



**Università
degli Studi
di Ferrara**

UNIVERSITÀ DEGLI STUDI DI FERRARA
DIPARTIMENTO DI INGEGNERIA

**Corso di Laurea in
Ingegneria Elettronica e Informatica**

**Monitoraggio di una infrastruttura IT mediante
tecnologie Open Source: studio e simulazione**

RELATORE:
Prof. Ing. **Mauro Tortonesi**

LAUREANDO:
Samuele Padula

CORRELATORE:
Ing. **Giulio Riberto**

ANNO ACCADEMICO 2019/2020

Indice

Elenco delle figure	ii
Introduzione	iii
1 Infrastruttura IT	1
1.1 Security Monitoring	1
1.2 Architettura di rete da monitorare	3
1.3 Virtualizzazione: VirtualBox	3
1.3.1 Impostazioni di rete	4
1.4 Firewall: pfSense	6
1.4.1 Configurazione di rete	7
1.5 Sistemi di monitoraggio	8
2 Zabbix: Sistema di monitoraggio	9
2.1 Introduzione	9
2.2 Architettura di Zabbix	10
2.3 Configurazione dei componenti principali	11
2.3.1 Host	11
2.3.2 Item	12
2.3.3 Trigger	14
2.3.4 Notification	16
2.3.5 Information flow in Zabbix	18
2.4 Zabbix Proxy	19
2.4.1 Data flow	20
2.5 Zabbix Agent	21
2.5.1 Active vs Passive	22
3 Configurazione del sistema di monitoring	24
3.1 Zabbix Server	24
3.1.1 Inizializzazione	25
3.1.2 Configurazione del database	26
3.1.3 Installazione e configurazione del server	26
3.2 Zabbix Proxy	27
3.2.1 Inizializzazione	27

3.2.2	Installazione e configurazione del proxy	28
3.2.3	Creazione del proxy dal front-end web	29
3.3	Zabbix Agent	29
3.3.1	Inizializzazione del server da monitorare	29
3.3.2	Installazione e configurazione dell'agent	30
3.3.3	Creazione dell'host dal front-end	31
3.4	Client-side	32
4	Simulazione di una procedura di difesa in risposta ad un attacco informatico	33
4.1	Configurazione di Ubuntu Server	33
4.1.1	Creazione dell'item	33
4.1.2	Creazione del trigger	34
4.1.3	Creazione dell'action	34
4.1.4	Gestione dei permessi	35
4.2	Configurazione di pfSense	35
4.2.1	Regole di pfSense	35
4.2.2	Scripts PHP	37
4.3	Sviluppo del Bot Telegram	39
4.3.1	Python	40
4.3.2	python-telegram-bot	40
4.3.3	paramiko	40
4.3.4	Bot Telegram	41
5	Conclusioni	44
	Bibliografia	45

Elenco delle figure

1.1	Mappa dell'infrastruttura IT monitorata	4
1.2	Configurazione schede di rete pfSense	7
2.1	Esempio di architettura Zabbix	10
2.2	Information flow in Zabbix	19
2.3	Flusso dei dati dello Zabbix Proxy attivo	20
2.4	Flusso dei dati dello Zabbix Proxy passivo	21
2.5	Agent passivo	22
2.6	Agent attivo (1)	23
2.7	Agent attivo (2)	23
3.1	Creazione dell'host Ubuntu Server dalla interfaccia web di Zabbix	31
4.1	Regole della WAN	36
4.2	Regole della LAN_57	37
4.3	Regole della LAN_58	37
4.4	Funzionalità del Bot Telegram	41
4.5	Selezione dell'interfaccia	42
4.6	Selezione dell'azione	42

Introduzione

Fino a qualche decade fa, le informazioni riguardanti lo stato della rete e le performance di un'intera infrastruttura erano raccolte e processate in un singolo "luogo", sia esso fisico o virtuale. L'arduo compito di analizzare questa enorme mole di dati era affidato all'amministratore di sistema, con evidenti risvolti negativi soprattutto nel campo della sicurezza informatica. Per poter avere una panoramica completa del sistema era, ed è necessario, poter monitorare qualsiasi suo aspetto istante per istante. Infatti a fronte di una crescita esponenziale del carico di dati trasmessi sulle reti di calcolatori, è fondamentale garantire il continuo funzionamento di un numero enorme di servizi anche in condizioni di stress o anomalia. In questo scenario, è intuibile come il *monitoring* assuma un ruolo cruciale per le reti odierne. Con il termine *monitoring* si vuole identificare una vasta area composta di software e hardware, il cui principale compito è quello di evidenziare le situazioni anomale che si presentano in un determinato sistema.

In ogni sistema di monitoraggio, lo scopo primario è quello di rilevare un guasto, cercando di prevenire un possibile fault. L'intervento proattivo in caso di problemi permette di migliorare stabilità e longevità di un componente. In particolare è interessante notare come riuscire a rilevare situazioni pericolose prima di un guasto, consenta di incrementare le prestazioni, perfezionare la sicurezza, affinare la gestione del sistema e innalzare l'efficienza.

Le **prestazioni** sono uno dei tanti Sacri Graal dell'informatica. I sistemi non sono mai abbastanza veloci per soddisfare le crescenti esigenze, bisogna quindi bilanciare le operazioni desiderate con le risorse disponibili. La **sicurezza** è un altro processo senza fine in cui ci si aspetta di utilizzare molti strumenti e tecnologie per prevenire e difendere. La **gestione del sistema** implica seguire un certo insieme di regole in un preciso ordine e citando Patrik Uytterhoeven: "*Un buon amministratore di sistema non fallisce mai le operazioni di management tranne quando le mette in at-*

to". Ultima ma non meno importante, l'**efficienza** che potrebbe essere considerata il primo passo per una migliore disponibilità e più alte prestazioni, il che aumenta l'importanza di conoscere quanto i propri sistemi siano efficienti.

Le reti di calcolatori odierne stanno assumendo dimensioni sempre più estese ed eterogenee a causa del repentino aumento di dispositivi ad esse connessi. La varietà di servizi offerti sulla rete e di device volti a garantire la confidenzialità, integrità e disponibilità dei dati erogati (firewall, NAC, IDS/IPS, antivirus), ha incrementato i tempi necessari per il rilevamento delle problematiche di un infrastruttura IT. Considerando anche i metodi attuati per la rilevazione e l'analisi dei problemi, spesso privi di uniformità, si è spinto il mercato a realizzare pacchetti software e framework in grado di ridurre lo spreco di risorse umane e rendere il monitoring un *arte sistematica*.

L'obiettivo di questa tesi è l'utilizzo e lo studio di un software open-source che dia la possibilità di controllare e visionare lo stato dei vari nodi all'interno della rete, dialogando in modo veloce ed efficace con i dispositivi di sicurezza preesistenti. Il lavoro di tesi è strutturato in quattro capitoli. Nel capitolo 1 viene descritta la composizione dell'infrastruttura IT indagata e gli strumenti di simulazione utilizzati. Nel capitolo 2 viene illustrato il principio di funzionamento del software Zabbix. Nel capitolo 3 si mostra la messa a punto del sistema di monitoraggio. Infine nel capitolo 4 viene simulata una procedura di difesa in risposta ad un cyber-attacco con l'ausilio di Zabbix e di un bot Telegram sviluppato in Python.

Capitolo 1

Infrastruttura IT

1.1 Security Monitoring

La sicurezza informatica, o Cyber Security, è un problema rilevante alla luce soprattutto del crescente numero di attività che si svolgono attraverso strumentazioni informatiche. Ed è proprio l'elevato numero di scenari possibili in ambito IT (Information Technology) e OT (Operational Technology) che richiede un continuo e approfondito monitoring delle reti, dei dispositivi e delle soluzioni adottate. Ecco allora coniato il termine Security Monitoring, il monitoraggio della sicurezza, il quale implica la raccolta e l'analisi di informazioni per rilevare comportamenti sospetti o modifiche di sistema non autorizzate all'interno della propria infrastruttura, definendo quali tipi di comportamento dovrebbero generare avvisi e attivare delle contromisure. Il monitoraggio della sicurezza è quindi il processo automatizzato di raccolta e analisi di indicatori di potenziali minacce alla sicurezza, al fine di valutarle e intraprendere l'azione appropriata. Data la natura onnipresente e inevitabile dei rischi alla sicurezza, per mantenere protetto il sistema sono indispensabili tempi di risposta rapidi ed è quindi fondamentale un monitoraggio della sicurezza continuo e automatizzato per un rapido rilevamento delle minacce e l'adozione di contromisure. È in un contesto simile che nasce il concetto di **SIEM Security Information and Event Management** che definisce un prodotto costituito da software e/o servizi che unisce capacità di **SIM Security Information Management** a quelle di **SEM Security Event Management**. Il SIM è un sistema di gestione delle informazioni che automatizza il processo di raccolta e orchestrazione dei log (ma non in tempo reale). I dati vengono raccolti e spediti a un server centralizzato tramite l'utilizzo di software agent installati sui vari dispositivi del sistema monitorato. La possibilità di usufruire di spazi di archiviazione a lungo termine unita all'analisi dei dati con-

sente la generazione di report personalizzati. Il SEM è una soluzione software che, in tempo reale, provvede al monitoraggio e alla gestione degli eventi che accadono all'interno della rete e sui vari sistemi di sicurezza, fornendo una correlazione e aggregazione tra essi. L'interfaccia è una console centralizzata, preposta ad attività di monitoraggio, segnalazione e creazione di risposte automatiche a determinati eventi.

È dunque necessario il monitoraggio della sicurezza al fine di prevenire molteplici potenziali problemi come vulnerabilità del software, configurazioni deboli, porte aperte non necessarie o intrusioni fisiche. I sistemi di monitoring, dedicati interamente alla sicurezza o meno, consentono il monitoraggio di reti e sistemi informatici grazie a funzionalità integrate come per esempio: controllo di file di configurazione e log, tramite trap SNMP, verifica dei certificati SSL o di servizi esposti erroneamente sulla rete. È infatti possibile usare il calcolo del **checksum** per rilevare modifiche ad eventuali file importanti o effettuare il monitoring del contenuto dei **file di configurazione** estraendo i parametri critici per il sistema (credenziali, hostname, etc...). Inoltre molti sistemi supportano l'integrazione con diversi tool, script e programmi per analizzare i dati, lasciando l'onere di verificare le vulnerabilità a programmi esterni ed analizzando i risultati estraendo informazioni utili sulla base delle quali inviare avvisi e notifiche. Infine è possibile analizzare i file di log di sistema o degli applicativi esistenti alla ricerca di informazioni relative alla sicurezza come accessi non riusciti, accessi riusciti (ad es. per utenti con privilegi elevati) o attuazione di tecniche di privilege escalation. Nella prossima sezione, verrà introdotta l'infrastruttura di rete monitorata, per poter simulare e testare le tecnologie open-source, oggetto di questo lavoro di tesi.

Citando Bruce Schneier:

La sicurezza è un processo, non un prodotto. I prodotti forniscono una certa protezione, ma l'unico modo per rendere sicuro il proprio business in un mondo insicuro è mettere in atto processi che riconoscano l'insicurezza intrinseca dei prodotti. Il trucco è ridurre il rischio di esposizione indipendentemente dai prodotti o dalle patch applicate.

1.2 Architettura di rete da monitorare

L'ambiente utilizzato per simulare l'infrastruttura e il monitoring è stato realizzato utilizzando varie macchine virtuali create tramite Oracle VM VirtualBox con sistema operativo Ubuntu 18.04.5 LTS a 64 bit ed è riportato in figura 1.1. I server presenti sono tre: Zabbix Server e Zabbix Proxy operanti su sistema operativo CentOS 8.2.2004 e Ubuntu Server 20.04. La connettività tra reti interne (netA e netB), LAN e Internet è stata mediata dal firewall pfSense Community Edition 2.4.5, tramite opportune regole definite che verranno illustrate in seguito. Infine, i client presenti nelle reti interne e dai quali è stato possibile accedere alla interfaccia web di pfSense sono due Linux Mint 19.1 64 bit, uno per ciascuna Internal Network. Per ciascuna VM sono state configurate ad hoc le opportune schede di rete virtuali per la trasmissione in LAN, per lo scambio dei messaggi tra le varie VM e per potersi collegare ad Internet esclusivamente per apportare gli aggiornamenti del sistema operativo e del software delle VM.

