Alma Mater Studiorum · Università di Bologna

DIPARTIMENTO DI INFORMATICA - SCIENZA E INGEGNERIA
Corso di Laurea in Ingegneria e scienze informatiche

Mitigazione delle minacce informatiche in ambiente Linux: analisi e sviluppo di soluzioni eBPF e XDP

Relatore:
Chiar.mo Prof.
Andrea Piroddi

Correlatrice:
Dott.ssa Serena Ferracci

Presentata da:
Alessandro Martini

Sessione IV
Anno Accademico 2023-2024

Not everyone who works hard is rewarded . . . However, all those who succeed have worked hard!

Hajime No Ippo - Kamogawa Genji

Introduzione

Nell'era digitale in cui ci troviamo immersi, la sicurezza informatica è diventata una delle principali preoccupazioni per individui, aziende e istituzioni governative. In particolare, i sistemi operativi Linux, noti per la loro affidabilità e flessibilità, sono sempre più soggetti ad attacchi informatici sofisticati e mirati. Di fronte a questa crescente minaccia, è fondamentale sviluppare strumenti di monitoraggio avanzati in grado di rilevare tempestivamente comportamenti sospetti e mitigare potenziali rischi per la sicurezza. Il focus principale di questa tesi è proprio quello di affrontare questa sfida, concentrandosi sul monitoraggio in tempo reale di sistemi Linux per rilevare comportamenti anomali e minacce alla sicurezza. Il monitoraggio in tempo reale rappresenta un'importante strategia difensiva, in quanto consente di identificare e rispondere prontamente a potenziali attacchi, garantendo un funzionamento sicuro e affidabile del sistema senza compromettere le prestazioni.
Un'azienda molto importante nell'ambito della cybersicurezza nazionale, nel nostro lavoro di sviluppo, ha implementato un agent dedicato al monitoraggio del sistema e alla mitigazione delle minacce. Questo agent raccoglie una vasta gamma di informazioni dal sistema al fine di ottenere una visione completa dello stato della macchina. Tuttavia, la raccolta di queste informazioni deve essere bilanciata per evitare un eccessivo overhead che potrebbe compromettere le prestazioni del sistema stesso.
La mia sfida principale è stata quella di selezionare le informazioni e gli eventi più rilevanti per l'identificazione di comportamenti sospetti, inviando e moti-

vando quanto scoperto all'azienda, senza dover apportare modifiche dirette al codice sorgente del kernel o caricare moduli aggiuntivi. Questa scelta è stata dettata dalla necessità di garantire la sicurezza del sistema senza comprometterne la distribuzione e l'installazione tramite pacchetti Linux standard. Per raggiungere questo obiettivo, è stata adottata una metodologia che si basa sull'utilizzo di tecnologie innovative come eBPF per la raccolta di informazioni a livello kernel space, consentendo di estendere le capacità del kernel in modo sicuro ed efficiente. Per la raccolta di informazioni a livello user space, sono state considerate diverse alternative, tra cui l'intercettazione delle system call e l'analisi dei file di log generati dalle applicazioni tramite gli strumenti auditd e inotify. Grazie a queste tecnologie e strumenti, la tesi si concentra su tre aspetti principali:

- Selezione delle informazioni e degli eventi rilevanti: il primo passo consiste nell'identificare e selezionare le informazioni e gli eventi più significativi per la rilevazione di comportamenti sospetti. Questo processo richiede un'approfondita comprensione delle macro famiglie di attacco e dei pattern di comportamento delle minacce, al fine di individuare quali informazioni sono cruciali per identificare potenziali rischi per la sicurezza del sistema.

- Implementazione di soluzioni di monitoraggio efficienti: una volta identificate le informazioni e gli eventi rilevanti, ci concentreremo sull'implementazione di soluzioni di monitoraggio efficienti, sia a livello kernel space che user space. Utilizzeremo tecnologie come eBPF per raccogliere informazioni a livello kernel space, garantendo la sicurezza del sistema senza modificare direttamente il codice sorgente del kernel. Per la raccolta di informazioni a livello user space, valuteremo diverse alternative, come l'intercettazione delle system call e l'analisi dei file di log generati dalle applicazioni attraverso strumenti come auditd e inotify.

- Valutazione delle prestazioni e dell'efficacia del sistema di monitoraggio: infine, valuteremo le prestazioni delle soluzioni implementate e ne

analizzeremo l'efficacia complessiva nel rilevare comportamenti sospetti e mitigare potenziali minacce alla sicurezza. Attraverso un'analisi approfondita dei dati raccolti e dei comportamenti identificati come sospetti, saremo in grado di valutare l'efficacia complessiva del sistema di monitoraggio e suggerire eventuali miglioramenti futuri.

L'obiettivo finale di questa ricerca è fornire una panoramica completa delle informazioni e degli eventi rilevanti per il monitoraggio del sistema, nonché di valutare le prestazioni delle soluzioni implementate a livello kernel space e user space. Attraverso un'analisi approfondita dei dati raccolti e dei comportamenti sospetti identificati, sarà possibile valutare l'efficacia complessiva del sistema di monitoraggio e suggerire eventuali miglioramenti futuri.
In sintesi, questa ricerca si propone di fornire una metodologia chiara e rigorosa per il monitoraggio dei sistemi Linux, offrendo una contribuzione significativa al campo della sicurezza informatica. Attraverso l'implementazione di soluzioni innovative e l'analisi approfondita dei dati raccolti, ci aspettiamo di fornire alle organizzazioni uno strumento prezioso per proteggere i propri asset digitali da potenziali minacce e garantire un funzionamento sicuro e affidabile dei loro sistemi informatici.
Il primo capitolo introduce il contesto e la motivazione alla base della ricerca, delineando le sfide e le necessità nel campo della sicurezza informatica e presentando gli obiettivi specifici della ricerca.
Il secondo capitolo offre una visione approfondita delle tecniche e delle tecnologie utilizzate nel monitoraggio dei sistemi Linux. Vengono presentati i concetti chiave relativi al monitoraggio in tempo reale, con un focus particolare sulle soluzioni implementate a livello kernel space e user space.
Il terzo capitolo descrive la metodologia adottata per affrontare i principali obiettivi della ricerca. Vengono illustrati i passaggi chiave coinvolti nella selezione delle informazioni e degli eventi rilevanti, nell'implementazione delle soluzioni di monitoraggio e nella valutazione delle prestazioni del sistema. Infine vengono analizzati i dati raccolti e discussi i comportamenti sospetti identificati, valutando l'efficacia complessiva del sistema di monitoraggio.

Il quarto capitolo presenta gli strumenti di test utilizzati per verificare la correttezza del codice.
Infine, il quinto capitolo offre una conclusione sintetica e una riflessione sulle implicazioni pratiche e le possibili direzioni future della ricerca.
Attraverso questa struttura ben definita, la tesi mira a fornire una guida completa e approfondita per coloro che sono interessati a comprendere e implementare soluzioni avanzate di monitoraggio dei sistemi Linux per garantire una maggiore sicurezza informatica.

Indice

Elenco delle figure

Capitolo 1

Background e contesto

1.1 Panoramica sul sistema Linux Ubuntu e sulle minacce informatiche

Nel contesto della cybersicurezza, è essenziale distinguere tra le varie distribuzioni software installate sulle nostre macchine, poiché ognuna potrebbe presentare proprietà e vulnerabilità uniche. Focalizzandoci sul sistema operativo Ubuntu, analizzeremo tre principali versioni: Canonical Ubuntu, Ubuntu Server e Ubuntu Core, quest'ultima specificamente progettata per l'Internet-of-Things (IoT).
Canonical Ubuntu, la versione desktop, è progettata per l'uso quotidiano da parte degli utenti finali, offrendo un'interfaccia grafica intuitiva e un'ampia gamma di applicazioni preinstallate. Le vulnerabilità tipiche includono quelle legate alle interfacce utente e alle applicazioni di terze parti.
Ubuntu Server, invece, è ottimizzato per l'ambiente server, fornendo prestazioni robuste e affidabili per gestire carichi di lavoro intensivi. Le sue vulnerabilità sono spesso legate a servizi di rete e configurazioni server specifiche, rendendo cruciale una gestione attenta della sicurezza del network e dell'hardware.
Infine, Ubuntu Core è una distribuzione minimalista ideata per l'IoT. Con un'architettura modulare e un forte focus sulla sicurezza, utilizza 'snap' per

gestire i pacchetti, riducendo il rischio di exploit grazie ad aggiornamenti automatici e al confinamento delle applicazioni. Le sfide di sicurezza per Ubuntu Core riguardano principalmente la protezione dei dispositivi con risorse limitate e l'integrità dei dati trasmessi.
In sintesi, mentre tutte queste distribuzioni condividono un nucleo linux comune, le loro differenze nell'uso previsto e nelle architetture sottostanti impongono approcci specifici per la gestione della sicurezza.

Le CVE

Le CVE, acronimo di Common Vulnerabilities and Exposures, rappresentano uno standard internazionale utilizzato per identificare e descrivere pubblicamente le vulnerabilità delle informazioni dei sistemi di sicurezza informatica. Questo sistema di catalogazione è stato istituito per fornire una chiara e concisa identificazione delle vulnerabilità dei sistemi informatici, facilitando così la comunicazione tra professionisti della sicurezza, fornitori di soluzioni e utenti finali. Di seguito sono riportati alcuni dettagli su cosa sono e a cosa servono le CVE:

1. **Identificazione univoca delle vulnerabilità**: Le CVE forniscono un'identificazione univoca per ogni vulnerabilità segnalata. Ogni CVE è assegnata a una specifica vulnerabilità e viene utilizzata come riferimento standard per indicare quella vulnerabilità in tutta la comunità della sicurezza informatica.

2. **Descrizione dettagliata delle vulnerabilità**: Ogni CVE include una descrizione dettagliata della vulnerabilità, compresi i suoi dettagli tecnici, le cause sottostanti, il potenziale impatto sulla sicurezza e le possibili contromisure o soluzioni.

3. **Facilitare la comunicazione**: Le CVE semplificano la comunicazione tra esperti di sicurezza, fornitori di software, organizzazioni di sicurezza e utenti finali. Utilizzando un identificatore standardizzato, è più facile

riferirsi a una specifica vulnerabilità durante le discussioni tecniche, la segnalazione dei problemi e la distribuzione delle correzioni.

