema per l'assegnazione dei nomi (il naming model) che identifica il sigolo dato all'interno del sistema, una parte per l'organizzazione dei dati (il data model) che permette di stabilire come venga rappresentata l'informazione ed una modalità per accesso ai dati. Il protocollo non definisce come queste operazioni debbano essere implementate, ma lascia i singoli server liberi di scegliere l'implementazione di preferenza, tuttavia definisce quale sia il funzionamento logico delle operazioni (ricerca, comparazione, aggiunta, rimozione, modifica, ...) e quale sia lo schema astratto (LDIF) per trattare i dati.

Poichè all'interno di LDAP le informazioni sono per lo più lette non si rendono necessari complessi meccanismi di roll-back o di sincronizzazione tra i vari servitori replicati, inoltre l'organizzazione delle informazioni è di tipo descrittivo non relazionale.

Modello dei dati La base di dati della directory può essere un qualsiasi sistema di archiviazione purchè esista un modulo che espone un'interfaccia

standard (detta *DBI*) in grado di pilotare il backend. DBI gestisce una serie di entry costituite a loro volta da un elenco di attributi. Ogni attributo ha un tipo specifico e può comparire più volte all'interno della stessa entry. Ogni entry inoltre possiede un *Distinguished Name* che permette di identificare univocamente la enry all'interno dell'albero detto *DIT*, che rappresenta come la base di dati venga esposta al mondo. Il DIT ha origine in una entry speciale detta *base* o *suffix*.

La scelta di usare una gerarchia per organizzare le informazioni consente da un lato un facile partizionamento ai fini dell'amministrazione, del controllo degli accessi e della collocazione fisica, dall'altro il facile reperimento dei riferimenti ai dati. Tuttavia vincola gli utilizzatori ad una determinata vista dell'organizzazione.

Replicazione LDAP prevede nativamente meccanismi per l'alta disponibilità aiutato dalla sua stessa struttura, che prevede rare scritture. Esistono due modelli base:

- modello multimaster (modello corrente): tutti i server accettano scritture, si fa uso di protocolli ad-hoc per aggiornare reciprocamente i vari DBI
- master-replice (modello storico ma diffuso): questo modello prevede la presenza di un server master che accetta le scritture e di uno o più server replica che accettano le letture. Periodicamente viene effettuata una copia del DBI del master sui server replica al fine di aggiornarne i dati. Di questo modello ne esistono tre varianti:
  - On master: il master accetta sia le scritture che le letture, mentre i replica solamente le letture. Periodicamente il DBI dei serve replica viene aggiornato dal master
  - By referral: in caso il client chieda una scrittura ad un server replica ricevera come risposta l'indirizzo del master su cui effettuarla. Il master poi autonomamente si occupa di propagare la modifica richiesta
  - By chain: in caso il client chieda una scrittura ad un server replica questo fingerà di poterla effettuare, ma in realtà ricorsivamente ripeterà la richiesta al master che ritornerà il risultato ed aggiornerà il DIT dei server replica

È inoltre possibile suddividere un DIT in più partizioni e collocarle su server differenti. Questo viene fatto per ottenere prestazioni migliori (si deve cercare all'interno di un albero più piccolo), per distinguere i dati su base geografica o amministrativa. Per realizzare questo comportamento sono sufficienti due link che colleghino il sottoalbero con l'albero principale: un *Subordinate knowledge link* che permette al server principale di connettere logicamente

il proprio albero con il sottoalbero contenuto nel server secondario ed un *Superior knowledge link* che contiene l'URI del server principale.

Formato delle entry Una entry è una collezione di attributi, che a differenza di un oggetto di un linguaggio di programmazione non è definito da una variabile di un dato tipo, ma direttamente dal tipo e da un valore. Tra gli attributi ce ne sono due che sono sempre presenti: `dn`, che specifica il DN della entry, e `objectClass`, che specifica quale sia/siano la classe/i a cui la entry appartiene. Una classe vincola quali attributi una entry debba avere obbligatoriamente o quali possa avere facoltativamente. Tra gli attributi è necessario sceglierne uno che faccia da "Relative Distinguished Name" (unico tra le entry aventi lo stesso BDN), che, associato al "Base Distinguished Name" (che identifica la entry a cui si sta attaccando la entry attuale), dà come risultato il DN della entry.