Le sezioni successive, forniscono una panoramica delle tecnologie e degli strumenti utilizzati per realizzare le soluzioni studiate.

1.3 Virtualizzazione: VirtualBox

Per virtualizzazione si intende l'astrazione di una risorsa fisica attraverso un'interfaccia logica che ne nasconda i dettagli implementativi. Nel caso specifico della virtualizzazione di sistemi informatici, consiste nella predisposizione e utilizzo di intere macchine virtuali, differenti per sistema operativo e applicazioni installate, sul medesimo sistema hardware. La predisposizione di un tale sistema viene realizzata mediante utilizzo di uno strato software intermedio detto VMM (Virtual Machine Monitor) che ha l'effetto di controllare e regolamentare l'accesso delle macchine virtuali alle risorse fisiche della macchina.

Oracle VM VirtualBox è un software gratuito e open-source per l'esecuzione di macchine virtuali (*guest*) all'interno di una macchina fisica (*host*). Ovvero un programma che emula il comportamento di una macchina fisica, per architetture x86 e 64 bit, che supporta sistemi operativi host come Windows, GNU/Linux e macOS e un gran numero di sistemi operativi guests. La versione utilizzata durante il tirocinio è la 6.0.14.

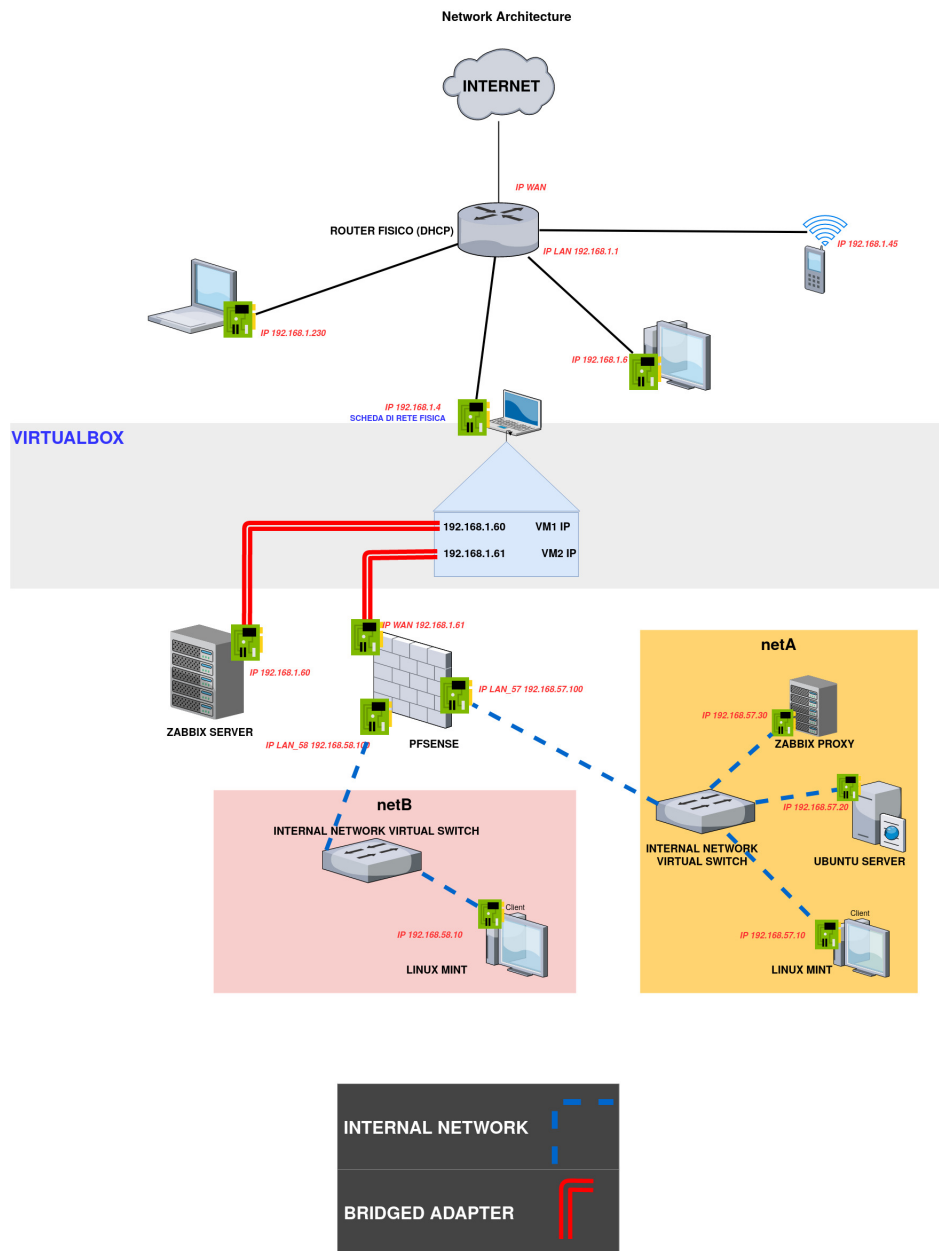


Figura 1.1: Mappa dell'infrastruttura IT monitorata

1.3.1 Impostazioni di rete

All'interno delle impostazioni di rete di VirtualBox sono possibili diverse soluzioni. Di seguito vengono descritte le caratteristiche delle principali interfacce:

- **NAT**: questa modalità di rete è abilitata come impostazione predefinita. Le virtual machine guest non sono accessibili dall'host o da altre macchine nella

rete di quest'ultimo e risultano essere isolate da tutte le altre VM.

L'indirizzo IP dell'adattatore di rete della VM viene ottenuto tramite DHCP. VirtualBox ha un server DHCP integrato e un engine NAT virtuale che utilizza l'adattatore di rete fisico dell'host come interfaccia di rete verso l'esterno. L'indirizzo predefinito del server DHCP virtuale utilizzato nella modalità NAT è 10.0.2.2 (questo è anche l'indirizzo IP del gateway predefinito per una macchina virtuale). La subnet mask è 255.255.255.0.

- **NAT Network:** tale modalità è simile alla modalità NAT utilizzata per configurare un router domestico o aziendale. Se si utilizza tale modalità per più macchine virtuali, queste possono comunicare tra loro. Le VM possono accedere ad altri host nella rete fisica e possono accedere a reti esterne, incluso Internet. Qualsiasi macchina appartenente a una rete esterna o alla rete fisica a cui è connessa la macchina host, non può accedere alle VM configurate per utilizzare questa modalità. Non è possibile accedere alla VM guest dalla macchina host, a meno che non si stia configurando il port forwarding nelle impostazioni di rete globali di VirtualBox;
- **Bridged Adapter:** in questo caso viene utilizzata per connettere l'adattatore di rete virtuale di una VM guest a una rete fisica a cui è connesso un adattatore di rete fisico della macchina host. In altre parole, i pacchetti di rete vengono inviati e ricevuti direttamente da/alla scheda di rete virtuale senza un instradamento aggiuntivo. Uno speciale driver con funzionalità di filtraggio della rete viene utilizzato da VirtualBox al fine di filtrare i dati dalla scheda di rete fisica dell'host.

Quando si utilizza tale modalità di rete, è possibile accedere all'host, agli host presenti sulla rete fisica e alle reti esterne, inclusa Internet. È possibile accedere alla VM dalla macchina host e da altri host (e VM) connessi alla rete fisica.

Questa modalità di rete può essere utilizzata per eseguire server su VM che devono essere completamente accessibili da una rete locale fisica (ad esempio accedere alla interfaccia web di Zabbix dall'host Ubuntu 18.04);

- **Internal Network:** le macchine virtuali i cui adattatori sono configurati per funzionare in tale modalità sono connesse a una rete virtuale isolata. Le VM connesse a questa rete possono comunicare tra di loro, ma non possono comu-
-

nicare con l'host VirtualBox o con altri host in una rete fisica o in reti esterne come Internet.

1.4 Firewall: pfSense

Un firewall è un componente di difesa perimetrale di una rete informatica, che monitora e controlla il traffico di rete in entrata e in uscita sulla base di regole di sicurezza predeterminate. Tipicamente stabilisce una "barriera" tra una rete interna affidabile e una rete esterna non affidabile (per esempio Internet) e sono spesso classificati come firewall di rete o host-firewall. Un firewall filtra il traffico sulla base di un insieme di regole, solitamente dette policy; esistono 2 criteri generali per l'applicazione delle regole:

- Criterio **default-deny o whitelist**: viene permesso solo ciò che viene dichiarato esplicitamente, il resto vietato (approccio classico e più sicuro);
- Criterio **default-allow o blacklist**: viene vietato solo ciò che è esplicitamente proibito, il resto viene permesso.

Un firewall può intraprendere una delle seguenti azioni con i pacchetti analizzati:

- **allow**: il firewall lascia passare il pacchetto;
- **deny**: il firewall blocca il pacchetto e lo rimanda al mittente;
- **drop**: il firewall blocca il pacchetto e lo scarta, senza inviare segnalazioni al mittente.

Ad oggi, sul mercato, sono presenti diverse soluzioni di firewall open-source, gratuite o proprietarie. Si è scelto di adottare *pfSense* in quanto, necessitando di requisiti hardware molto bassi ed essendo esente da costi di licenza, è risultato essere uno dei firewall con il miglior rapporto qualità/prezzo. Inoltre è uno dei migliori firewall open-source ed uno di quelli più utilizzati a livello mondiale al giorno d'oggi, ideale per le esigenze di espansione dei servizi e delle piccole e medie imprese. L'utilizzo di un sistema basato su pacchetti consente all'installazione di base di pfSense di occupare poco spazio, ma allo stesso tempo offre agli utenti la possibilità di installare solo i pacchetti necessari per il proprio ambiente.

pfSense è un firewall/router software open-source; fornisce un firewall completo, versatile, e completamente configurabile utilizzando l'hardware di un comune computer compatibile con la distribuzione. Basato su FreeBSD e il firewall Packet Filter derivato da OpenBSD, offre una potente piattaforma firewall e router ed include una lunga lista di pacchetti che permettono di espandere facilmente le funzionalità senza compromettere la sicurezza del sistema. L'intero sistema è gestibile ed aggiornabile attraverso un'interfaccia web, rendendo il sistema accessibile anche a chi non ha alcuna conoscenza del sistema FreeBSD. Il progetto nasce nel 2004 come fork di *m0n0wall*, un altro progetto di router/firewall nato per installazioni embedded; la differenza sostanziale con *m0n0wall* è la maggiore flessibilità hardware e le maggiori funzionalità disponibili. Alcune funzionalità offerte da pfSense:

- Firewall e Router;
- Wireless access point;
- Proxy e Web Content Filtering;
- DHCP, DNS e VPN server;
- Reporting e Monitoring;
- IPS/IDS.

1.4.1 Configurazione di rete

All'interno di VirtualBox, le schede di rete di pfSense sono state configurate come in figura 1.2. Infatti, dovendo offrire una connessione ad Internet (WAN), si è scelto di adottare la modalità *Bridged*, come illustrato nella sezione 1.3.1, con l'interfaccia di rete `eno1` dell'host. Le interfacce LAN e OPT1 di pfSense sono state dedicate per le *Internal Network* `netA` e `netB`.



Figura 1.2: Configurazione schede di rete pfSense

Si è scelta tale configurazione e non una maggiormente flessibile come quella che avrebbe fatto uso delle VLAN (Virtual Local Address Network) in quanto il virtual

switch presente all'interno dell'engine di VirtualBox è di tipo "unmanaged", dunque privo di possibilità di configurazioni ma capace esclusivamente di replicare i dati sulle interfacce di rete destinatarie.