4. **Prioritizzazione delle correzioni**: Le CVE consentono agli amministratori di sistema di valutare e prioritizzare le correzioni delle vulnerabilità in base alla loro gravità e al rischio che rappresentano per l'ambiente informatico. Le organizzazioni possono utilizzare le CVE per identificare le vulnerabilità più critiche e garantire che vengano affrontate tempestivamente.

5. **Monitoraggio delle minacce**: Le CVE forniscono un quadro chiaro delle vulnerabilità esistenti, consentendo alle organizzazioni di monitorare le minacce e adottare misure preventive per proteggere i propri sistemi e dati sensibili.

1.1.1 Canonical Ubuntu

Canonical Ubuntu è la distribuzione Linux più utilizzata, particolarmente apprezzata dai neofiti che si avvicinano al mondo Linux. Questo successo è dovuto principalmente alla sua natura user-friendly, che rende l'utilizzo del sistema operativo intuitivo anche per coloro che non hanno familiarità con il mondo della tecnologia. Basata su Debian, Ubuntu utilizza il desktop environment GNOME come interfaccia grafica predefinita, offrendo un'esperienza coerente e piacevole per gli utenti. Alcune CVE che questa distribuzione ha sofferto sono state:

- **CVE-2022-48703** [4]: Questa CVE riguarda una vulnerabilità nel kernel Linux relativa al driver termico `thermal/int340x_thermal`. In alcuni casi, quando il componente GDDV (Generic Distributed Data Vault) restituisce un pacchetto con un buffer di lunghezza zero, la funzione `kmemdup()` all'interno del kernel restituisce `ZERO_SIZE_PTR` (0x10). Di conseguenza, quando la funzione `data_vault_read()` tenta di accedere al valore 0x10 nel data_vault, si verifica un errore di

dereferenziazione di puntatore NULL, causando un malfunzionamento del kernel. Questa vulnerabilità è stata risolta utilizzando la funzione `ZERO_OR_NULL_PTR()` per controllare se il valore restituito da `kmemdup()` è `ZERO_SIZE_PTR` o NULL, mitigando così il problema di dereferenziazione di puntatore NULL .

- **CVE-2021-3560** [2]: Questa vulnerabilità in Polkit (PolicyKit) ha permesso ad attaccanti locali di ottenere privilegi elevati. La vulnerabilità è stata corretta nelle versioni di Ubuntu 20.04 LTS e superiori, ma ha avuto un impatto significativo su alcune versioni di Ubuntu più vecchie.

- **CVE-2023-50269 e CVE-2024-25617** [7] [9] : Queste vulnerabilità riguardano il proxy Squid, che è ampiamente utilizzato su Ubuntu per la gestione del traffico web. Avrebbero potuto essere sfruttate per causare un attacco denial of service, portando al crash di Squid. Le patch sono state rilasciate per Ubuntu 18.04 e versioni successive.

1.1.2 Ubuntu Server

Ubuntu Server è una solida piattaforma progettata per sviluppare e ospitare una vasta gamma di servizi, tra cui posta elettronica, servizi DNS, server web, database e server di file. Quando viene installato, il sistema è configurato in modo da non presentare alcun collegamento verso l'esterno e include solo il software essenziale per garantire la sicurezza del server. Grazie alla sua natura modulare e alle opzioni di configurazione flessibili, è possibile creare rapidamente un server LAMP (Linux, Apache, MySQL, PHP) funzionante in pochi minuti, direttamente durante il processo di installazione. Questa caratteristica, insieme ad altre funzionalità predefinite, può essere selezionata durante l'installazione, semplificando notevolmente il processo di configurazione e integrazione di ciascun componente del server LAMP. Questo approccio offre numerosi vantaggi: innanzitutto, le chiare scelte predefinite aumentano la sicurezza complessiva del server, poiché limitano le vulnerabi-

lità potenziali e riducono la superficie di attacco. In secondo luogo, il tempo di installazione è notevolmente ridotto consentendo di ottenere rapidamente un ambiente di sviluppo o di produzione funzionante. Inoltre, c'è un minore rischio di configurare erroneamente il server poiché le impostazioni predefinite sono progettate per garantire un funzionamento sicuro e affidabile. La gestione degli aggiornamenti è semplificata grazie alla funzionalità di aggiornamento automatico che cerca autonomamente gli aggiornamenti necessari e richiede all'utente di confermare l'installazione. Questo approccio riduce notevolmente il tempo e lo sforzo necessario per mantenere il server aggiornato e sicuro nel tempo. In definitiva, l'utilizzo di Ubuntu Server porta a un costo complessivo di gestione inferiore, poiché riduce i tempi di installazione e manutenzione, minimizza il rischio di errori di configurazione e semplifica la gestione degli aggiornamenti. Alcune vulnerabilità che questa distribuzione ha sofferto sono state:

- **CVE-2023-39325** [6] : Un client HTTP/2 malintenzionato che crea rapidamente richieste e le resetta immediatamente può causare un consumo eccessivo delle risorse del server. Sebbene il numero totale di richieste sia limitato dall'impostazione http2.Server.MaxConcurrentStreams, il reset di una richiesta in corso consente all'attaccante di creare una nuova richiesta mentre quella esistente è ancora in esecuzione. Con la correzione applicata, i server HTTP/2 ora limitano il numero di goroutine di gestione in esecuzione contemporaneamente al limite di concorrenza del flusso (MaxConcurrentStreams). Le nuove richieste che arrivano quando si è raggiunto il limite (il che può avvenire solo dopo che il client ha resettato una richiesta esistente e in corso) saranno messe in coda fino a quando un gestore non esce. Se la coda delle richieste cresce troppo, il server terminerà la connessione. Questo problema è stato risolto anche in golang.org/x/net/http2 per gli utenti che configurano manualmente HTTP/2. Il limite predefinito di concorrenza del flusso è di 250 flussi (richieste) per connessione HTTP/2. Questo valore può essere regolato utilizzando il pacchetto golang.org/x/net/http2.

- **CVE-2022-26352** [3] : Questa vulnerabilità ha riguardato il pacchetto *openssl*, con un impatto su vari servizi di rete, che potrebbero essere esposti a attacchi di tipo denial-of-service (DoS).

- **CVE-2021-3493** [1] : Questa vulnerabilità nel pacchetto snapd permetteva agli attaccanti di sfruttare un errore di validazione degli input, potenzialmente consentendo l'esecuzione di un codice arbitrario.

1.1.3 Ubuntu Core

Ubuntu Core rappresenta una versione altamente specializzata, progettata per soddisfare le esigenze uniche dell'Internet of Things (IoT) e di altre implementazioni embedded. Ubuntu Core offre una piattaforma leggera e altamente ottimizzata che può essere eseguita su una vasta gamma di dispositivi, inclusi robot e gateway, consentendo loro di svolgere una serie di compiti specifici in modo affidabile e sicuro; a differenza delle versioni Canonical Ubuntu e Ubuntu Server, Ubuntu Core è progettato per operare su una scala molto più piccola, ma con la stessa robustezza e stabilità dei sistemi operativi citati precedentemente. Condivide il kernel, le librerie e i sistemi software con la controparte standard di Ubuntu, ma è ottimizzato per l'esecuzione su dispositivi embedded e altre risorse limitate.
Una delle caratteristiche distintive di Ubuntu Core è la sua architettura basata su app isolate, questo significa che le applicazioni operano in un ambiente completamente isolato l'una dall'altra, garantendo che eventuali problemi o vulnerabilità in un'app non influenzino le altre e, inoltre, l'accesso alle risorse di sistema è strettamente controllato attraverso permessi espliciti, garantendo un ambiente sicuro e affidabile.
Gli aggiornamenti in Ubuntu Core sono gestiti in modo che siano resistenti a condizioni di hardware o di rete impreviste, consentendo al sistema di rimanere stabile e affidabile anche in situazioni critiche. I pacchetti di aggiornamento sono immutabili e dotati di una firma digitale persistente, garantendo l'integrità e la sicurezza del sistema. Un altro elemento chiave di Ubuntu

Core è l'uso dei pacchetti snap, che sono confinati in una sandbox di sicurezza ristretta, questo significa che le applicazioni snap non hanno accesso alle risorse di sistema al di fuori del loro ambiente di esecuzione, garantendo un ambiente sicuro e protetto per le applicazioni. Le ultime vulnerabilità che questa distribuzione ha sofferto sono state:

- **CVE-2024-29040** [10] : Questa vulnerabilità nel TPM2 (Trusted Platform Module) Software Stack potrebbe permettere l'esecuzione di codice arbitrario. Ha impatti significativi sulla sicurezza per i sistemi che utilizzano TPM2 per operazioni critiche, incluso Ubuntu Core. Questo tipo di problema può compromettere il sistema operativo nei dispositivi embedded, poiché il TPM viene utilizzato per l'integrità e la verifica della sicurezza di questi sistemi.

- **CVE-2023-51385** [8] : Colpisce OpenSSH e riguarda un difetto di autenticazione che permette agli attaccanti di bypassare certe protezioni. Ubuntu Core utilizza OpenSSH per il remote management sicuro, quindi questa vulnerabilità può rappresentare un rischio in ambienti embedded, specialmente per dispositivi IoT che si affidano a connessioni remote per il controllo.

- **CVE-2023-3446** [5] : Una vulnerabilità in OpenSSL che può portare a errori di gestione della memoria, aumentando il rischio di attacchi DoS o di esecuzione di codice arbitrario. OpenSSL è ampiamente utilizzato per cifrare le comunicazioni nei sistemi Ubuntu, inclusi quelli embedded come Ubuntu Core, esponendo potenzialmente il sistema a compromissioni durante le trasmissioni dati sensibili.

1.1.4 Cosa e quali sono i malware informatici

Per malware si intendono tutti quei programmi che, una volta penetrati in un sistema informatico, cercano di disturbare le azioni svolte dagli utenti o di sottrarne informazioni importanti. A seconda della modalità in cui

entrano in un sistema informatico e di come riescano quindi a diffondersi è possibile classificarli.

- **Virus**: I virus informatici rappresentano una classe insidiosa di malware che, una volta attivati, si insinuano nel sistema operativo e cercano di diffondersi attraverso la manipolazione dell'ambiente circostante. Agiscono modificando il codice di altri software e file, contaminandoli con la propria logica dannosa. Il loro obiettivo primario è propagarsi attraverso il sistema, sfruttando vulnerabilità e lacune nella sicurezza. Quando un file infetto da virus viene trasferito o copiato su un altro sistema, il virus viaggia insieme a esso, pronto a infiltrarsi nell'ambiente del nuovo ospite. La natura ricorsiva dei virus informatici li rende particolarmente pericolosi, in quanto continuano a cercare nuove vittime da infettare, creando così una spirale di contagio che può diffondersi rapidamente attraverso reti e sistemi interconnessi.

- **Worms**: I worm informatici rappresentano una categoria sofisticata di malware che agisce senza la necessità di legarsi ad altri software o file esistenti. A differenza dei virus, i worm attaccano direttamente il sistema operativo, sfruttando vulnerabilità e falle di sicurezza per propagarsi e replicarsi attraverso Internet. Questi agenti malevoli sono progettati per diffondersi in modo autonomo e aggressivo, sfruttando le connessioni di rete e i protocolli di comunicazione per infettare altri dispositivi e sistemi connessi. Utilizzano metodi sofisticati di ingegneria sociale per attirare gli utenti ad aprire file o cliccare su link dannosi, facilitando così la loro diffusione. Il modus operandi dei worm li rende particolarmente pericolosi, poiché possono diffondersi rapidamente e incontrollabilmente attraverso reti di computer interconnessi. Una volta infiltrati in un sistema, possono causare danni significativi alla sicurezza e alla stabilità del sistema, compromettendo la privacy e la sicurezza dei dati degli utenti.

- **Trojan Horse**: I Trojan Horse, noti anche come trojan, rappresentano una delle forme più subdole e diffuse di malware. A differenza dei virus e dei worm, i trojan non cercano di replicarsi autonomamente, ma vengono nascosti all'interno di software apparentemente normali o innocui. Gli utenti spesso cadono vittima di questi attacchi mentre navigano su Internet alla ricerca di versioni gratuite o piratate di software a pagamento. Una volta che un trojan è stato installato sul sistema dell'utente, può iniziare a svolgere una serie di azioni dannose, che possono includere il furto di informazioni sensibili, il danneggiamento o il blocco del sistema, il controllo remoto del computer e molto altro ancora. Questi malware sono progettati per agire in modo silenzioso e furtivo, spesso sfuggendo alla rilevazione da parte degli strumenti di sicurezza tradizionali. Nonostante non abbiano la capacità di replicarsi autonomamente, i trojan rappresentano comunque una minaccia significativa per la sicurezza informatica, costituendo circa il 70% di tutti i malware conosciuti. Questo è dovuto alla loro capacità di nascondersi all'interno di software apparentemente legittimi e alla loro capacità di ingannare gli utenti, inducendoli a installarli volontariamente. La diffusione dei trojan avviene principalmente attraverso Internet, ma possono anche essere distribuiti attraverso supporti di massa come chiavette USB o dischi rigidi esterni.

- **Backdoor**: Le backdoor sono una tipologia di software progettata per bypassare le normali procedure di autenticazione di un sistema informatico, consentendo l'accesso non autorizzato a risorse o funzionalità normalmente protette. Questi strumenti possono essere utilizzati sia in modo lecito, ad esempio per scopi di manutenzione o recupero dati, sia in modo illegittimo da parte di attaccanti malintenzionati. Un esempio comune di backdoor è rappresentato dai software di accesso remoto, che permettono agli amministratori di sistema di controllare e gestire un computer da remoto. Tuttavia, se queste stesse funzionalità sono sfruttate da un attaccante per ottenere un accesso non autorizzato al

sistema, si configura un utilizzo illecito della backdoor. Un esempio pratico di backdoor è rappresentato dal software TeamViewer, ampiamente utilizzato per il controllo remoto dei computer. Se un utente malintenzionato riesce ad installare segretamente il software TeamViewer su un computer, può ottenere un accesso non autorizzato al sistema, bypassando le normali procedure di autenticazione.

- **Spyware**: Lo spyware è una categoria di software malevolo progettato per raccogliere informazioni sensibili e riservate sui dispositivi e sui comportamenti degli utenti, senza il loro consenso o conoscenza. Questi programmi operano in modo invisibile, monitorando attivamente le attività dell'utente e raccogliendo una vasta gamma di dati, tra cui informazioni personali, abitudini di navigazione, password, informazioni finanziarie e molto altro ancora. Una volta che le informazioni sono state raccolte, lo spyware le trasferisce silenziosamente a un destinatario remoto, che può essere un hacker o un criminale informatico. Questi dati possono quindi essere utilizzati per scopi illeciti, come il furto di identità, la frode finanziaria, il phishing o altri tipi di truffe. In alcuni casi, gli attaccanti possono anche sfruttare le informazioni raccolte per condurre campagne di ingegneria sociale, manipolando gli utenti e inducendoli a compiere azioni dannose o a rivelare ulteriori informazioni sensibili. Un esempio pratico di spyware è rappresentato dai keylogger, che registrano segretamente tutte le tastiere battute dall'utente e inviano i dati a un server remoto. Questi programmi possono essere installati su computer infetti senza il consenso dell'utente e possono raccogliere informazioni sensibili come password, numeri di carta di credito e altre informazioni personali.

- **Rootkit**: I rootkit sono una categoria sofisticata di malware progettata per ottenere e mantenere l'accesso non autorizzato a un sistema informatico, operando in modo furtivo e nascosto all'interno dell'infrastruttura del sistema operativo. Questi software malevoli sono spesso

utilizzati da attaccanti per mascherare le loro attività e mantenere un controllo discreto e persistente sul sistema target. I rootkit generalmente operano in coppia, con uno dei componenti che svolge la funzione di un normale software o utilità di sistema, mentre l'altro esegue azioni malevole in modo silenzioso e occulto. Ad esempio, un componente del rootkit potrebbe mimetizzarsi come un processo di sistema legittimo, come /bin/login su sistemi Unix, mentre il secondo componente potrebbe essere utilizzato per eseguire attività dannose, come il monitoraggio delle attività dell'utente, la raccolta di informazioni sensibili o la creazione di backdoor per accesso remoto. Un esempio pratico di rootkit è rappresentato dal rootkit Sony BMG, che è stato incluso nei CD musicali distribuiti da Sony BMG Entertainment nel 2005. Questo rootkit è stato progettato per proteggere i CD da copie non autorizzate, ma ha provocato gravi preoccupazioni per la sicurezza informatica in quanto si è dimostrato difficile da rilevare e rimuovere. Inoltre, il rootkit ha aperto una backdoor sul sistema dei computer che lo hanno eseguito, consentendo a potenziali attaccanti di ottenere un accesso non autorizzato.

- **Rabbit**: il Rabbit è un tipo di malware noto per la sua capacità di replicarsi estremamente velocemente all'interno di un sistema informatico, fino a utilizzare tutte le risorse disponibili. Questo comportamento aggressivo e invasivo può causare gravi danni al sistema, rendendolo instabile, rallentandone le prestazioni e compromettendo la sua funzionalità complessiva. Il nome "Rabbit" deriva dalla sua tendenza a moltiplicarsi rapidamente, come i conigli, occupando spazio e risorse fino a esaurirli completamente. Questo comportamento lo rende particolarmente dannoso e difficile da contenere, poiché può saturare rapidamente la capacità di elaborazione del sistema, rendendolo inutilizzabile. Una volta infiltrato all'interno del sistema, il Rabbit inizia a replicarsi e diffondersi attraverso la rete o altri mezzi di trasmissione, sfruttando vulnerabilità e falle di sicurezza per propagarsi in modo rapido e incon-

trollato. Questo processo di replicazione continua fino a quando tutte le risorse del sistema non sono esaurite, causando danni irreversibili e compromettendo la sicurezza e la stabilità del sistema. Un esempio pratico di Rabbit è rappresentato dal malware Slammer, che ha colpito severamente i sistemi informatici nel 2003. Slammer era un worm progettato per sfruttare una vulnerabilità nel software Microsoft SQL Server, e si è diffuso rapidamente attraverso Internet, saturando le reti e compromettendo migliaia di server in tutto il mondo in poche ore.

- **Keylogger**: I keylogger sono software progettati per registrare segretamente tutte le attività di input digitale effettuate su un sistema operativo, inclusa la digitazione di tasti, il copia e incolla di testo e altre operazioni di input. Questi programmi operano in modo silenzioso e invisibile agli utenti, consentendo agli attaccanti di monitorare e registrare tutte le attività dell'utente senza il loro consenso. La caratteristica distintiva dei keylogger è la loro capacità di catturare in modo discreto e persistente tutte le informazioni digitate dall'utente, comprese le password, i numeri di carta di credito, le informazioni personali e altre informazioni sensibili. Queste informazioni vengono quindi memorizzate localmente sul sistema o inviate all'attaccante attraverso una connessione Internet, consentendo loro di accedere e utilizzare queste informazioni per scopi malevoli. I keylogger possono essere installati su un sistema in vari modi, inclusa l'inclusione nei software scaricati, l'installazione da parte di un utente malintenzionato con accesso fisico al sistema o l'integrazione in dispositivi hardware, come chiavette USB, che vengono poi posizionati nei computer di utenti ignari. Una volta attivati, i keylogger lavorano in modo furtivo, registrando tutte le attività di input dell'utente senza la loro conoscenza o consenso. Questi programmi sono spesso utilizzati per scopi fraudolenti, come il furto di credenziali di accesso a siti web, informazioni finanziarie o altri dati sensibili. Possono anche essere utilizzati per monitorare le attività degli utenti, raccogliere informazioni personali o commerciali e

compromettere la privacy e la sicurezza delle informazioni.

- **Rogue antispyware**: I rogue antispyware, conosciuti anche come rogue o fraud tool, sono una categoria di malware che si mascherano da software antivirus o antispyware legittimi al fine di ingannare gli utenti e indurli ad acquistare la versione completa del software, spesso a un costo esorbitante. Questi programmi malevoli sono progettati per apparire come soluzioni di sicurezza affidabili, offrendo funzionalità di scansione e rimozione dei malware apparentemente efficaci. La caratteristica distintiva dei rogue antispyware è la loro capacità di simulare il comportamento di un software di sicurezza legittimo, conducendo scansioni del sistema che producono risultati falsati. Durante queste scansioni, i rogue antispyware possono generare rapporti ingannevoli che segnalano la presenza di numerose minacce e infezioni sul sistema, anche se tali minacce potrebbero non esistere affatto o essere completamente inoffensive. L'obiettivo finale dei rogue antispyware è quello di convincere gli utenti a acquistare la versione completa del software, promettendo di risolvere i problemi segnalati durante la scansione. Un esempio pratico di rogue antispyware è rappresentato da programmi che si presentano come antivirus affidabili e offrono scansioni del sistema gratuite. Durante la scansione, questi programmi possono segnalare la presenza di numerosi malware immaginari, convincendo gli utenti a pagare per una versione completa del software per rimuoverli.