L'interfaccia `emo1` associata alla WAN è stata configurato con un indirizzo IP statico **192.168.1.61** all'interno della rete fisica così da essere facilmente raggiungibile dal Bot Telegram presentato successivamente. È stato necessario creare un gateway per consentire a pfSense di accedere ad Internet.

Le interfacce di rete locale (`em1` e `em2`) sono state configurate assegnando rispettivamente gli indirizzi IP statici **192.168.57.100** (`LAN_57`) e **192.168.58.100** (`LAN_58`).

1.5 Sistemi di monitoraggio

Esistono molteplici strumenti di monitoraggio sul mercato, alcuni dei quali proprietari, il che significa che è necessario acquistare una licenza per poter utilizzare il prodotto (come SolarWinds, PRTG Network Performance Monitor, ManageEngine). Una valida alternativa sono le piattaforme open source, come Nagios, Checkmk Raw Edition e Zabbix.

Ci sono diverse ragioni per cui scegliere Zabbix rispetto ad altre soluzioni anche e soprattutto in ambienti enterprise. Zabbix offre la libertà di usare una soluzione open-source, evitando qualsiasi vendor lock-in. Questo include non solo il codice sorgente di Zabbix, liberamente accessibile, ma anche i componenti di base su cui esso si poggia (Linux, Apache, MySQL/PostgreSQL, PHP). Il setup e la configurazione di Zabbix sono semplici e ben strutturati, assicurando nel tempo un ridotto costo totale di proprietà. L'agent nativo di Zabbix, disponibile per la gran parte di versioni Windows, Unix e Linux consente un monitoraggio molto ampio e approfondito. Tutte le informazioni (sia le configurazioni, sia i dati raccolti dalla rete) sono memorizzate in un database relazionale per una elaborazione semplice e accessibile.

Capitolo 2

Zabbix: Sistema di monitoraggio

2.1 Introduzione

Il monitoraggio della rete fornisce le informazioni necessarie agli amministratori di rete, per determinare in tempo reale, se una rete funziona in modo ottimale. Grazie all'ausilio di software di monitoraggio della rete, gli amministratori possono identificare in modo proattivo le carenze, ottimizzare l'efficienza e mantenere uno storico degli eventi. Allo stato attuale esistono diverse soluzioni software open-source e non in grado di fornire un feedback sul funzionamento di un infrastruttura IT.

Zabbix è un software di monitoring open-source e distribuito, in grado di raccogliere dati da qualsiasi tipo di dispositivo (server, network devices, virtual machine, web-sites) e verificare la salute e integrità di server e client. Grazie alla possibilità di poter configurare alert e messaggi di notifica tramite una vasta gamma di media types, è possibile anche pre-configurare una rapida risposta ai problemi rilevati mediante azioni automatiche.

Le informazioni raccolte vengono memorizzate all'interno di un database relazionale con la possibilità di poterle visionare sinteticamente tramite opportune funzioni di visualizzazione come mappe e grafici custom (aggiornate in real-time) in grado di combinare item e valori multipli all'interno di una singola view .

È basato su un sistema di monitoraggio centralizzato ma consente di scalare fino a reti grandi e complesse, monitorabili da remoto grazie alle funzionalità di Distributed Monitoring offerte dai componenti Zabbix Proxy. Supporta il monitoraggio tramite funzionalità sia di polling che di trapping, offre una serie di agent software

disponibili per la maggior parte di sistemi operativi client o server based e consente anche di basarsi esclusivamente su metodi agent-less (SSH, IPMI, SNMP) per riuscire a coprire il maggior numero di dispositivi possibili.

2.2 Architettura di Zabbix

Zabbix offre differenti modalità per poter monitorare i diversi aspetti di una infrastruttura IT. Può essere definito come un sistema di monitoraggio semi-distribuito con una gestione centralizzata. Molte installazioni hanno un unico sistema centrale, ma è possibile utilizzare il distributed monitoring tramite proxy e Zabbix agents. Il diagramma seguente mostra una configurazione Zabbix relativamente semplice con molte delle funzionalità di monitoraggio utilizzate e diverse categorie di dispositivi collegati:

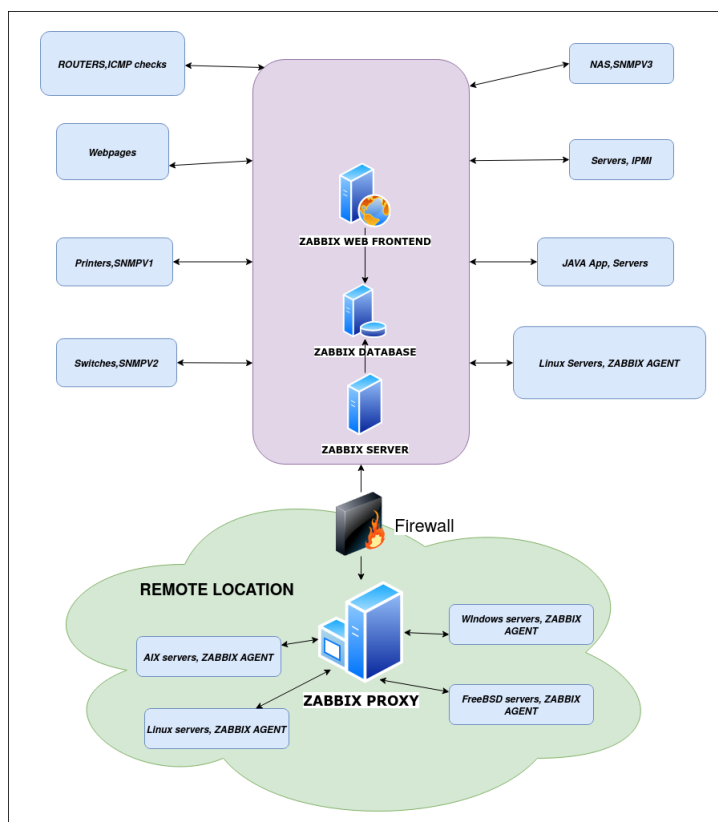


Figura 2.1: Esempio di architettura Zabbix

Il server Zabbix monitora direttamente più dispositivi, ma una posizione remota (per esempio posta alle spalle di un firewall) è più facilmente controllabile tramite

uno Zabbix Proxy. Il componente centrale è il database Zabbix, che supporta diversi backend. Il server Zabbix e il frontend web Zabbix, possono risiedere entrambi sulla stessa macchina o in server separati. Durante l'esecuzione di ciascun componente su una macchina separata, sia il server Zabbix che il frontend web Zabbix necessitano di avere accesso al database di Zabbix, e il frontend web Zabbix deve accedere anche al server Zabbix per visualizzare lo stato del server e per poter avere accesso ad altre funzionalità. Sebbene sia perfettamente corretto eseguire tutti e tre i componenti del server su una singola macchina, potrebbero esserci dei buoni motivi per separarli, come per esempio sfruttare un database o un server web ad alte prestazioni esistente. In generale, i dispositivi monitorati hanno uno scarso controllo sui dati monitorati: la gran parte della configurazione è infatti centralizzata. Questo approccio riduce drasticamente la capacità, di un singolo sistema mal configurato, di danneggiare l'intera configurazione di monitoring. Nel diagramma seguente, abbiamo una panoramica della configurazione di base di Zabbix con il server Zabbix, il server web e il database relazionale. Durante gli studi, i tre componenti sono stati installati su un'unica macchina. Per una più dettagliata illustrazione della loro configurazione, si rimanda alle sezioni presenti nel capitolo 3.

Questo tipo di configurazione può essere estesa aggiungendo molteplici proxy ottenendo una configurazione "*proxy-based*". La configurazione basata su proxy è implementata con un server Zabbix e diversi proxy: un proxy per stabilimento, data center o per ogni segmento remoto di rete che deve essere monitorato. Questa configurazione è di facile manutenzione e offre il vantaggio di avere un sistema centralizzato maggiormente controllabile, ottenendo il giusto equilibrio tra grandi ambienti da dover monitorare e complessità.

2.3 Configurazione dei componenti principali

Il software presente in Zabbix Server, per monitorare un qualunque dispositivo o applicazione, possiede gli strumenti di seguito descritti.

2.3.1 Host

Un host viene configurato per definire server fisici, virtuali, workstation, apparati di rete, dispositivi di memorizzazione o altri device presenti sulla rete. Se si vogliono monitorare particolari parametri è necessario anticipare la creazione del dispositivo al quale verranno correlati gli item. I parametri che possono essere configurati sono:

- **Host name:** il nome del dispositivo. Se su tale host è eseguito anche uno Zabbix Agent allora è necessario avere uniformità tra tale valore e quello specificato all'interno del file di configurazione dell'agent;
- **Visible name** [opzionale]: indica il nome che verrà visualizzato nelle liste, mappe, grafici, ecc.;
- **Groups:** indica il gruppo al quale l'host appartiene;
- **Interfaces:** il tipo di interfaccia del dispositivo Agent, SNMP, JMX e IPMI. una volta selezionata, è necessario inserire l'indirizzo IP o il DNS name, la porta ed, infine, selezionare quale interfaccia deve essere quella di default;
- **Monitored by proxy:** specifica se il dispositivo deve essere monitorato dal Zabbix server (no proxy) oppure da Zabbix proxy. In questo caso si seleziona il nome del proxy corrispondente.

2.3.2 Item

È la modalità base per ottenere informazioni da un sistema. Rappresenta un insieme di informazioni che si vogliono collezionare a partire da un host o da un qualunque altro dispositivo monitorato tramite i check effettuati da Zabbix Server o Proxy. Ogni item rappresenta il valore ottenuto dall'esecuzione di un controllo su un determinato host ed è univocamente identificato tramite una *key* che ne descrive anche il tipo di dato reperito. Ciascun item creato viene associato obbligatoriamente ad un host. Zabbix offre una vasta gamma di template con item predefiniti da poter collegare al dispositivo che si vuole monitorare.

I diversi parametri dell'item sono:

- **Name:** il nome dell'item;
 - **Type:** il tipo di item. Le opzioni possibili sono:
 - Zabbix agent / Zabbix agent (active): vengono usati gli agent di Zabbix (vedi sezione 3.3 per raccogliere le metriche. I primi permettono di effettuare check passivi, i secondi check attivi;
 - Simple check: sono controlli remoti effettuati in modalità agentless. In tal caso, non sono necessari Zabbix agent in quanto il server o il proxy sarà responsabile dell'esecuzione del check;
-

- SNMP agent: il reperimento dati avviene tramite protocollo SNMP ed è tipicamente utilizzato quando si vogliono monitorare dispositivi come stampanti, switch di rete, router o uPS che, solitamente, sono SNMP-enabled e su cui non sarebbe pratico installare determinati sistemi o Zabbix agent;
- SNMP trap: questa soluzione è l'opposta della precedente. In questo caso i dati vengono inviati da un dispositivo SNMP-enabled e memorizzati da Zabbix. Solitamente i trap vengono inviati al verificarsi di specifiche condizioni e l'agent si connette al server sulla porta 162 (in contrapposizione alla porta 161 lato agent che viene utilizzata per il polling);
- Zabbix internal: consente di effettuare il controllo di processi interni a Zabbix; in altre parole permette il monitoraggio di quello che succede sul server o sul proxy;
- Zabbix trapper: permette di prendere dati in ingresso piuttosto che eseguire delle query. Seppur molto simile all'active item, viene solitamente scelto quando si vuole conoscere il dato solo se ha superato una certa soglia stabilita o raggiunta una condizione critica stabilita dal dispositivo (es. un disco guasto o un interfaccia di rete down). È utile per qualsiasi dato ottenuto utilizzando altri strumenti esterni a Zabbix, custom script o qualsiasi altro metodo;
- External check: il server esegue comandi esterni come shell script o binary e memorizza i valori restituiti nell'item, il che consente di trasmettere qualsiasi informazione non disponibile con gli item predefiniti. Anche in questo caso non sono richiesti agenti attivi sul dispositivo che si vuole monitorare;
- HTTP agent: consente di eseguire il polling dei dati da una pagina Web utilizzando il protocollo HTTP/HTTPS con la possibilità anche, ad esempio, di convertire gli header della response in JSON o leggere dati forniti da un API in XML o JSON.