- **Ransomware**: Il ransomware è una forma sofisticata di malware progettata per cifrare tutti i dati personali e crittografarli sul disco rigido di un computer infetto, rendendoli inaccessibili all'utente. Questo tipo di malware è noto per la sua capacità di causare danni gravi e irreversibili, poiché una volta che i dati sono stati cifrati, è estremamente difficile, se non impossibile, ripristinarli senza la chiave di decrittazione corretta. La caratteristica distintiva del ransomware è la sua modalità di operare: una volta che il malware infetta il sistema, cifra tutti i file

crittografandoli con un algoritmo di crittografia avanzato. Successivamente, il ransomware visualizza un messaggio di richiesta di riscatto all'utente, informandolo che per ottenere la chiave di decrittazione e ripristinare l'accesso ai propri dati, è necessario pagare un "riscatto" tramite un sistema di pagamento non rintracciabile, come Bitcoin o altre criptovalute. Questo tipo di estorsione digitale è estremamente lucrativo per gli autori del ransomware, poiché il pagamento del riscatto può variare da centinaia a migliaia di dollari, a seconda della gravità dell'infezione e del valore dei dati crittografati per l'utente. Tuttavia, non c'è alcuna garanzia che il pagamento del riscatto porterà al ripristino dei dati, e spesso gli utenti possono rimanere senza i propri file anche dopo aver pagato il riscatto richiesto. Un esempio pratico di ransomware è rappresentato da varianti come WannaCry e Petya, che hanno colpito migliaia di utenti e organizzazioni in tutto il mondo, causando danni finanziari e compromettendo la sicurezza dei dati. Questi attacchi dimostrano la gravità della minaccia rappresentata dal ransomware e l'importanza di adottare misure di sicurezza informatica avanzate per proteggere i dati da questo tipo di attacchi.

- **Bomba logica o a tempo**: Una bomba logica o a tempo è un tipo insidioso di malware progettato per attivarsi solo quando si verificano determinate condizioni prestabilite, come il raggiungimento di un certo numero di record in un database, la cancellazione di un file specifico o lo scadere di un timer temporizzato. La caratteristica distintiva delle bombe logiche o a tempo è la loro natura silenziosa e stealth: una volta introdotte nel sistema, queste minacce rimangono latenti e non rilevabili fino al verificarsi delle condizioni prestabilite. Questo le rende particolarmente pericolose, poiché gli utenti non sono consapevoli della loro presenza e possono essere colpiti senza preavviso quando le azioni malevole vengono innescate. Una volta che le condizioni prestabilite vengono soddisfatte, la bomba logica o a tempo mette in atto le sue funzioni malevole, che possono includere la cancellazione di file critici,

la corruzione dei dati, il blocco del sistema o altre azioni dannose. Questo tipo di malware è progettato per causare danni significativi e destabilizzare il sistema dell'utente senza essere facilmente individuato o neutralizzato.

- **Bomba a decompressione**: Una bomba a decompressione è una forma avanzata di malware progettata per sfruttare la vulnerabilità dei programmi di decompressione degli archivi per eseguire attacchi contro un sistema informatico. Questo tipo di malware prende di mira i software che gestiscono la decompressione di archivi, come ad esempio file ZIP o RAR, e sfrutta le loro caratteristiche per rendere inutilizzabile il sistema su cui viene aperto l'archivio dannoso. La caratteristica distintiva delle bombe a decompressione è la loro struttura intricata e ingannevole: questi archivi malevoli sono progettati in modo che, una volta aperti e decompressi, contengano file all'interno di file, ricorsivamente, in un ciclo infinito. In pratica, questo significa che l'archivio può sembrare piccolo in dimensioni, ma una volta estratto, i file risultanti occupano molto più spazio su disco, sovraccaricando rapidamente il sistema e saturando le risorse disponibili. L'obiettivo finale di una bomba a decompressione è quello di sovraccaricare il sistema e renderlo inutilizzabile o estremamente lento, impedendo agli utenti di accedere ai propri dati o di utilizzare il computer in modo efficace. Questo tipo di attacco può avere gravi conseguenze, inclusi danni irreversibili ai file di sistema, perdita di dati critici e interruzioni delle attività aziendali.

1.1.5 Alcune tipologie di attacco informatico

- **SQL Injection:** l'SQL Injection è una delle vulnerabilità più comuni e dannose nei sistemi di gestione dei database relazionali (RDBMS). Consiste nell'inserimento di istruzioni SQL malevole all'interno delle query SQL inviate alle applicazioni web tramite input utente, consentendo agli attaccanti di manipolare il comportamento del database.

Tuttavia, se un'applicazione web non valida o filtra in modo inadeguato i dati forniti dagli utenti prima di incorporarli nelle query SQL, gli attaccanti possono sfruttare questa debolezza inserendo istruzioni SQL malevole. I potenziali danni causati dall'SQL Injection possono essere estesi e gravi. Ecco alcuni dei rischi associati:

  - Accesso non autorizzato ai dati sensibili: gli attaccanti possono utilizzare l'SQL Injection per accedere a dati sensibili memorizzati nel database, come informazioni personali, credenziali utente, dettagli finanziari e altro ancora.
  - Modifica o eliminazione dei dati: una volta ottenuto l'accesso al database, gli attaccanti possono modificare o eliminare dati sensibili, compromettendo l'integrità e la coerenza dei dati.
  - Esecuzione di comandi SQL dannosi: gli attaccanti possono eseguire comandi SQL dannosi per compromettere il sistema, ad esempio eliminando o corrompendo tabelle, modificando le impostazioni del database o eseguendo altre azioni dannose.
  - Denial of Service (DoS): gli attaccanti possono utilizzare l'SQL Injection per eseguire query SQL complesse o inefficienti che consumano risorse del database, portando a un'interruzione del servizio per gli utenti legittimi.

- **DoS:** un attacco DoS, acronimo di Denial of Service, è un tipo di attacco informatico in cui l'attaccante tenta di rendere un server, un servizio o una rete inaccessibile agli utenti legittimi. Questo viene realizzato inviando una quantità eccessiva di richieste o pacchetti al server, con l'intento di sovraccaricare le risorse di sistema o di rete, rendendo il server incapace di rispondere a richieste legittime.

  L'attacco DoS differisce dagli attacchi DDoS (Distributed Denial of Service), in quanto proviene solitamente da una singola fonte, piuttosto che da una rete distribuita di dispositivi compromessi. Tuttavia,

l'obiettivo rimane lo stesso: interrompere l'accesso a un servizio online o ridurre drasticamente le sue prestazioni.

  - **Flooding del traffico:** uno degli approcci più comuni per un attacco DoS è il flooding del traffico, in cui l'attaccante invia una grande quantità di dati o richieste a un server in un breve lasso di tempo. Questo può causare il consumo eccessivo della larghezza di banda disponibile o delle risorse di calcolo del server, portandolo all'inaccessibilità.
  - **Attacchi a livello di protocollo:** in alcuni casi, gli attacchi DoS sfruttano vulnerabilità nei protocolli di comunicazione, come TCP o ICMP. Ad esempio, un attacco SYN flood può tentare di esaurire la tabella delle connessioni del server, sfruttando il processo di handshake TCP per inviare richieste di connessione incomplete.
  - **Attacchi a livello applicativo:** questi attacchi mirano specificamente alle risorse dell'applicazione. Un esempio comune è il bombardamento di richieste HTTP, dove l'attaccante invia migliaia di richieste HTTP, sovraccaricando il server web e costringendolo a esaurire le risorse necessarie per rispondere ai normali utenti.

- **DDoS:** un attacco DDoS, acronimo di Distributed Denial of Service, è un tipo di attacco informatico in cui un gran numero di dispositivi o sistemi, noti come botnet, inviano simultaneamente una vasta quantità di richieste ad un server o un servizio online. L'obiettivo principale di un attacco DDoS è quello di sopraffare il server bersaglio, rendendolo inaccessibile agli utenti legittimi. Queste richieste possono essere di diversi tipi, come richieste HTTP, ping o altre richieste di rete. Il server, non essendo in grado di distinguere tra richieste legittime e illegittime, tenta di processarle tutte. Tuttavia, a causa del volume massiccio di richieste in arrivo, il server diventa rapidamente sovraccarico e non

può più rispondere in modo efficace alle richieste legittime degli utenti. Esistono diversi tipi di attacchi DDoS, tra cui:

  - Attacchi di amplificazione: in cui gli hacker sfruttano server vulnerabili per amplificare il traffico inviato al server bersaglio, rendendo l'attacco più potente.
  - Attacchi di sovraccarico della larghezza di banda: in cui gli hacker inviano un'enorme quantità di dati al server, consumando tutta la larghezza di banda disponibile e rendendo il server inaccessibile.
  - Attacchi di sovraccarico delle risorse: in cui gli hacker inviano una grande quantità di richieste complesse o dispendiose in termini di risorse al server, facendo sì che il server esaurisca le risorse di elaborazione disponibili.

- **XSS (Cross-Site Scripting):** l'attacco XSS, o Cross-Site Scripting, è una delle vulnerabilità più comuni e pericolose che possono colpire le applicazioni web. Questo tipo di attacco consente agli hacker di inserire script malevoli all'interno delle pagine web visualizzate dagli utenti, sfruttando le vulnerabilità presenti nei siti web o nelle applicazioni web. Per comprendere meglio come funziona un attacco XSS, immagina di visitare un sito web che consente agli utenti di inserire commenti o messaggi. Se il sito non valida o filtra in modo inadeguato i dati inseriti dagli utenti prima di renderli sulla pagina web, un aggressore potrebbe inserire script JavaScript malevoli all'interno del campo di testo del commento. Quando un altro utente visita la pagina e visualizza il commento compromesso, il browser eseguirà automaticamente lo script, senza che l'utente si accorga di nulla. Le conseguenze di un attacco XSS possono essere molto gravi e variegate:

  - Furto di dati sensibili: Gli attaccanti possono utilizzare gli script XSS per rubare dati sensibili degli utenti, come username, password, informazioni di pagamento e altro ancora. Ad esempio,

uno script XSS potrebbe inviare i dati compilati in un modulo di login a un server controllato dall'attaccante.

- Phishing: Gli attaccanti possono creare pagine web fasulle, che sembrano autentiche, e utilizzare gli script XSS per reindirizzare gli utenti verso queste pagine. Questo è particolarmente pericoloso se la pagina web falsa imita una pagina di login di un sito web legittimo, poiché gli utenti potrebbero essere indotti a inserire le proprie credenziali, che vengono poi rubate dagli aggressori.
- Defacement: Gli attaccanti possono utilizzare gli script XSS per modificare il contenuto di un sito web, sostituendo pagine o aggiungendo contenuti offensivi o dannosi. Questo danneggia la reputazione del sito e può causare gravi danni all'immagine dell'azienda o dell'organizzazione proprietaria del sito.
- Esecuzione di azioni non autorizzate: Gli script XSS possono anche essere utilizzati per eseguire azioni non autorizzate a nome dell'utente, come inviare messaggi, effettuare acquisti o eseguire transazioni finanziarie.

- **Command Injection:** l'attacco di Command Injection è una delle minacce più pericolose per la sicurezza dei sistemi informatici, particolarmente diffusa nelle applicazioni web e nei sistemi che interagiscono con l'utente tramite comandi esterni, come le interfacce a riga di comando (CLI). In questo tipo di attacco, gli aggressori sfruttano le vulnerabilità presenti in un'applicazione che accetta input dall'utente e li elabora come comandi eseguibili dal sistema operativo. Questi input non vengono validati o filtrati correttamente dall'applicazione, consentendo agli aggressori di inserire comandi dannosi all'interno dei campi di input. Una delle azioni più comuni che gli hacker fanno assieme alla command injection è quella di eseguire una reverse shell: la Reverse Shell è una tecnica utilizzata per eseguire comandi appositamente progettati per avviare un programma che si mette in ascolto su una

porta specifica del server. Una volta che il server è stato compromesso attraverso l'attacco di Command Injection con reverse shell, gli aggressori possono quindi connettersi al server da remoto tramite un client. Questa connessione consente loro di ottenere un accesso interattivo al sistema, simile a quello che avrebbero se fossero connessi direttamente al server tramite un terminale. Una volta stabilita la connessione, gli aggressori possono eseguire ulteriori comandi sul sistema, trasferire file da e verso il server, modificare le impostazioni del sistema e persino installare altri software dannosi per mantenere l'accesso persistente al sistema. Inoltre, possono utilizzare la Reverse Shell per nascondere le loro attività dannose, rendendo più difficile per gli amministratori di sistema rilevare e rispondere all'attacco.

- **Attacco di Man-in-the-Middle (MitM):** un attacco Man-in-the-Middle (MitM) è una tecnica utilizzata dagli aggressori per intercettare e manipolare la comunicazione tra due parti che credono di comunicare direttamente tra loro. Questo tipo di attacco è particolarmente dannoso perché consente agli aggressori di intercettare dati sensibili, come informazioni di login, dati finanziari o messaggi di testo, e persino di modificare o iniettare nuovi dati nella comunicazione senza che le parti coinvolte se ne accorgano. Per eseguire un attacco MitM, gli aggressori devono posizionarsi strategicamente tra la vittima e il destinatario della comunicazione. Ciò può essere fatto in diversi modi, ad esempio attraverso l'intercettazione del traffico di rete in un punto di accesso non protetto, l'uso di un malware per intercettare il traffico su un dispositivo compromesso o l'accesso fisico a un dispositivo di rete per manipolare il flusso di dati. Una volta posizionati tra le due parti, gli aggressori possono iniziare a intercettare e manipolare il traffico di rete in diversi modi:

  - **Interrogazione passiva**: gli hacker possono semplicemente intercettare il traffico di rete senza modificarlo. In questo mo-

do possono raccogliere informazioni sensibili, come password o informazioni di login, trasmesse in chiaro attraverso la rete.

- **Interrogazione attiva**: gli hacker possono modificare attivamente il traffico di rete per iniettare, alterare o eliminare dati. Ad esempio, possono sostituire una pagina web autentica con una versione contraffatta contenente malware o phishing, possono modificare i dettagli di pagamento in una transazione finanziaria o possono intercettare e modificare messaggi di posta elettronica.
- **Denial of Service (DoS)**: gli hacker possono interrompere la comunicazione tra le due parti causando un'interruzione del servizio.

1.2 Tecnologie e strumenti utilizzati per la difesa dei nostri sistemi

1.2.1 eBPF

eBPF, o Extended Berkeley Packet Filter, è un framework all'interno del kernel Linux che consente agli sviluppatori di scrivere e caricare programmi che vengono eseguiti in modo sicuro all'interno del kernel stesso. Questi programmi, noti come programmi BPF, possono essere utilizzati per filtrare, analizzare e modificare il flusso dei pacchetti di rete, oltre che per effettuare monitoraggio e tracciamento delle risorse di sistema.
Una delle caratteristiche distintive di eBPF è la sua capacità di eseguire codice sicuro all'interno del kernel senza compromettere la stabilità o la sicurezza del sistema. Questo è possibile grazie a un rigoroso modello di sicurezza che limita il tempo di esecuzione e le risorse utilizzate dai programmi BPF, evitando così il rischio di crash del kernel o di exploit di sicurezza.
Grazie alla sua flessibilità e alle prestazioni elevate, eBPF è diventato uno strumento fondamentale per lo sviluppo di applicazioni di rete, sicurezza, monitoraggio delle prestazioni e altro ancora. È ampiamente utilizzato in

una varietà di scenari, tra cui firewall di rete, sistemi di rilevamento delle intrusioni, analisi del traffico di rete e ottimizzazione delle prestazioni del kernel [21].
Negli ultimi anni, l'ecosistema eBPF ha visto una rapida crescita, con un crescente numero di strumenti, librerie e framework che supportano lo sviluppo e la gestione dei programmi BPF. Ciò include strumenti come BCC (BPF Compiler Collection), che semplifica lo sviluppo e il debug dei programmi BPF, e progetti come Cilium e XDP (eXpress Data Path), che sfruttano le capacità di eBPF per fornire funzionalità avanzate di rete e sicurezza

1.2.2 Auditd

Auditd è una componente essenziale per la sicurezza e la gestione dei sistemi Linux, fornendo un potente strumento per il monitoraggio e la registrazione delle attività di sistema.
Auditd, o il demone Linux Audit, è un sistema di auditing a livello user che, nonostante non sia profondo come eBPF (Kernel Level) riesce a tracciare una vasta gamma di eventi, dall'accesso al file system all'esecuzione di processi, dalle connessioni di rete alle azioni degli utenti. La sua modalità di funzionamento si basa sull'intercettazione delle chiamate di sistema, registrandole in un file di log per un'analisi successiva.
La configurazione di Auditd avviene attraverso un insieme di regole definite dall'utente, che specificano quali eventi devono essere registrati, questo offre una flessibilità notevole, consentendo agli amministratori di sistema di adattare il monitoraggio alle esigenze specifiche del loro ambiente. Una volta configurato, Auditd lavora in background, registrando gli eventi in un file di log che può essere esaminato con il comando ausearch [13].
I vantaggi di utilizzare Auditd sono molteplici: innanzitutto, contribuisce in modo significativo al miglioramento della sicurezza dei sistemi Linux, consentendo il rilevamento tempestivo e la risposta alle violazioni della sicurezza, inoltre, facilita la risoluzione dei problemi di sistema, consentendo agli amministratori di identificare rapidamente la causa di eventuali anomalie o

malfunzionamenti.
Tuttavia, non possiamo ignorare i limiti delle regole di Auditd esistenti: la mancanza di specificità in alcune regole può portare a un eccessivo numero di falsi positivi, rendendo più difficile l'individuazione di attività dannose. Inoltre, le regole di Auditd potrebbero non essere in grado di rilevare attacchi complessi che prevedono più passaggi o che utilizzano tecniche sofisticate per eludere il rilevamento, infine, la limitata visibilità sulle applicazioni e sull'attività di rete potrebbe rappresentare una lacuna nella capacità di rilevamento di determinati tipi di minacce.

1.2.3 Inotify

Inotify è un'API che fornisce un meccanismo per monitorare le modifiche al file system in Linux. Consente ai programmi di ricevere notifiche quando file o directory vengono creati, modificati o eliminati [17]. Questo può essere utile per una varietà di scopi, come:

- Rilevare modifiche ai file di configurazione
- Tracciare il download dei file
- Monitorare l'accesso ai file da parte di utenti o processi specifici

Inotify si basa su un sistema di "watch", quando un programma apre un'istanza di inotify, può specificare una lista di directory da monitorare, il kernel monitorerà quindi queste directory e invierà notifiche al programma ogni volta che viene rilevata una modifica.
Le notifiche vengono inviate utilizzando un descrittore di file: quando viene rilevata una modifica, il kernel scriverà un messaggio nel descrittore di file; il programma può quindi leggere il messaggio per determinare quale tipo di modifica è avvenuta e quale file o directory è stato interessato [17].
Tuttavia, presenta anche alcuni problemi:

- Può essere inefficiente monitorare un grande numero di directory. Inotify utilizza un meccanismo di polling per monitorare le directory, il che significa che deve controllare ogni directory per le modifiche a intervalli regolari. Questo può essere inefficiente se un programma sta monitorando un grande numero di directory, poiché può consumare una quantità significativa di tempo CPU.

- Può essere difficile gestire un grande numero di eventi simultanei. Inotify può gestire solo un numero limitato di eventi simultanei. Se si verificano un grande numero di modifiche contemporaneamente, alcuni eventi potrebbero essere persi.

- Può essere difficile gestire eventi che si verificano rapidamente. Inotify potrebbe non essere in grado di gestire eventi che si verificano rapidamente. Questo può essere un problema se un programma sta cercando di tracciare le modifiche a un file che viene modificato frequentemente.

Capitolo 2

Metodologie di analisi e sviluppo

2.1 Raccomandazioni per l'ambiente di test

L'esecuzione di codice eBPF richiede un ambiente di sviluppo configurato correttamente per garantire il corretto funzionamento dei programmi basati su BPF (Berkeley Packet Filter). Di seguito sono descritte le principali componenti dell'ambiente e le relative configurazioni necessarie per supportare l'esecuzione di codice eBPF [11].

2.1.1 Requisiti di Sistema

- **Kernel Linux 4.4 o successivo**: eBPF è integrato nel kernel Linux e le versioni più recenti del kernel forniscono un supporto più completo e stabile per le funzionalità eBPF [11].

- **Compilatore LLVM/Clang**: il codice eBPF viene compilato utilizzando LLVM/Clang, che genera il bytecode compatibile con il kernel Linux [21].