È importante notare che il tipo di item viene impostato per singolo item, non per singolo host. Questo permette una grande flessibilità durante la configurazione degli host da monitorare. Ad esempio, è possibile utilizzare ICMP per controllare la disponibilità generale di un host sulla rete, un agente Zabbix per monitorare lo stato di alcuni servizi, i simple checks per analizzare altri servizi TCP e un trapper per ricevere dati personalizzati, tutto sullo stesso

host. La scelta del tipo di item dipenderà dalla connettività della rete, dal set di funzionalità dell'host monitorato e dalla facilità di implementazione;

- **Key:** specifica esplicitamente quali dati devono essere raccolti per un determinato item. è una sorta di nome tecnico per l'elemento. Il valore della chiave deve essere univoco per ciascun host. Per alcuni tipi di item, il campo che è effettivamente l'identificatore univoco dei dati raccolti potrebbe essere l'OID SNMP (Simple Network Management Protocol Object Identifiers) oppure un sensore IPMI e la Key verrà usata solo per fare riferimento all'elemento. Dunque ogni key esegue un'operazione differente e varia in base al type selezionato in precedenza;
- **Host interface:** permette di selezionare l'interfaccia host;
- **Update interval:** rappresenta l'intervallo di tempo dopo il quale viene aggiornato il dato;
- **New application/Application:** consente di creare una nuova applicazione a cui l'item apparterrà (o di assegnare l'item ad un applicazione preesistente). L'applicazione è un gruppo di item accomunati da una medesima caratteristica; ad esempio, item che permettono di calcolare dati differenti relativi alla CPU potrebbero essere raggruppati in un'applicazione chiamata CPU Check;
- **Preprocessing:** esiste un tab chiamato "Preprocessing" che permette di manipolare i dati ottenuti dagli item prima di salvarli nel database, tramite l'utilizzo di espressioni regolari o operazioni di replace, right/left trim, conversioni tra tipi o esecuzione di codice Javascript.

2.3.3 Trigger

In Zabbix, un trigger è una voce contenente un'espressione logica grazie alla quale è possibile valutare i dati raccolti dagli item e visualizzare lo stato corrente del sistema, in modo da riconoscere automaticamente i problemi relativi agli item monitorati. Dunque gli item si occupano solamente di raccogliere le informazioni. Tali informazioni vengono analizzate attraverso delle espressioni, che consentono di definire una soglia per cui un dato può essere considerato accettabile o meno. Un trigger può assumere tre stati:

- **PROBLEM:** è stata superata una certa soglia e dunque il dato non è più accettabile secondo quanto definito in fase di configurazione;
-

- **OK**: non c'è più match tra l'espressione del trigger e il valore corrente dell'item;
- **UNKNOWN**: si presenta quando non sono presenti abbastanza dati per determinare lo stato corrente. Ad esempio, se si vuole calcolare il valore medio degli ultimi 5 minuti ma non sono presenti dati negli ultimi 10 minuti.

I parametri configurabili sono:

- **Name**: indica il nome che vogliamo dare al trigger. Può contenere le macro come HOST.NAME, HOST.CONN, HOST.IP, ecc.;
 - **Severity**: rappresenta l'importanza del trigger. Ad ogni tipo di severity è associata una rappresentazione grafica del trigger (un colore per ogni severity), un allarme audio (un suono per ogni severity) e diversi user media, cioè canali di notifica differenti per ogni severity. Una caratteristica importante delle severity è che possono essere create e personalizzate con cambiamento di nome e/o di colore;
 - **Problem expression**: espressione logica utilizzata per definire la condizione di un problema. Un problema viene sollevato solamente dopo che tutte le condizioni incluse nell'espressione sono state soddisfatte, ovvero l'espressione restituisce TRUE. Il problema viene risolto non appena l'espressione restituisce FALSE, a meno che non vengano specificate condizioni di recovery aggiuntive nella sezione Recovery expression;
 - **OK event generation**: rappresenta le opzioni per la generazione di un evento di tipo OK. Le opzioni possibili sono:
 - **Expression**: gli eventi OK sono generati basandosi sulla *Problem expression*;
 - **Recovery expression**: gli eventi OK sono generati se la *Problem expression* è valutata a FALSE e la *Recovery expression* è valutata a TRUE;
 - **None**: il trigger non ritorna mai ad uno stato OK da solo;
 - **PROBLEM event generation mode**: rappresenta la modalità per la generazione di eventi relativi ad un problema:
 - **Single**: viene generato un singolo evento quando un trigger passa dallo stato OK a PROBLEM per la prima volta;
 - **Multiple**: viene generato un evento al momento della valutazione di ogni stato PROBLEM del trigger;
-

- **OK event closes:**
 - **All problems** se l'evento OK risolve tutti i problemi del trigger;
 - **All problems if tag values match** se l'evento OK risolve solo quei problemi che fanno match con i valori dell'event tag;
- **Allow manual close:** consente di chiudere manualmente l'evento relativo al problema generato dal trigger.

Alcune volte lo stato di un servizio esposto sulla rete o la raggiungibilità di un dispositivo all'interno di una LAN può dipendere da altri servizi o dispositivi. Ad esempio un server-web potrebbe offrire un front-end per la web-mail e disattivare (per qualche strano motivo) l'HTTP server ogni qualvolta l'SMTP server non è disponibile. Questo vuol dire che il servizio web dipende dal servizio SMTP e configurando il trigger con tale dipendenza è possibile inviare notifiche anche in situazioni come queste.

2.3.4 Notification

Una volta terminata la configurazione di item e trigger, ogni qualvolta un trigger cambia di stato vengono generati degli eventi ai quali è possibile collegare delle azioni da compiere in automatico. Zabbix consente l'invio di notifiche dunque è necessario definire quali tipi di utenti, quando e con quali modalità possono essere notificati al verificarsi di un particolare evento. Per poter inviare e ricevere notifiche da Zabbix è sufficiente:

- **Definire i media types:** I media sono canali di consegna delle notifiche e degli allarmi da parte di Zabbix. Esistono diversi tipi di media (email, telegram) che possono essere configurati in Zabbix. Qui vengono presentati i due tipi di media che sono stati testati e utilizzati durante la simulazione. Per una completa descrizione di tutti gli altri, si rimanda alla documentazione ufficiale [1].
 - **E-mail:** per poter creare questo tipo di canale è necessario configurare l'e-mail come media e assegnare indirizzi specifici agli utenti tramite specifici parametri.
 - **Telegram:** per poter creare questo tipo di canale è necessario registrare un nuovo *bot Telegram*. I bot sono applicazioni di terze parti che vengono eseguite all'interno di Telegram. Gli utenti possono interagire con i bot inviando loro messaggi, comandi e richieste in linea. [2]

- **Configurare un azione:** la visualizzazione di un problema nell'interfaccia web non è sufficiente per la sua risoluzione. Per questo motivo, oltre a notificare l'errore è importante comprendere se è possibile risolvere tale problematica automaticamente. Le *action* assicurano l'esecuzione di un'azione in relazione allo scatenarsi di un trigger. Le azioni consistono in condizioni e operazioni; in sostanza, quando le condizioni sono soddisfatte, allora le operazioni vengono eseguite. Se si vuole eseguire un'operazione in risposta a un evento è necessario configurare un'azione. Le azioni possono essere definite in risposta ad eventi di qualsiasi origine supportata da Zabbix. I tipi di event source supportati sono:
 - Trigger event: quando lo stato del trigger cambia da OK a PROBLEM e viceversa;
 - Discovery event: quando viene effettuato il discovery della rete;
 - Auto registration event: quando nuovi active agent si auto registrano;
 - Internal event: quando gli item non sono più supportati o i trigger passano allo stato di UNKNOWN.

I valori che possono essere settati per una nuova azione sono:

- Name: nome dell'azione che deve essere univoco;
- Conditions: lista delle condizioni.

Nel tab *Operations*, è possibile configurare le operazioni da eseguire. Le opzioni da settare sono:

- Default operation step duration: rappresenta la durata di un'operazione. Ad esempio se settiamo 3600 secondi significa che se viene eseguita l'operazione, dovrà passare un'ora prima del prossimo step;
- Operations: in questo campo vengono mostrate le operazioni con i loro dettagli: *Steps* rappresenta gli step di escalation a cui l'operazione viene assegnata; *Details* specifica il tipo di operazione e i suoi destinatari; *Start in* indica quando l'operazione deve essere eseguita in seguito ad un evento; *Duration* rappresenta la durata dello step dell'operazione; *Action* permette di modificare o rimuovere l'operazione;
- Operation details: questo blocco viene utilizzato per configurare i dettagli dell'operazione, in particolare per selezionare il tipo di operazione:
 - * Send Message: permette di mandare il messaggio all'utente in seguito ad un evento;

- * Remote Command: consente di eseguire automaticamente dei comandi sul dispositivo monitorato se sono soddisfatte determinate condizioni.
- Recovery/Update operations: vengono configurate le operazioni che consentono di notificare gli utenti quando i problemi vengono risolti/subiscono degli aggiornamenti.

2.3.5 Information flow in Zabbix

Una volta configurati, tramite il frontend di Zabbix, il reperimento dei dati (item), la definizione delle soglie (trigger) e le istruzioni da eseguire una volta che una soglia è superata (action), riassumiamo il flusso di informazioni che avviene tra le diverse entità di Zabbix. In riferimento alla figura di esempio 2.2, all'interno dell'installazione di Zabbix Server si è creato un host di prova (*HOST_A*), che contiene un item (*CPU LOAD*). Un trigger fa riferimento a tale item. Ogni volta che l'espressione del trigger uguaglia l'attuale valore dell'item, il trigger passa allo stato PROBLEM. Quando cessa il match con il valore dell'item, il trigger ritorna nello stato OK. Ogni volta che il trigger cambia di stato, un evento viene generato. Ogni evento contiene i dettagli relativi al cambiamento di stato del trigger: quando si è verificato il cambiamento di stato e qual'è il nuovo stato. Se viene configurata un'azione, è possibile aggiungere varie condizioni così che solo alcuni eventi vengano scatenati. Nell'esempio in questione non viene aggiunta alcuna condizione pertanto tutti gli eventi verranno generati. Ogni azione contiene inoltre delle operazioni, che definiscono esattamente cosa deve essere fatto. Alla fine, qualche operazione è effettivamente compiuta, cosa che di solito avviene al di fuori del server Zabbix, come l'invio di un'e-mail o di una notifica tramite Telegram.

è controllare e snellire il flusso di dati monitorati attraverso le reti e il proxy ci dà la possibilità di dividere e separare gli item e i dati sulle diverse reti.

2.4.1 Data flow

Zabbix proxy può operare in due diverse modalità, **attiva** e **passiva**. L'impostazione predefinita è il proxy attivo. In questa configurazione, il proxy avvia le connessioni verso il server Zabbix. Una connessione è usata per recuperare le informazioni di configurazione sugli oggetti da monitorare mentre un'altra è inizializzata per inviare le misurazioni al server. È possibile cambiare e modificare la frequenza di queste due attività impostando le seguenti variabili nel file di configurazione del proxy:

Listing 2.1: /etc/zabbix/zabbix_proxy.conf

```
1 ConfigFrequency=3600
2 DataSenderFrequency=1
```

I valori sono espressi in secondi. Lato server Zabbix, è necessario impostare attentamente il valore di **StartTrappers**. Questo valore deve essere maggiore del numero di tutti i proxy e nodi attivi presenti nell'infrastruttura da monitorare. I processi trapper, infatti, gestiscono tutte le informazioni in arrivo dai proxy.

Si noti che il server eseguirà il fork dei processi aggiuntivi come richiesto, se necessario, ma è fortemente consigliabile eseguire prima tutti i processi necessari durante l'avvio. Ciò ridurrà l'overhead durante il normale funzionamento.