- **BCC (BPF Compiler Collection)**: è uno strumento essenziale per lo sviluppo e il debug di programmi eBPF, fornendo un set di librerie e strumenti per facilitare la scrittura del codice [11].

2.1.2 Configurazioni del Kernel

Oltre a un kernel compatibile, è necessario configurare alcune opzioni del kernel per abilitare correttamente il supporto a eBPF. Le principali configurazioni includono:

- `CONFIG_BPF=y`: questa opzione abilita il sottosistema BPF nel kernel.
- `CONFIG_BPF_SYSCALL=y`: consente l'uso delle syscall `bpf()` per la gestione di programmi BPF.
- `CONFIG_NETFILTER_XDP=y`: questa opzione è necessaria per l'uso di XDP (eXpress Data Path), un'estensione eBPF per il filtraggio dei pacchetti a livello di rete.
- `CONFIG_BPF_JIT=y`: abilita il Just-In-Time (JIT) compiler per BPF, migliorando le prestazioni del codice eBPF.

2.1.3 Strumenti per il Debug e il Monitoraggio

Per facilitare lo sviluppo e il debug di programmi eBPF, è possibile utilizzare i seguenti strumenti:

- **bpftool**: è uno strumento incluso nel kernel Linux che consente di ispezionare i programmi *BPF* caricati, le mappe e le statistiche.
- **tcpdump** e **Wireshark**: strumenti per l'analisi del traffico di rete che possono essere utilizzati per verificare il comportamento dei filtri BPF.
- **perf**: strumento per analizzare le prestazioni del codice eBPF, consentendo di verificare l'efficienza e l'impatto sulle prestazioni di rete.

Una configurazione adeguata di questo ambiente assicura che il codice eBPF possa essere sviluppato, eseguito e monitorato correttamente, garantendo prestazioni elevate e una gestione efficiente delle risorse di sistema.

2.2 eBPF

In questo paragrafo riprenderemo il concetto di cos'è eBPF, e di come può essere uno strumento fondamentale per l'analisi e prevenzione di possibili attacchi malevoli o possibili comportamenti sospetti, in maniera più approfondita e precisa.
Secondo la documentazione ufficiale di eBPF [15], questa tecnologia viene sviluppata nel kernel Linux e può eseguire programmi in maniera sandbox, in un contesto privilegiato come il kernel del sistema operativo. Viene utilizzato per estendere in modo sicuro ed efficiente le capacità del kernel senza richiedere la modifica del codice sorgente o il caricamento di moduli di esso [15].

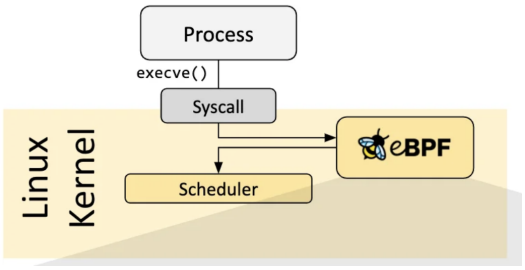

Figura 2.1: come eBPF funziona.
(Sorgente Immagine: `ebpf.io/what-is-ebpf/`)

Quando un hook predefinito non esiste per una specifica esigenza, è possibile creare un kernel probe (kprobe, letteralmente "sonda del kernel") o un user probe (uprobe, letteralmente "sonda dell'utente") per collegare programmi

eBPF praticamente ovunque nel kernel o nelle applicazioni utente 2.1. Questo significa che se non c'è un punto di aggancio predefinito all'interno del codice del kernel o dell'applicazione utente dove desideri eseguire un'operazione tramite eBPF, puoi creare manualmente un hook tramite kprobe o uprobe per raggiungere il tuo obiettivo. Questo offre una grande flessibilità nell'utilizzo di eBPF per monitorare e analizzare il comportamento del sistema o delle applicazioni, anche in contesti in cui non esistono hook predefiniti [15].

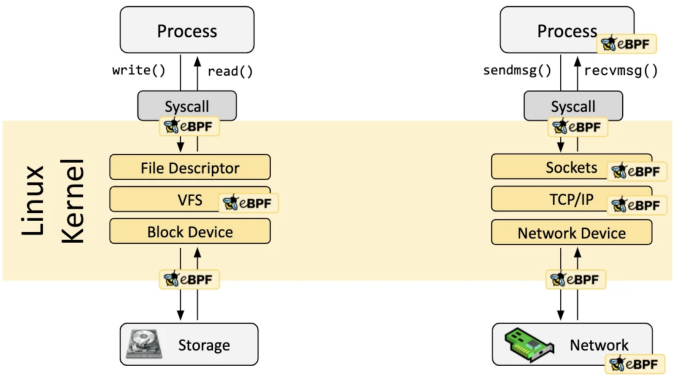

Figura 2.2: eBPF probe. (Sorgente Immagine: `ebpf.io/what-is-ebpf/`)

In molti scenari, eBPF non viene utilizzato direttamente, ma indirettamente tramite progetti come Cilium, bcc o bpftrace, che forniscono un'astrazione sopra eBPF e non richiedono la scrittura diretta di programmi, ma offrono invece la possibilità di specificare definizioni basate sull'intento che vengono quindi implementate con eBPF. Se non esiste un'astrazione a un livello più alto, i programmi devono essere scritti direttamente. Il kernel Linux si aspetta che i programmi eBPF siano caricati sotto forma di bytecode. Anche se è ovviamente possibile scrivere direttamente il bytecode, la pratica di sviluppo più comune è sfruttare un insieme di compilatori come LLVM per compilare il codice pseudo-C in bytecode eBPF [21].
Una volta individuato l'hook desiderato, il programma eBPF può essere ca-

ricato nel kernel Linux utilizzando la chiamata di sistema bpf. Questo viene tipicamente fatto utilizzando una delle librerie eBPF disponibili [15].

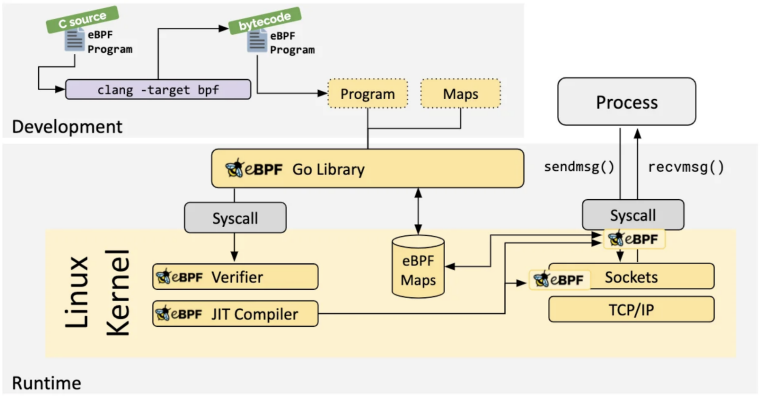

Figura 2.3: eBPF struttura
(Sorgente Immagine: `ebpf.io/what-is-ebpf/`)

Un'ulteriore caratteristica di eBPF è il passaggio di verifica, come mostrato in figura 2.3, cioè un passaggio intermedio al caricamento del file che va a controllare:

- controlla che il programma soddisfi diverse condizioni, ad esempio: il processo che carica il programma eBPF possiede i privilegi richiesti
- a meno che non sia abilitato l'eBPF non privilegiato, solo i processi privilegiati possono caricare programmi eBPF
- il programma non causa crash o danni al sistema. Il programma viene sempre eseguito fino al completamento (ovvero il programma non rimane in un loop infinito, bloccando ulteriori elaborazioni).

La fase di compilazione Just-in-Time (JIT) 2.4 traduce il bytecode generico del programma nel set di istruzioni specifico della macchina per ottimizzare la velocità di esecuzione del programma. Questo rende i programmi eBPF eseguiti con la stessa efficienza del codice del kernel compilato nativamente o del codice caricato come modulo del kernel. Un aspetto vitale dei programmi

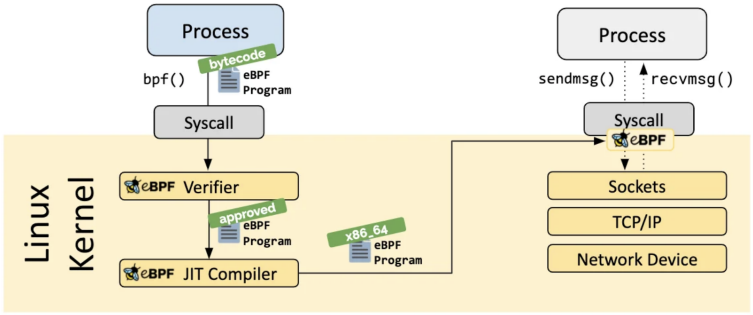

Figura 2.4: verifica del codice eBPF
(Sorgente Immagine: `ebpf.io/what-is-ebpf/`)

eBPF è la capacità di condividere le informazioni raccolte e di memorizzare lo stato. A tal fine, i programmi eBPF possono sfruttare il concetto di mappe eBPF per memorizzare e recuperare dati in una vasta gamma di strutture dati. Le mappe eBPF possono essere accessibili dai programmi eBPF così come dalle applicazioni nello spazio utente tramite una chiamata di sistema. Di seguito è riportato un elenco incompleto dei tipi di mappe supportati per dare un'idea della diversità delle strutture dati. Per vari tipi di mappe, è disponibile sia una variante condivisa che una per CPU.

- Tabelle hash, Array
- LRU (Least Recently Used)
- Buffer ad anello
- Trace dello stack
- LPM (Longest Prefix Match)

I programmi eBPF non possono richiamare funzioni kernel arbitrarie. Consentire ciò vincolerebbe i programmi eBPF a versioni specifiche del kernel e complicherebbe la compatibilità dei programmi. Invece, i programmi eBPF possono effettuare chiamate di funzione alle funzioni helper, una API ben nota e stabile offerta dal kernel [15].

2.2.1 Utilizzare eBPF per il monitoraggio dei processi

Il monitoraggio dei processi serve come componente fondamentale della sicurezza in tempo reale. Fondamentalmente, può rilevare processi inattesi o pattern di esecuzione che non dovrebbero verificarsi in un ambiente di produzione. Ad esempio, un server web in un ambiente di produzione non dovrebbe mai avviare una shell, e l'uso di un gestore di pacchetti per installare nuove dipendenze su un host potrebbe sollevare preoccupazioni [24]. Per fornire un vero albero dei processi per ciascun processo, viene utilizzata la cache dei processi dello spazio utente. Un vero albero dei processi si riferisce alla genealogia di tutti i processi che portano al processo che ha scatenato l'allarme, indipendentemente dallo stato dei processi genitori. Questa capacità è assente in molti strumenti convenzionali di sicurezza in tempo reale: l'esame del file system proc rivela che quando un processo termina, i suoi figli si uniscono immediatamente al processo con l'identificatore. Ciò fa sì che il kernel perda il contesto genealogico del processo, che potrebbe essere essenziale per identificare il servizio host in uso. Un altro vantaggio interessante di approfondire nel kernel oltre il livello della chiamata di sistema è la capacità di accedere a informazioni tipicamente non disponibili nello spazio utente. Ad esempio, il livello di un file nel file system sovrapposto. Queste informazioni comportano significative implicazioni per la sicurezza, poiché possono determinare se il file eseguito faceva parte dell'immagine di base del container o se è stato modificato (o creato) dalla versione originale dell'immagine di base. Inoltre, le credenziali del processo possono essere raccolte e integrate con altri eventi, consentendo la raccolta di un set completo di ID utente e di gruppo, capacità del kernel e metadati dei file eseguibili.

2.2.2 Monitoraggio attività di rete tramite XDP

XDP rappresenta una soluzione che contrasta sostanzialmente il bypass del kernel, introducendo la programmabilità tramite programmi eBPF 2.5 direttamente nello stack di rete del sistema operativo. Questo viene realizza-

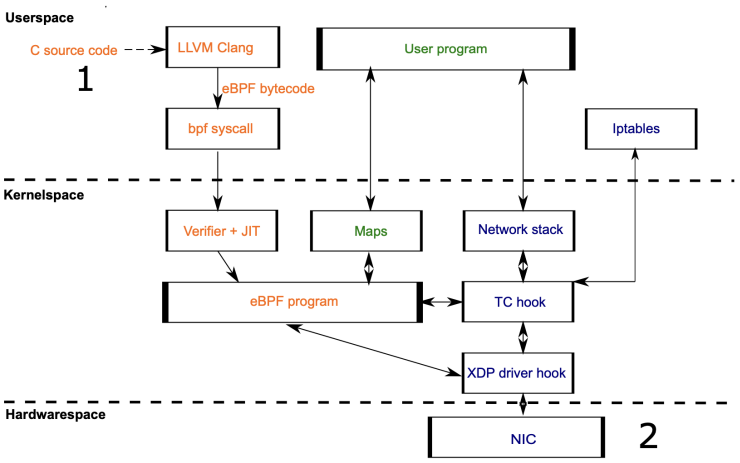

Figura 2.5: eBPF & XDP environment
(Autore Immagine: Huub Van Wieren)

to aggiungendo un nuovo strato nello stack di rete del kernel. Un programma eBPF può essere eseguito direttamente nel driver della NIC al primo punto di ricezione di un pacchetto, un'operazione conosciuta come 'XDP driver hook'. La syscall bpf consente l'esecuzione del bytecode eBPF come un 'programma eBPF'. È necessaria una specifica all'interno della syscall per indicare l'uso dell' 'XDP driver hook' 2.5, e di conseguenza, il programma eBPF può essere denominato 'programma XDP' [25].
Nella difesa contro gli attacchi Dos e DDoS, una caratteristica cruciale di un filtro di rete è la capacità di eliminare i pacchetti dannosi il più rapidamente possibile. XDP offre una velocità di scarto dei pacchetti significativamente superiore rispetto ai metodi tradizionali come Iptables.
Questo vantaggio deriva dal fatto che XDP può effettuare il filtraggio direttamente nel driver della scheda di rete, una fase antecedente rispetto a quella in cui Iptables opera. Il metodo più rapido per Iptables di scartare i pacchetti avviene nella fase di prerouting vicino all'hook di controllo del traffico (TC), dove i pacchetti in arrivo hanno già attraversato lo spazio del driver. Tuttavia, questo 'TC hook' segue l' 'XDP driver hook', risultando

quindi notevolmente più lento [25].

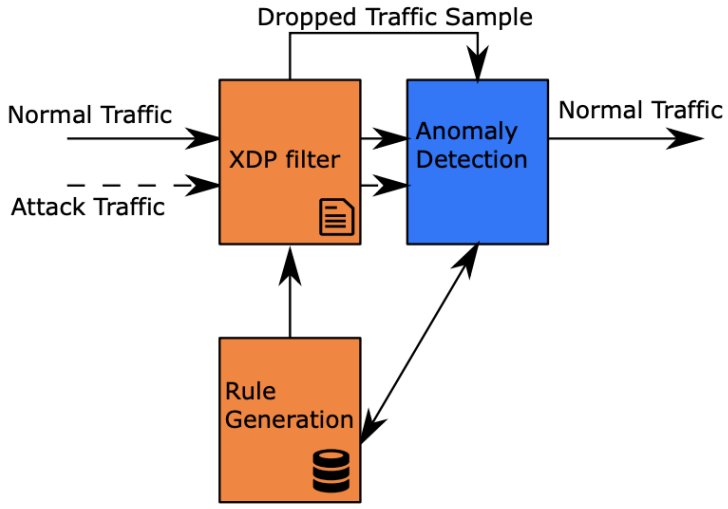

Figura 2.6: Struttura di codice XDP contro attacchi DoS/DDoS (Autore Immagine: Huub Van Wieren)

2.2.3 Implementazione di codice eBPF tramite BCC

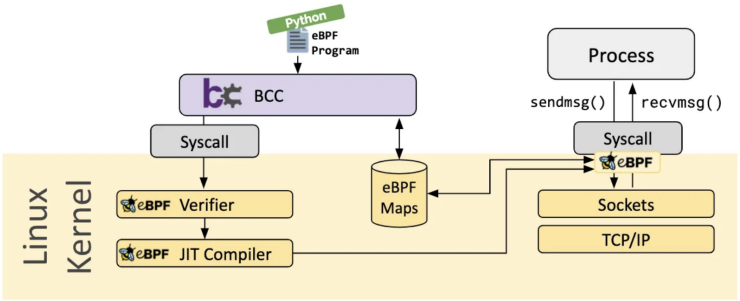

Figura 2.7: Come funziona bcc.
(Sorgente Immagine: `ebpf.io/what-is-ebpf/`)

BCC

Per sviluppare una porzione dell'agent tramite codice eBPF è stato utilizzato il framework BCC.
BCC (BPF Compiler Collection) è un toolkit potente e versatile per lo sviluppo di codice eBPF (Extended Berkeley Packet Filter). BCC fornisce una serie di strumenti, librerie e utilità per semplificare la scrittura, il caricamento e il debug di programmi eBPF. È particolarmente utile per l'osservabilità, il monitoraggio e il debug dei sistemi Linux [20].

Componenti Principali di BCC

Librerie e API

- **libbcc**: La libreria principale di BCC che offre API per caricare, attaccare e interagire con programmi eBPF.
- **Python e C/C++ Bindings**: BCC fornisce binding per Python e C/C++, rendendo più semplice l'interazione con eBPF da linguaggi di alto livello.

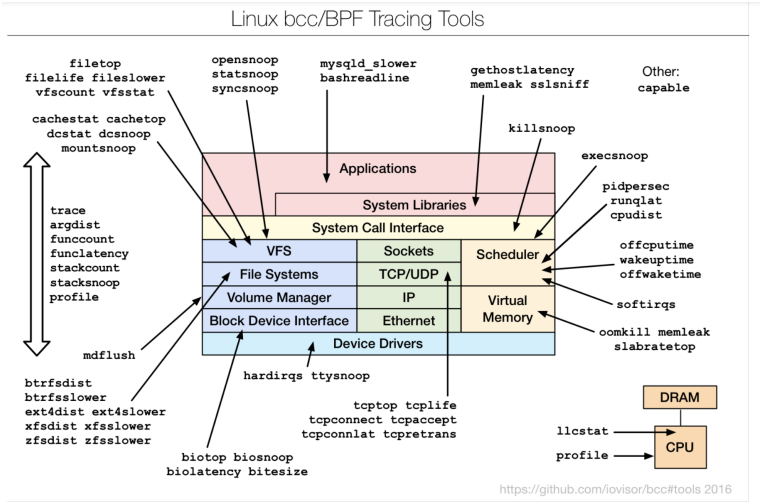

Figura 2.8: System calls che eBPF può intercettare.
(Sorgente Immagine: `www.brendangregg.com/ebpf.html`)

Compilazione Just-In-Time (JIT)

BCC utilizza LLVM per la compilazione Just-In-Time (JIT) di codice eBPF, permettendo di scrivere codice eBPF in C e compilarlo dinamicamente.

Vantaggi dell'utilizzo di BCC

- **Facilità d'uso:** BCC semplifica enormemente l'interazione con eBPF grazie alle sue API ad alto livello e agli strumenti preconfezionati.

- **Potente e flessibile:** BCC consente di scrivere programmi eBPF complessi per un'ampia gamma di utilizzi, dalla tracciatura di sistema al monitoraggio delle prestazioni.

- **Community e supporto:** BCC è ben supportato e ha una vasta comunità di utenti, con molte risorse disponibili per l'apprendimento e il troubleshooting.

Ply

Ply è un dynamic tracer per Linux basato su eBPF, è stato progettato pensando ai sistemi embedded, scritto in C e tutto ciò che serve per funzionare è libc e un moderno kernel Linux con supporto eBPF, il che significa che non dipende da LLVM per la generazione del programma [12]. Ha una sintassi simile al C per scrivere script ed è fortemente ispirato da awk(1) e dtrace(1).
Un esempio di codice è il seguente:

```
#!/usr/bin/env ply

kprobe:SyS_*
{
    @syscalls[caller] = count();
}
```

Questa probe verrà allegata a tutte le funzioni il cui nome inizia con SyS_, cioè a tutte le chiamate di sistema. Su ciascuna chiamata di sistema, la probe si attiverà e indicizzerà nell'aggregazione definita dall'utente syscalls utilizzando il chiamante del contatore del programma come chiave e urterà un contatore.
Ply compilerà lo script, lo allegherà alle probe corrispondenti e inizierà a raccogliere i dati. All'uscita, ply scaricherà i valori di tutte le mappe e aggregazioni definite dall'utente [12].