- **Proxy attivo:** facendo riferimento al seguente diagramma:

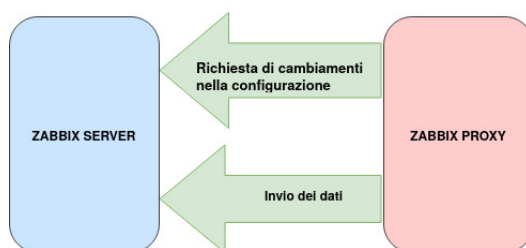


Figura 2.3: Flusso dei dati dello Zabbix Proxy attivo

il server attenderà di ricevere le richieste dal proxy. Il proxy attivo è il modo più efficiente per alleggerire il carico computazionale del server. In effetti, il server rimane solamente in attesa che gli venga chiesto di possibili cambiamenti nella configurazione o di ricevere nuovi dati di monitoraggio. D'altro canto, i proxy vengono solitamente impiegati per monitorare i segmenti di rete protetti con rigide politiche sul traffico in uscita e di solito sono installate all'interno di DMZ. In questo tipo di scenari, normalmente, è molto difficile, se non tramite opportune configurazioni sui firewall e i dispositivi di controllo di accesso alla rete, ottenere l'autorizzazione da parte del proxy per avviare la comunicazione con il server. Sfortunatamente, non è solo una questione di policy. Le DMZ devono essere isolate il più possibile dalle reti interne. In genere, è spesso più facile e più accettato dal punto di vista della sicurezza, avviare una connessione dalla rete interna verso la DMZ. In questo tipo di scenario, il proxy passivo è molto utile;

- **Proxy passivo:** il proxy passivo è quasi un'immagine speculare della configurazione del proxy attivo, come si vede dal diagramma seguente:

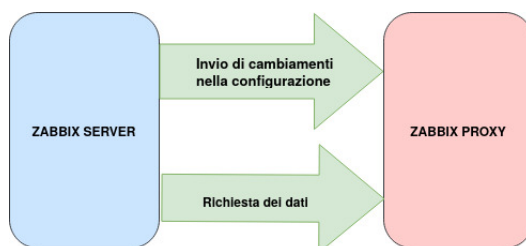


Figura 2.4: Flusso dei dati dello Zabbix Proxy passivo

Con questa configurazione, il server Zabbix contatterà periodicamente il proxy per informarlo di eventuali modifiche nella configurazione e per richiedere i valori relativi agli item monitorati dal proxy.

2.5 Zabbix Agent

Zabbix agent viene distribuito su uno specifico target di un'infrastruttura IT per monitorare attivamente le risorse e le applicazioni locali (dischi rigidi, memoria, statistiche del processore, ecc.). L'agente raccoglie le informazioni operative localmente e segnala i dati al server Zabbix per un'ulteriore elaborazione. In caso

di errori (come un disco rigido pieno o un processo di servizio bloccato), il server Zabbix può avvisare attivamente gli amministratori della particolare macchina che ha segnalato l'errore. Gli agenti Zabbix sono estremamente efficienti grazie all'uso di systemcalls native per la raccolta di informazioni statistiche.

Uno Zabbix agent può eseguire **check passivi e attivi**. In un **check passivo**, l'agent risponde a una richiesta di dati. Il server Zabbix (o proxy) richiede dati, ad esempio, il carico della CPU e lo Zabbix agent restituisce il risultato. I **check attivi** richiedono un'elaborazione più complessa. L'agente deve prima recuperare un elenco di item dal server Zabbix per poterli elaborare in modo indipendente. Ogni volta che sarà completata la fase di generazione dei valori, ne invierà periodicamente di nuovi al server. La possibilità di eseguire check passivi o attivi viene configurata selezionando il Type dell'item da monitorare (per una maggiore descrizione fare riferimento alla sezione 2.3.2). L'agente Zabbix elabora item sia di tipo "Zabbix agent" che "Zabbix agent (active)".

2.5.1 Active vs Passive

Anche se abbiamo discusso che gli agent Zabbix sono attivi o passivi, un agent in realtà non è né l'uno né l'altro: la direzione dei collegamenti è determinata dal tipo di item monitorato. Un agent può (e, per impostazione predefinita, lo fa) operare in entrambe le modalità contemporaneamente. Saremo noi a scegliere quale tipo di item, attivo o passivo, utilizzare. In modo sommario e veloce è possibile affermare che gli **item attivi sono consigliati**. Per comprendere il perché, è necessario fare riferimento alla figura 2.5, in cui viene mostrata la connessione con un agent passivo.

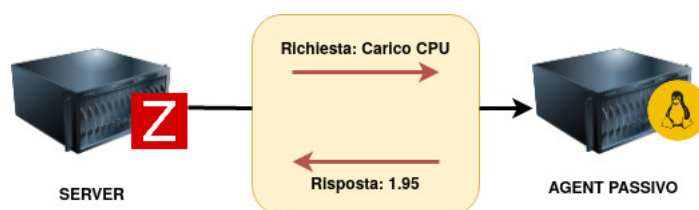


Figura 2.5: Agent passivo

Ad ogni valore richiesto è associata una connessione da instaurare.

Un agent attivo è più complicato in quanto si connette al server Zabbix e chiede di monitorare un elenco di item. Il server quindi risponde con gli item, i loro intervalli e qualsiasi altra informazione rilevante come in figura 2.6.

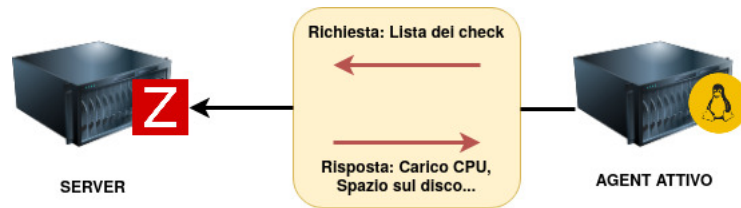


Figura 2.6: Agent attivo (1)

A questo punto, la connessione viene chiusa e l'agent inizia a raccogliere le informazioni. Una volta ottenuti alcuni valori, li invia al server, come in figura 2.7.

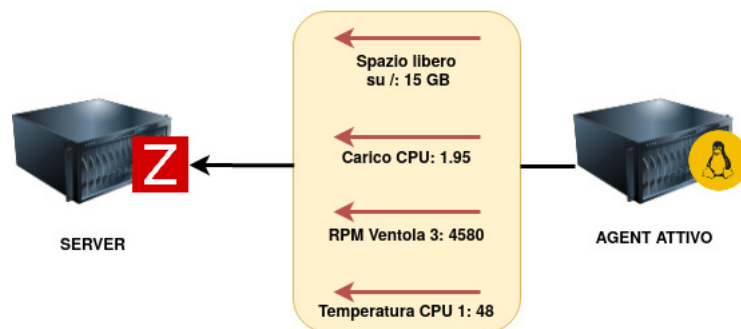


Figura 2.7: Agent attivo (2)

È importante notare che un agent attivo può inviare più valori in un'unica connessione. Di conseguenza, gli agent attivi di solito comporteranno un carico inferiore sul server Zabbix e una quantità inferiore di connessioni di rete, funzionando anche se la topologia di rete o i firewall non consentono la connessione dal server all'agent Zabbix.

Capitolo 3

Configurazione del sistema di monitoring

In questo capitolo verrà illustrata la struttura della rete da monitorare e la composizione dei server e client all'interno dell'ambiente virtuale Virtualbox, evidenziandone anche le connessioni e configurazioni.

3.1 Zabbix Server

All'inizio sono stati configurati i diversi componenti di Zabbix tra cui anche il database MySQL, tutti in esecuzione sulla stessa macchina. Durante gli studi è stata usata l'ultima release LTS di Zabbix 5.0. Il server Zabbix è il processo centrale del software Zabbix ed esegue il polling e il trapping dei dati, calcola i trigger e invia notifiche agli utenti. È il componente centrale a cui agenti e proxy Zabbix riferiscono i dati sulla disponibilità e l'integrità dei sistemi.

Il server è il repository centrale in cui sono archiviati tutti i dati di configurazione, statistici e operativi, ed è l'entità che avviserà attivamente gli amministratori quando sorgono problemi in uno dei sistemi monitorati. Il funzionamento di un server Zabbix di base è suddiviso in tre componenti distinti; essi sono: **server Zabbix**, **frontend web** e il **database**.

Tutte le informazioni di configurazione di Zabbix sono archiviate nel database, con cui interagiscono sia il server che il frontend web. Ad esempio, quando viene creato un nuovo item utilizzando il frontend web (o le API) viene aggiunto alla tabella degli items nel database. Quindi, circa una volta al minuto, il server Zabbix interrogherà la tabella degli item per ottenere l'elenco degli item attivi. Tale

elenco viene quindi archiviato in una cache all'interno del server Zabbix, ed è per questo motivo che possono essere necessari fino a due minuti prima che le modifiche apportate nel frontend Zabbix vengano visualizzate nella sezione dati più recente.

3.1.1 Inizializzazione

Dopo aver creato una macchina virtuale con un adattatore di rete in modalità **Bridged** (vedi sezione 1.3.1), e aver installato CentOS 8.2.2004, procediamo alla configurazione di rete. Viene assegnato l'hostname **centos8** e l'indirizzo IP statico associato alla interfaccia di rete in modalità Bridged:

Listing 3.1: /etc/sysconfig/network-scripts/ifcfg-enp0s3

```
1
2 TYPE="Ethernet" #interfaccia di tipo Ethernet
3 PROXY_METHOD="none" #alcun proxy usato per la connettivita'
4 BROWSER_ONLY="no" #relativo alla configurazione proxy
5 #BOOTPROTO: metodo usato per la configurazione di IPv4
6 #none equivale a static == IP statico
7 BOOTPROTO="none"
8 #DEFROUTE: il NetworkManager considerera' questa
9 #connessione come quella di default
10 DEFROUTE="yes"
11 #IPv4_FAILURE_FATAL: se una connessione e' configurata sia
12 #per IPv4 che per IPv6, con questa opzione impostata su yes,
13 #la configurazione di tale connessione verra' segnalata come
14 #non riuscita, anche se IPv6 e' impostato e IPv4 non lo e'
15 IPv4_FAILURE_FATAL="yes"
16 IPv6INIT="yes" #Inizializza l'interfaccia per l'IPv6
17 IPv6_AUTOCONF="yes" #Utilizzo del Neighbor Discovery Protocol
18 IPv6_DEFROUTE="yes"
19 IPv6_FAILURE_FATAL="no"
20 NAME="enp0s3" #Nome user-friendly per il profilo di connessione.
21 #DEVICE: il nome dell'interfaccia del dispositivo a cui e' associato ...
    questo profilo. La variabile puo' essere omessa quando il profilo ...
    deve essere applicato a piu' dispositivi
22 DEVICE="enp0s3"
23 #ONBOOT: la connessione deve essere instaurata automaticamente (non ...
    solo durante l'avvio)
24 ONBOOT="yes"
25 IPADDR="192.168.1.60" #indirizzo IP statico
26 PREFIX=24 #prefisso della subnet mask
27 GATEWAY="192.168.1.1" # impostato come gateway l'IP di pfSense
```

In ambienti in produzione, è probabile che si debbano gestire configurazioni con **Security-Enhanced Linux (SELinux)** abilitato. Tuttavia, SELinux è piuttosto complesso e fuori dallo scopo di questo lavoro di tesi, pertanto è necessario disabilitare SELinux prima di iniziare con l'installazione di Zabbix.

Procediamo all'installazione del server web Apache:

```
1 [centuser@centos8]$ sudo dnf install httpd
```

e del DBMS MySQL (in realtà nei repository ufficiali CentOS è presente come suo sostituto MariaDB 10.3):

```
1 [centuser@centos8]$ sudo dnf install mysql-server
```

3.1.2 Configurazione del database

Creiamo il database e l'utente che verrà usato dal server. Inoltre impostiamo la password e consentiamo, all'utente appena creato, di eseguire qualsiasi azione sul database precedentemente creato:

```
1 shell> mysql -uroot -p<password>
2 mysql> create database zabbix character set utf8 collate utf8_bin;
3 mysql> create user 'zabbix'@'localhost' identified by '<password>';
4 mysql> grant all privileges on zabbix.* to 'zabbix'@'localhost';
5 mysql> quit;
```

e importiamo lo schema iniziale:

```
1 [centuser@centos8 ~]$ zcat ...
   /usr/share/doc/zabbix-server-mysql*/schema.sql.gz | mysql -uzabbix ...
   -p <password>
```

3.1.3 Installazione e configurazione del server

Installiamo i repository di Zabbix e installiamo lo Zabbix server, il frontend e l'agent. Infine abilitiamo lo start dell'agent e del server al boot di sistema. Per maggiori dettagli riguardo la procedura di installazione si rimanda alla documentazione ufficiale [4].

Una volta completata l'installazione e la creazione del database di MySQL, vengono settati alcuni parametri da modificare:

Listing 3.2: /etc/zabbix/zabbix_server.conf

```
1
2 #valore dell'hostname concorde con quello assegnato alla macchina
3 Hostname=centos8
4
5 #Il valore di ciascuno di questi parametri deve essere uguale a quelli
6 #scelti durante la fase di installazione. E` importante modificarli
7 #manualmente in quanto la procedura di installazione non lo prevede
8 #in modo automatico
9 DBHost=localhost
10 DBName=zabbix
11 DBUser=zabbix
12 DBPassword=<password>
```

Una volta completata la configurazione tramite il wizard del frontend web, è possibile accedere alla dashboard di controllo di Zabbix server.

3.2 Zabbix Proxy

In questa sezione vengono descritte le scelte effettuate durante l'installazione e configurazione del proxy.

3.2.1 Inizializzazione

Una volta istanziata una macchina con sistema operativo CentOS 8.2.2004 ed hostname **centos8-ZabbixProxy** è stata effettuata la configurazione dell'indirizzo IP da assegnare staticamente all'interfaccia di rete della Internal Network netA:

Listing 3.3: /etc/sysconfig/network-scripts/ifcfg-enp0s8

```
1 TYPE="Ethernet" #interfaccia di tipo Ethernet
2 PROXY_METHOD="none" #alcun proxy usato per la connettivita'
3 BROWSER_ONLY="no" #relativo alla configurazione proxy
4 #BOOTPROTO: metodo usato per la configurazione di IPv4
5 #none equivale a static == IP statico
6 BOOTPROTO="none"
7 #DEFROUTE: il NetworkManager considerera' questa
8 #connessione come quella di default
```

```
9 DEFROUTE="yes"
10 #IPV4_FAILURE_FATAL: se una connessione e' configurata sia
11 #per IPv4 che per IPv6, con questa opzione impostata su yes,
12 #la configurazione di tale connessione verra' segnalata come
13 #non riuscita, anche se IPv6 e' impostato e IPv4 non lo e'
14 IPV4_FAILURE_FATAL="yes"
15 IPV6INIT="yes" #Inizializza l'interfaccia per l'IPv6
16 IPV6_AUTOCONF="yes" #Utilizzo del Neighbor Discovery Protocol
17 IPV6_DEFROUTE="yes"
18 IPV6_FAILURE_FATAL="no"
19 NAME="enp0s17" #Nome user-friendly per il profilo di connessione.
20 #DEVICE: il nome dell'interfaccia del dispositivo a
21 #cui e' associato questo profilo. La variabile puo'
22 #essere omessa quando il profilo deve essere
23 #applicato a piu' dispositivi
24 DEVICE="enp0s17"
25 #ONBOOT: la connessione deve essere instaurata automaticamente (non ...
    solo durante l'avvio)
26 ONBOOT="yes"
27 IPADDR="192.168.57.30" #indirizzo IP statico
28 PREFIX=24 #prefisso della subnet mask
29 GATEWAY="192.168.57.100" # impostato come gateway l'IP del router della ...
    rete fisica
```

3.2.2 Installazione e configurazione del proxy

Prima di procedere all'installazione, facendo riferimento a [5] è importante installare la giusta versione del proxy per evitare problemi di incompatibilità con il server, infatti:

"Zabbix 5.0.x server can only work with Zabbix 5.0.x proxies. Zabbix 5.0.x proxies can only work with Zabbix 5.0.x server."

Una volta installato il proxy è possibile passare alla sua configurazione.

Ecco alcuni parametri da modificare una volta completata l'installazione e la creazione del database di MySQL:

Listing 3.4: /etc/zabbix/zabbix_proxy.conf

```
1 Hostname=centos8-ZabbixProxy
2 ProxyMode=0 #proxy attivo
3 #Se ProxyMode=0, specificare l'IP/DNS dello Zabbix Server da #cui ...
    ricevere le info di configurazione e a cui mandare i #dati, ...
```

```
altrimenti se ProxyMode=1 specificare una lista di #IP/DNS dei ...
Zabbix Server. Le connessioni in ingresso #verranno accettate ...
solamente da questi indirizzi elencati.
4 Server=192.168.1.60
5 DBHost=localhost
6 DBName=zabbix_proxy
7 DBUser=zabbix
8 DBPassword=<password>
```

3.2.3 Creazione del proxy dal front-end web

Per concludere la creazione del proxy, è necessario informare il server della sua presenza.

Il **Proxy name** deve essere lo stesso specificato nel file *zabbix_proxy.conf* come indicato nella sezione 3.2.2. Scegliamo come **Proxy mode** la modalità Active. È possibile inoltre specificare per ragioni di sicurezza, a partire dalla versione 4.0, nel campo **Proxy address**, gli indirizzi IP dai quali il server accetterà le connessioni in ingresso.

3.3 Zabbix Agent

Verrà illustrato il processo di installazione e configurazione dell'agent sulla macchina Ubuntu Server 18.04.

3.3.1 Inizializzazione del server da monitorare

Una volta creata una macchina virtuale con sistema operativo Ubuntu Server 20.04 ed hostname **ubuntuserver** è stata effettuata la configurazione dell'indirizzo IP da assegnare staticamente all'interfaccia di rete della Internal Network netA. A partire dalla versione 17.10 di Ubuntu è possibile configurare le impostazioni di rete tramite l'utilizzo del tool Netplan [6].

Netplan è un utility per configurare facilmente la rete su un sistema Linux. È sufficiente creare un file YAML che descriva le interfacce di rete da configurare. Da questa descrizione, Netplan genererà tutta la configurazione necessaria per il tool di rendering presente all'interno del sistema (NetworkManager o Systemd-network).

Listing 3.5: /etc/netplan/00-installer-config.yaml

```
1 #This is the network config written by 'subiquity'
2 network:
3   ethernets:
4     enp0s8:
5       addresses:
6         - 192.168.57.20/24 #IP statico assegnato alla macchina
7       gateway4: "192.168.57.100" #IP di pfSense
8       nameservers:
9         addresses:
10        - "192.168.57.100" #risoluzione effettuata da pfSense
11        - "8.8.8.8" #risoluzione tramite DNS google
12 version: 2
13 renderer: NetworkManager
```

3.3.2 Installazione e configurazione dell'agent

Una volta identificato il codename della release di Ubuntu Server installata, è possibile scaricare e installare il package `zabbix-agent`, abilitandolo in `systemd` all'avvio. Per una descrizione maggiormente dettagliata della procedura di installazione si rimanda alla documentazione ufficiale [7].

È necessario modificare alcuni parametri dell'agent per poter completare la configurazione. Di seguito ne vengono mostrati alcuni tra i più rilevanti modificati all'interno del file `/etc/zabbix/zabbix_agentd.conf` :

E' possibile limitare i controlli sull'agent creando una blacklist, una whitelist o una loro combinazione, come mostrato di seguito.

```
1 AllowKey=system.run[*]
2 AllowKey=system.cpu.load
3 AllowKey=net.if.in[*]
4 AllowKey=vfs.file.md5sum[]
5 AllowKey=net.*.service[*]
6
7 DenyKey=*
```

La entry `EnableRemoteCommand=1` risulta essere deprecata a partire dalla versione 5, in favore delle più sicure allow/deny list presentate in precedenza. È comunque necessario fino alla 5.0.2 lasciare ad 1 il suo valore per abilitare i comandi remoti.

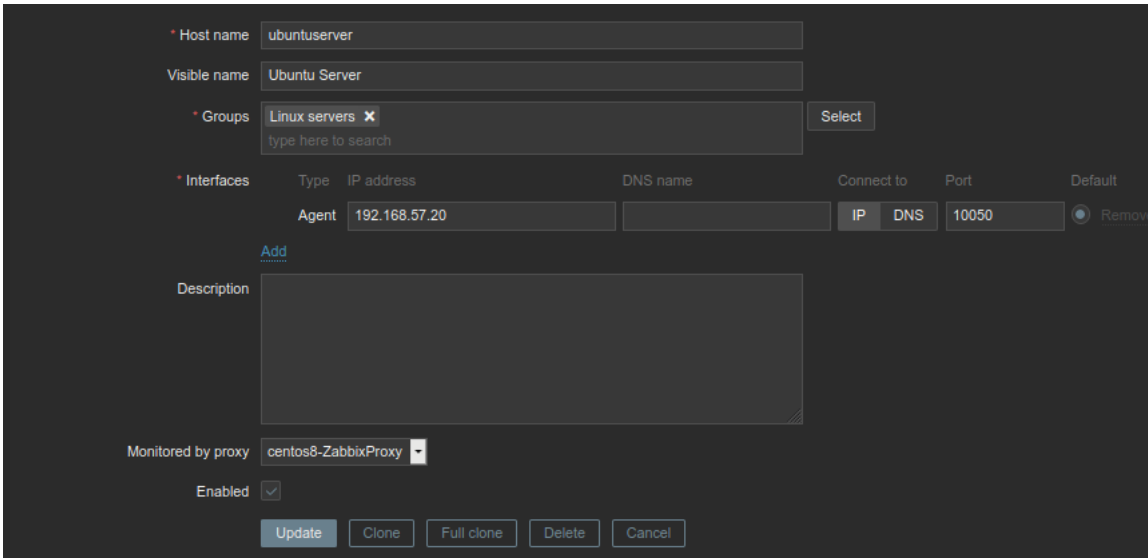
Inoltre, è possibile mantenere nei log lo storico dei comandi remoti eseguiti, attraverso il parametro `LogRemoteCommand=1`.

È necessario specificare una lista di IP/DNS separati da una virgola, che identificano i server/proxy Zabbix per i passive check. Le richieste in ingresso verranno accettate solamente dagli IP specificati in questo parametro come di seguito: **Server=192.168.57.30 #IP di Zabbix Proxy**.

In più, bisogna specificare una lista di coppie IP/DNS:porta separati da una virgola, che identificano i server/proxy Zabbix per gli active check, in questo modo: **ServerActive=192.168.57.30**. Se tale parametro non è specificato, i controlli attivi sono disabilitati. Infine bisogna specificare un valore per l'hostname: **Hostname=ubuntuserver**, nel caso in cui vengano effettuati dei check attivi. Tale valore deve coincidere con quello specificato nel frontend.

3.3.3 Creazione dell'host dal front-end

Facendo riferimento alla procedura di creazione di un host tramite la interfaccia web di Zabbix esposta nella sezione 2.3.1, creiamo l'host che identifica l'Ubuntu Server all'interno della rete:



The screenshot displays the Zabbix web interface for creating a new host. The form includes the following fields and options:

- Host name:** Input field containing 'ubuntuserver'.
- Visible name:** Input field containing 'Ubuntu Server'.
- Groups:** A dropdown menu showing 'Linux servers' with a search prompt 'type here to search' and a 'Select' button.
- Interfaces:** A table with columns for Type, IP address, DNS name, Connect to, Port, and Default. One interface is listed: Type 'Agent', IP address '192.168.57.20', Connect to 'IP', Port '10050', and a 'Remove' button. An 'Add' link is visible below the table.
- Description:** A large text area for entering a description.
- Monitored by proxy:** A dropdown menu set to 'centos8-ZabbixProxy'.
- Enabled:** A checked checkbox.
- Buttons:** 'Update', 'Clone', 'Full clone', 'Delete', and 'Cancel'.

Figura 3.1: Creazione dell'host Ubuntu Server dalla interfaccia web di Zabbix

3.4 Client-side

Sono state create due virtual-machine, una per ciascuna rete interna, così da poter simulare, come fatto nel capitolo 4, l'isolamento di una subnet a seguito di un attacco o vulnerabilità rilevata. Il sistema operativo usato per entrambe le macchine è Linux Mint 19.1 64 bit e l'utility per la configurazione della rete è di nuovo Netplan [6]. Per il *guest* della **netA** si è scelto l'IP **192.168.57.10** e gateway **192.168.57.100**, ovvero l'IP di *pfSense* sull'interfaccia LAN. Infine per il *guest* della **netB** si è scelto l'IP **192.168.58.10** e gateway **192.168.58.10**, ovvero l'IP di *pfSense* sull'interfaccia OPT1.

Capitolo 4

Simulazione di una procedura di difesa in risposta ad un attacco informatico

In questo capitolo viene simulato un cyber-attacco ai danni di uno dei server presenti nella rete con conseguente attivazione del trigger precedentemente creato. Infine viene mostrata una procedura di difesa rapida tramite il Bot Telegram sviluppato [8].

4.1 Configurazione di Ubuntu Server

In questa sezione viene illustrata la creazione di un item e il relativo trigger di prova, per poter simulare un evento critico sulla macchina server Ubuntu. Inoltre viene illustrata la configurazione dei permessi per l'esecuzione dei comandi necessari al monitoring della macchina.

4.1.1 Creazione dell'item

Si è scelto di simulare l'attacco monitorando il file contenente le password cifrate */etc/shadow*, per poter intercettare un qualsiasi cambiamento delle password non autorizzato. In particolare sono stati scelti i seguenti valori per i parametri:

- **Name:** *MD5SUM of /etc/shadow* in quanto viene calcolato l'MD5SUM del file;

- **Type:** *Zabbix agent(active)* si è scelto tale tipo di item a seguito delle caratteristiche evidenziate nella sezione 2.3.2;
- **Key:** `system.run[sudo /usr/bin/md5sum /etc/shadow | cut -f 1 -d " "]` viene calcolato l'MD5SUM sul file contenente le password cifrate. Il risultato ottenuto è nella forma: `"f1e61a7698487d30392b61c72e173a1f /etc/shadow"`. Viene dunque usato il comando **cut** per poter estrarre solamente il primo campo (**-f 1**) sulla base di un carattere separatore specifico (**-d " "**) ottenendo come risultato: `"f1e61a7698487d30392b61c72e173a1f"`. È importante notare come sia stato necessario usare il programma **sudo** per poter ottenere i permessi necessari alla lettura del file. Si rimanda alla sezione 4.1.4 per una maggiore spiegazione della configurazione;
- **Type of information:** il tipo di informazione è testuale;
- **Update interval:** è stato scelto un intervallo di 30 secondi per poter facilitare la simulazione.

4.1.2 Creazione del trigger

Una volta creato l'item, associamogli un controllo da eseguire tramite la creazione di un trigger. Sono stati assegnati i seguenti valori a ciascun parametro:

- **Name:** *Password changed*;
- **Expression:** `ubuntuserver:system.run[sudo /usr/bin/md5sum /etc/shadow | cut -f 1 -d " "].diff()=1` consente di verificare se il valore appena rilevato e quello precedente differiscono. Nel caso siano diversi, la funzione ritorna un valore pari a 1 altrimenti ritorna 0;
- **OK event generation:** *None* per evitare che il problema si risolva in modo automatico con il controllo successivo. Per ulteriori dettagli fare riferimento alla sezione 2.3.3.

4.1.3 Creazione dell'action

Infine è necessario creare l'azione da eseguire a seguito dell'attivazione del trigger. A seguito del rilevamento di un cambiamento all'interno del file `/etc/shadow`, verrà inviata una notifica tramite il Bot Telegram configurato dalla interfaccia web. Per una spiegazione dettagliata della configurazione del Bot Telegram, si rimanda alla

documentazione ufficiale [2]. Una volta collegata l'azione da eseguire al trigger precedentemente creato, nel tab *Operations* vengono definite le azioni da compiere. Per questo esempio si è scelto di inviare un messaggio all'utente *Zabbix Administrator* tramite il Bot Telegram.

4.1.4 Gestione dei permessi

L'agente Zabbix su UNIX è progettato per essere eseguito come utente non root [9], nello specifico, eseguendo il comando `ps aux | grep zabbix_agentd`, è possibile individuare l'utente proprietario del processo. L'accesso in lettura e scrittura al file `/etc/shadow` è consentito solamente ad utenti privilegiati (*root*), dunque è necessario configurare i permessi di accesso al file. Ciò è possibile tramite la configurazione del programma **sudo** (substitute **user do**) che permette di eseguire altri programmi assumendo l'identità (e di conseguenza anche i privilegi) di altri utenti. Il programma ha bisogno di un particolare file, `/etc/sudoers`, che è il responsabile della politica di assegnazione dei privilegi sudo agli utenti di un sistema [10]. Di seguito la policy creata per l'utente *zabbix*:

```
1 #Consente all'utente zabbix (1), da qualsiasi terminale (2)
2 #evitando l'immissione della password (3), di calcolare l'md5sum del file
3 #/etc/shadow (4)
4 zabbix ALL=NOPASSWD: /usr/bin/md5sum /etc/shadow
5 # (1) (2) (3) | (4) |
```

4.2 Configurazione di pfSense

In questa sezione vengono espone le scelte effettuate durante lo studio, relative alla creazione delle regole del firewall e gli script PHP realizzati per poterle manipolare.

4.2.1 Regole di pfSense

Si è scelto di evitare la creazione delle regole *"on-demand"* in fase di risposta alla simulazione di attacco presentata in questo capitolo, ma di configurare e creare le due regole (una per l'interfaccia della LAN_57 e l'altra per l'interfaccia della LAN_58) in modo da poterle semplicemente abilitare o disabilitare tramite il Bot Telegram sulla base delle notifiche ricevute da Zabbix Server. Si fa notare inoltre, come sia

stato necessario creare due regole e non una sola per poter isolare completamente la subnet **192.168.57.0/24**, in quanto pfSense processa le regole esclusivamente sulla base della interfaccia di rete dalla quale i pacchetti provengono. Pertanto è necessario configurare una regola sulla interfaccia di rete della *LAN_57* per poter bloccare qualsiasi traffico in uscita e una regola sulla interfaccia di rete della *LAN_58* per bloccare il traffico in ingresso diretto verso la *LAN_57*:

- **WAN**: con riferimento alla figura 4.1 si nota come sia stato abilitato il traffico in ingresso sulla porta 22 (SSH) e sulla porta 80 (HTTP) del firewall da qualsiasi dispositivo della rete WAN (192.168.1.0/24). Dietro tale scelta c'è la necessità di dover condurre dei test o compiere procedure di debug di eventuali problemi. Inoltre tale configurazione consente di eseguire alcuni comandi tramite SSH dallo Zabbix Server e comandare dunque da remoto quali regole attivare o disattivare in automatico;

States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input checked="" type="checkbox"/>	0/0 B	*	Reserved Not assigned by IANA	*	*	*	*	*	Block bogon networks	
<input type="checkbox"/>	<input checked="" type="checkbox"/> 0/24 KIB	IPv4 TCP	WAN net	*	This Firewall	22 (SSH)	*	none	Allow SSH from net of Zabbix Server [192.168.1.0/24]	
<input type="checkbox"/>	<input checked="" type="checkbox"/> 0/10 KIB	IPv4 TCP	WAN net	*	This Firewall	80 (HTTP)	*	none	Allow HTTP from net of Zabbix Server [192.168.1.0/24]	

Figura 4.1: Regole della WAN

- **LAN_57**: in riferimento alla figura 4.2 è possibile notare che esiste una regola *Anti-lockout* abilitata per impostazione predefinita che impedisce la configurazione delle regole del firewall in modo tale da bloccare l'utente dall'interfaccia web o dall'accesso tramite SSH. È impostata come prima regola così da avere la precedenza su tutte le altre. Inoltre sono state create altre due regole: la prima, che risulta disabilitata, **blocca** (loggando tutti i pacchetti) il traffico proveniente dalla *LAN_57* e diretto verso qualsiasi IP (e porta) diverso dalla *LAN_57*. La seconda **consente** il traffico dalla *LAN_57* verso qualsiasi IP e porta;

States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
✓ 0 / 340 KIB	*	*	*	LAN_57 Address	80 22	*	*		Anti-Lockout Rule	⚙️
✗ 0 / 0 B	IPv4+6	LAN_57 net	*	! LAN_57 net	*	*	none		Allow only packets inside LAN_57 (cyber-attack on UbuntuServer)	🔗 🛠️ 📄 🗑️
✓ 2 / 12.51 MIB	IPv4+6	LAN_57 net	*	*	*	*	none		Default allow LAN to any rule	🔗 🛠️ 📄 🗑️

Figura 4.2: Regole della LAN_57

- **LAN_58**: basandosi sulla figura 4.3, sono presenti due regole: la prima, che è disabilitata, **blocca** (loggando tutti i pacchetti) il traffico proveniente da qualsiasi IP e porta e diretto verso la *LAN_57*. La seconda invece **consente** il traffico dalla *LAN_58* verso qualsiasi IP e porta.

States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
✗ 0 / 0 B	IPv4+6	*	*	LAN_57 net	*	*	none		Block packets comes from any to LAN_57 (cyber-attack on UbuntuServer)	🔗 🛠️ 📄 🗑️
✓ 42 / 1.94 MIB	IPv4+6	LAN_58 net	*	*	*	*	none		Default allow LAN to any rule	🔗 🛠️ 📄 🗑️

Figura 4.3: Regole della LAN_58

4.2.2 Scripts PHP

In questa sezione vengono presentati gli script PHP sviluppati durante il periodo di studio, che verranno eseguiti dal Bot Telegram illustrato nella sezione 4.3. A fronte di diverse soluzioni già sviluppate per poter automatizzare la gestione di pfSense, si è deciso di implementare una soluzione altamente personalizzabile e flessibile. Sono stati creati per la simulazione, due script in *PHP*, per poter manipolare il file di configurazione del firewall `/cf/conf/config.xml`. Infatti pfSense memorizza tutte le impostazioni di configurazione, comprese le impostazioni per i package, in un file in formato XML.

Amministrazione delle regole

È necessario includere alcuni script PHP già presenti all'interno della distribuzione, nella cartella `/etc/inc`:

```
1 require_once("filter.inc");
2 require_once("config.inc");
```

Lo script accetta in input un parametro testuale, **enable/disable**, che consente di abilitare/disabilitare le regole create nella sezione 4.2.1. Successivamente viene letta la configurazione del firewall e caricata all'interno di un array. La lettura avviene dalla cache o dal file config.xml sulla base di un parametro booleano. Sono state inoltre inizializzate due variabili con il valore dell'ID delle regole da attivare/disattivare:

```
1 global $config;
2 $config = parse_config(true);
3 $track_id_LAN57 = '1596729173';
4 $track_id_LAN58 = '1596738586';
```

La seguente funzione costituisce il cuore dell'intero script. Infatti, per ogni regola estratta dall'array *\$config* e indirizzata "by reference" per poterne cambiare il valore successivamente, si controlla se è una delle regole di interesse. In caso positivo, viene settato il valore del campo "disabled" in base al parametro passato dallo script, altrimenti ne viene fatto l'unset.

```
1 function isolateLAN57($isolate = 'enable') {
2
3     global $track_id_LAN57;
4     global $track_id_LAN58;
5     global $config;
6     global $result;
7
8     foreach ($config[filter][rule] as &$value) {
9         if (strcmp($value[tracker], $track_id_LAN57) === 0 ) {
10            //this is the rule on lan57 that allow only packets inside LAN_57
11            if (strcmp($isolate, 'disable') === 0) {
12                $value[disabled] = true;
13            } else {
14                unset($value[disabled]);
15            }
16        } elseif (strcmp($value[tracker], $track_id_LAN58) === 0 ) {
17            //this is the rule on lan58 that block packets comes from any ...
18            //to LAN_57
19            if (strcmp($isolate, 'disable') === 0) {
20                $value[disabled] = true;
21            } else { unset($value[disabled]);
```

```
21         }
22     }
23 }
24 # Modifica l'array ponendo una flag di
25 # successo se il codice e' stato eseguito fin qui
26 $result["success"] = true;
27 }
```

Infine viene scritta la nuova configurazione e viene dato come output un valore ottenuto dall'*or assignment* con la chiamata a funzione *filter_configure()* la quale ricarica le regole in modo asincrono.

Visualizzazione delle regole

Lo script che gestisce la visualizzazione delle regole, contiene la parte di inizializzazione e recupero della configurazione uguale allo script precedente. È richiesto, in input, di specificare tramite un parametro stringa, l'interfaccia da cui estrarre le regole del firewall. Con un ciclo *for*, vengono identificate all'interno dell'array, le regole di interesse e viene generato un *JSON (Javascript Object Notation)* come output grazie alla funzione built-in *json_encode*, che ritorna una stringa JSON-encoded.

4.3 Sviluppo del Bot Telegram

In questa sezione viene illustrato per grandi linee lo sviluppo del Bot Telegram e le scelte compiute durante il processo di realizzazione e di test dell'applicativo.

Telegram è un servizio di messaggistica istantanea basato su cloud ed erogato senza fini di lucro dalla società *Telegram LLC*. Da giugno 2015, Telegram ha introdotto una piattaforma per permettere, a sviluppatori terzi, di creare i Bot. I Bot sono degli account Telegram, gestiti da un programma, che offrono molteplici funzionalità con risposte immediate e completamente automatizzate. I messaggi e i comandi inviati dall'utente, vengono processati dal software in esecuzione sul server del Bot. Telegram mette a disposizione una serie di API, tramite un'interfaccia HTTPS, per gli sviluppatori che vogliono implementare chatBot, ma sono disponibili anche librerie e framework di terze parti.

4.3.1 Python

Python è un linguaggio di programmazione dinamico ad alto livello, orientato agli oggetti. Offre un forte supporto all'integrazione con altri linguaggi e programmi ed è fornito di una estesa libreria standard. Rilasciato pubblicamente per la prima volta nel 1991, supporta diversi paradigmi di programmazione, come quello object-oriented, quello imperativo e quello funzionale. Offre una tipizzazione dinamica forte e il controllo dei tipi viene eseguito a runtime (dynamic typing). Usa un garbage collector per la liberazione automatica della memoria. È fornito di una libreria built-in estremamente ricca, la sintassi è pulita così come i suoi costrutti. I blocchi logici infatti, vengono costruiti semplicemente allineando le righe allo stesso modo, incrementando la leggibilità e l'uniformità del codice. Python è un linguaggio pseudo-interpretato: un interprete si occupa di analizzare il codice sorgente (semplici file testuali con estensione *.py*) e, se sintatticamente corretto, di eseguirlo. Questo lo rende un linguaggio portabile. Una volta scritto un sorgente e ottenuta la versione corretta dell'interprete, esso può essere interpretato ed eseguito sulla gran parte delle piattaforme attualmente utilizzate. Si è deciso di utilizzare questo linguaggio per le potenzialità appena citate, in particolare perché è considerato tra i migliori linguaggi per l'implementazione di Bot Telegram e perché sul web vanta di community numerose e attive in questo campo.

4.3.2 python-telegram-bot

È una libreria che fornisce un'interfaccia Python per l'utilizzo delle API messe a disposizione da Telegram per l'implementazione di Bot. È compatibile con le versioni Python2.7, 3.3+ e PyPy. Oltre alla pura implementazione delle API, la libreria presenta una serie di classi di alto livello per rendere lo sviluppo di Bot semplice e immediato. Queste classi sono contenute nel sottomodulo *telegram.ext*. La libreria è rilasciata sotto licenza LGPL-3. Per ulteriori dettagli si rimanda al sito ufficiale. [11]

4.3.3 paramiko

Paramiko [12] è un'implementazione in Python del protocollo SSHv2, che fornisce funzionalità sia client che server. Paramiko è una libreria scritta in Python (2.7, 3.4+) che supporta i protocolli SSHv1 e SSHv2, consentendo la creazione di client e la connessione a server SSH. Dipende da PyCrypto e dalle librerie di crittografia per tutte quelle operazioni di crittografia e consente la creazione di tunnel crittogra-

fati locali, remoti e dinamici. Tra i principali vantaggi di questa libreria, possiamo evidenziare che:

- incapsula le difficoltà legate all'esecuzione di script automatizzati su server SSH in un modo comodo e di facile comprensione per qualsiasi programmatore;
- supporta il protocollo SSH2 tramite la libreria PyCrypto, che lo utilizza per implementare tutti quei dettagli di crittografia a chiave pubblica e privata;
- consente l'autenticazione tramite chiave pubblica/privata o autenticazione con utente e password.

4.3.4 Bot Telegram

In questa sezione vengono illustrate e descritte le componenti e funzioni principali del Bot Telegram realizzato.

Sono state scritte diverse funzioni Python per ciascuna feature offerta dal Bot Telegram come mostrato in figura 4.4.

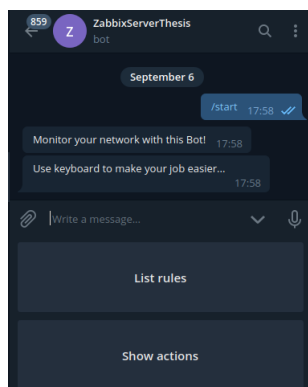


Figura 4.4: Funzionalità del Bot Telegram

In particolare, è possibile elencare le regole create per ciascuna interfaccia (vedi figura 4.5) ed abilitare o disabilitare una regola preesistente grazie ad una comoda tastiera con bottoni interattivi (vedi figura 4.6).

Qualsiasi dato viene reperito sul momento da pfSense, servendosi di una connessione SSH grazie alla libreria Python *paramiko* presentata nella sezione 4.3.3. La funzione che consente di eseguire i comandi remoti è la seguente:

```
1 def exec_command(command_pfSense):
```

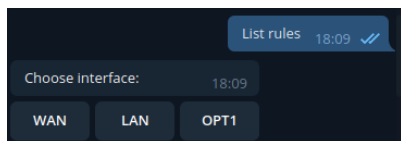


Figura 4.5: Selezione dell'interfaccia

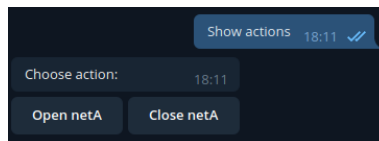


Figura 4.6: Selezione dell'azione

```

2   global username_pfSense
3   global pwd_pfSense
4   global ip_pfSense
5   result = None
6   client = paramiko.SSHClient()
7   client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
8   client.connect(ip_pfSense, username=username_pfSense, ...
9                 password=pwd_pfSense)
9   stdin, stdout, stderr = client.exec_command(command_pfSense)
10  # rstrip(): This method returns a copy of the string
11  #           in which all chars have been stripped from
12  #           the end of the string (default whitespace characters).
13  firstline = stdout.readlines()[0].rstrip()
14  client.close()
15  return firstline

```

Ciascuno script PHP restituisce in output un JSON contenente un ulteriore JSON in corrispondenza del campo 'result', che può assumere il valore {'success':True} oppure {'success':False} sulla base del risultato dell'operazione eseguita. Tale output viene restituito dalla funzione *exec_command*. In base al tipo di script PHP eseguito, è possibile ottenere all'interno del JSON anche un ulteriore output come risultato dell'azione richiesta. Di seguito una parte del JSON contenente la lista delle regole richieste per l'interfaccia WAN:

```

1 {
2   "result":{
3     "success":true
4   },
5   "rules":[
6     {
7       "id":"",
8       "tracker":"1596820672",
9       "type":"pass",
10      "interface":"wan",
11      "ipprotocol":"inet46",
12      "statetimeout":"",

```



```
13     "statetype": "keep state",
14     "os": "",
15     "protocol": "tcp",
16     "source": {
17         "network": "wan"
18     },
19     "destination": {
20         "network": "(self)",
21         "port": "22"
22     },
23     "log": "",
24     "descr": "Allow SSH from net of Zabbix Server [192.168.1.0\24]",
25     "created": {
26         "time": "1596820672",
27         "username": "admin@192.168.57.10 (Local Database)"
28     },
29     "updated": {
30         "time": "1598889476",
31         "username": "admin@192.168.57.10 (Local Database)"
32     }
33 }
34 ]
35 }
```

Capitolo 5

Conclusioni

Con il lavoro svolto, durante il periodo di internato, è stato possibile sfruttare le potenzialità messe a disposizione da *Zabbix* per studiare e comprendere l'importanza del monitoring e della fault detection di qualsiasi infrastruttura IT.

In particolar modo, la diretta sperimentazione e simulazione in ambiente virtuale, ha consentito di capire a fondo le problematiche che potrebbero presentarsi in un contesto reale e di come il software *Zabbix* possa essere uno strumento molto valido grazie alla sua flessibilità e intuitività. Un infrastruttura informatica anche molto complessa tipicamente risulta essere composta da svariati dispositivi con caratteristiche e finalità diverse. *Zabbix* si è rivelato essere uno strumento altamente integrabile con la maggior parte di dispositivi di sicurezza e protezione esistenti. Il sistema di dialogo tra *Zabbix* e il firewall *pfSense*, implementato tramite il *Bot Telegram*, è stato creato per risolvere un problema pratico ed è risultato essere una soluzione altamente personalizzabile secondo le proprie esigenze, a differenza di altre soluzioni poco flessibili presenti sul panorama odierno. Il sistema è stato sviluppato in modo da essere utilizzato sia da utenti esperti, sia da quelli che possiedono meno dimestichezza nell'utilizzo dello smartphone. Tramite un interfaccia intuitiva infatti, è possibile agire direttamente sul firewall *pfSense* a seguito di segnalazioni ricevute da *Zabbix*.

Anche se il risultato finale rispetta le caratteristiche desiderate, sarebbe interessante introdurre ulteriori funzioni che possano ampliare le possibilità di gestione del firewall da remoto. Tra queste, potrebbe rivelarsi utile, la capacità di creare delle viste per ciascun tipo di utente che interagisce con il *Bot Telegram* e la possibilità di gestire maggiormente le regole del firewall e le sue impostazioni.

Bibliografia

- [1] Receiving problem notification. [Online]. Disponibile: <https://www.zabbix.com/documentation/current/manual/quickstart/notification>
- [2] Zabbix + Telegram. [Online]. Disponibile: <https://www.zabbix.com/integrations/telegram>
- [3] R. O. Patrik Uytterhoeven, *Zabbix 4 Network Monitoring*, 3° edizione. Packt, 2019.
- [4] Download and install zabbix. [Online]. Disponibile: https://www.zabbix.com/download?zabbix=5.0&os_distribution=red_hat_enterprise_linux&os_version=8&db=mysql
- [5] Version compatibility. [Online]. Disponibile: <https://www.zabbix.com/documentation/current/manual/appendix/compatibility>
- [6] super-fortnight. [Online]. Disponibile: <https://netplan.io/>
- [7] Download and install zabbix agents. [Online]. Disponibile: https://www.zabbix.com/download_agents
- [8] super-fortnight. [Online]. Disponibile: <https://github.com/lilpilgit/super-fortnight>
- [9] Agent zabbix. [Online]. Disponibile: <https://www.zabbix.com/documentation/current/manual/concepts/agent>
- [10] D. A. Tevault, *Mastering Linux Security and Hardening*. Packt, 2018.
- [11] python-telegram-bot. [Online]. Disponibile: <https://python-telegram-bot.org/>
- [12] Paramiko. [Online]. Disponibile: <http://www.paramiko.org/>

- [13] S. K. L. Andrea Dalle Vacche, *Zabbix Network Monitoring Essentials*. Packt, 2015.
- [14] J. P. Chris Buechler, *The pfSense Book*. Electric Sheep Fencing LLC, 2019.
-