2.3 Analisi delle alternative per la raccolta di informazioni a livello user

2.3.1 Auditd

Auditd è un'utilità Linux che registra eventi rilevanti per la sicurezza del sistema. Può tracciare eventi come l'accesso ai file, le chiamate di sistema e i

comandi degli utenti [26]. Auditd può essere utilizzato per rilevare e indagare violazioni della sicurezza e per conformarsi ai requisiti normativi.

Come funziona auditd?

Come descritto da Red Hat [22], Auditd è un demone che viene eseguito in background e monitora il sistema per eventi correlati alla sicurezza. Quando si verifica un evento, auditd registra l'evento su un file chiamato audit.log, che può essere visualizzato utilizzando il comando ausearch.

Auditd può essere configurato per registrare una varietà di eventi diversi. I tipi di eventi più comuni registrati da auditd includono [26]:

- Eventi di accesso ai file: Questi eventi vengono registrati quando un utente accede a un file nel sistema. Gli eventi includono informazioni come l'utente che ha accesso al file, il file che è stato accesso e il tipo di accesso eseguito.

- Eventi di chiamata di sistema: Questi eventi vengono registrati quando un utente effettua una chiamata di sistema. Gli eventi includono informazioni come l'utente che ha effettuato la chiamata di sistema, la chiamata di sistema effettuata e gli argomenti passati alla chiamata di sistema.

- Eventi di comando dell'utente: Questi eventi vengono registrati quando un utente esegue un comando nel sistema. Gli eventi includono informazioni come l'utente che ha eseguito il comando, il comando che è stato eseguito e gli argomenti passati al comando.

Auditd può essere configurato per registrare eventi in una varietà di posizioni diverse. Le posizioni più comuni per i log di audit includono:

- L'hard disk locale: Questa è la posizione più comune per i log di audit. I log di audit memorizzati sull'hard disk locale possono essere visualizzati utilizzando il comando ausearch.

- Un server remoto: I log di audit possono anche essere memorizzati su un server remoto. Questo può essere utile per le organizzazioni che desiderano centralizzare i loro log di audit. I log di audit memorizzati su un server remoto possono essere visualizzati utilizzando il comando aureport.

- Un servizio basato su cloud: Esistono diversi servizi basati su cloud che offrono di memorizzare i log di audit. Questa può essere un'opzione comoda per le organizzazioni che non desiderano gestire i propri log di audit.

Un esempio di scrittura di regola audit, che va a monitorare i file [22]: /var/log/syslog, var/log/auth.log e /var/log/mysql/error.log può essere:

```
-w /var/log/syslog -p wa -k syslog_changes
-w /var/log/auth.log -p wa -k authlog_changes
-w /var/log/mysql/error.log -p wa -k mysql_errorlog_changes
```

- **-w /var/log/syslog**: Specifica il percorso del file `/var/log/syslog` da monitorare.

- **-p wa**: Specifica i permessi da monitorare, dove 'w' sta per scrittura e 'a' sta per attributi (modifica dei metadati). Questo significa che la regola traccerà qualsiasi attività di scrittura o modifica degli attributi del file.

- **-k syslog_changes**: Specifica la chiave di audit associata alla regola. In questo caso, la chiave di audit è `syslog_changes`, che può essere utilizzata per filtrare e cercare eventi di audit correlati al file di log del sistema.

Le altre due righe seguono lo stesso schema, monitorando rispettivamente i file di log di autenticazione (`/var/log/auth.log`) e gli errori di MySQL (`/var/log/mysql/error.log`).

2.3.2 Inotify

Inotify è un sottosistema del kernel Linux che fornisce un modo per i programmi di monitorare file e directory per i cambiamenti [17]. Inotify può essere utilizzato per tracciare una varietà di tipi di cambiamenti diversi, tra cui:

- Creazione di file
- Eliminazione di file
- Modifica dei file
- Rinomina dei file
- Creazione di directory
- Eliminazione di directory
- Rinomina delle directory

Come funziona inotify?
Inotify funziona creando un descrittore di watch per ogni file o directory che si desidera monitorare. Un descrittore di watch è un identificatore univoco che viene utilizzato per tracciare il file o la directory. Quando si verifica una modifica a un file o a una directory, inotify invia una notifica al programma che ha creato il descrittore di watch [17].
Il programma che ha creato il descrittore di watch può quindi utilizzare la notifica per intraprendere azioni appropriate. Ad esempio, un programma potrebbe utilizzare inotify per:

- Registra la modifica su un file o una directory
- Aggiorna un database
- Attiva un backup
- Invia una notifica via email

Inotify è uno strumento potente che può essere utilizzato per monitorare file e directory per i cambiamenti. È uno strumento prezioso per gli amministratori di sistema e gli sviluppatori che hanno bisogno di tracciare i cambiamenti nel file system [17].

```
/var/log/syslog IN_MODIFY /path/to/script.sh
/var/log/auth.log IN_MODIFY /path/to/script.sh
/var/log/mysql/error.log IN_MODIFY /path/to/script.sh
```

La regola di inotify sopra definita monitora tre file di log specifici all'interno del sistema. Di seguito sono fornite le spiegazioni dettagliate dei componenti della regola:

- **/var/log/syslog**: Specifica il percorso del file `/var/log/syslog` da monitorare.
- **IN_MODIFY**: Specifica l'evento da monitorare, in questo caso la modifica del file. Ciò indica che il programma monitorerà i cambiamenti apportati ai file di log specificati.
- **/path/to/script.sh**: Specifica il percorso dello script da eseguire quando si verifica l'evento specificato. In questo caso, verrà eseguito lo script `script.sh` quando si verifica una modifica ai file di log.

Le altre due righe seguono lo stesso schema, monitorando rispettivamente i file di log di autenticazione (`/var/log/auth.log`) e gli errori di MySQL (`/var/log/mysql/error.log`).

Capitolo 3

Snippet XDP e impatto applicativo

3.1 Elenco degli eventi e delle informazioni da monitorare

Quando un programma eBPF viene attaccato a un evento di rete, cioè all'arrivo di un pacchetto, può intercettare i pacchetti e modificarli, inoltrarli o scartarli. Il traffico può essere catturato in tre punti del suo ciclo di vita nel kernel, cioè in tre hook:

1. **Socket filter**: Intercetta i pacchetti dopo l'instradamento IP. Tuttavia, non possono essere apportate modifiche al traffico, che può solo essere osservato.

2. **Traffic Control (TC)**: L'hook del Traffic Control intercetta il traffico in entrata e in uscita (cioè ingress e egress) proprio prima che venga passato al Netfilter.

3. **eXpress Data Path (XDP)**: L'hook XDP si trova subito dopo il driver della scheda di rete. Può intercettare il traffico in modalità Driver (o Native) e in modalità Generica (o skb). La prima modalità deve

essere supportata dal particolare driver, è il punto più vicino alla scheda e ha vantaggi prestazionali elevati poiché il kernel non crea alcuna struttura dati, quindi non c'è praticamente alcun overhead. La seconda modalità è un modo per utilizzare XDP su driver che non la supportano. In entrambe le modalità, gli hook XDP possono operare solo sul traffico RX (cioè il traffico ricevuto dalla scheda dall'esterno dell'host) [23].

3.2 Descrizione dell'agent

Un esempio di codice XDP contro gli attacchi DoS può essere il seguente snippet realizzato tramite l'utilizzo del framework BCC, utilizzando Python per la parte userspace e codice eBPF per il programma XDP. Il seguente progetto simula un firewall contro gli attacchi DoS, andando ad autorizzare i client sospetti tramite SYN-Cookie e deallocando la memoria creata da connessioni sospette tramite NetFilter.

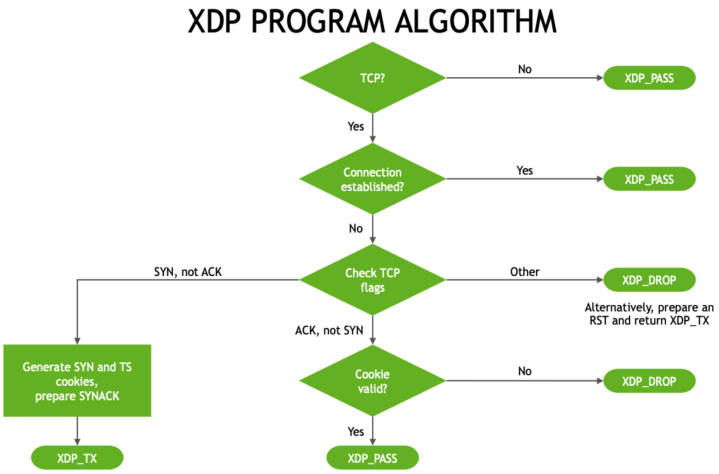

Figura 3.1: XDP firewall algoritmo
Autore Immagine: Maxim Mikityanskiy

Mi sono basato sul seguente algoritmo 3.1 per implementare lo snippet interessato.

Definizione delle Mappe

```
#include <uapi/linux/bpf.h>
#include <linux/bpf_common.h>
#include <linux/in.h>
#include <linux/if_ether.h>
#include <linux/ip.h>
#include <linux/tcp.h>

BPF_HASH(ip_blocked_map, __u32, __u64);  // Mappa per IP
    bloccati
BPF_HASH(ip_syn_count, __u32, __u64);    // Mappa per
    conteggio dei pacchetti SYN

__attribute__((section("xdp"), used))
int xdp_firewall(struct xdp_md *ctx);
```

Sono andato a creare 2 mappe bpf, la prima mappa, chiamata "ip_blocked_map" si riferisce alla mappa nella quale andrò a salvare tutti gli indirizzi IP cui a priori andrò a scartare i pacchetti entranti.
La seconda, "ip_syn_count", si riferisce a quella mappa utilizzata per salvare temporaneamente, tutti gli ip che inviano il primo SYN di connessione.
La funzione `xdp_firewall` è definita con l'attributo `__attribute__`, che specifica la sezione del codice `xdp`, consentendone l'esecuzione a livello XDP. L'attributo `used` indica che la funzione è usata direttamente dal kernel e non può essere eliminata dall'ottimizzatore.
La funzione `xdp_firewall` è dichiarata, con un parametro `ctx` di tipo `struct xdp_md`, che contiene informazioni sul pacchetto di rete in transito attraverso l'interfaccia di rete.

Analisi integrità dei pacchetti

```
int xdp_firewall(struct xdp_md *ctx) {
    void *data_end = (void *)(long)ctx->data_end;
    void *data = (void *)(long)ctx->data;
    struct ethhdr *ether = data;
    if (data + sizeof(*ether) > data_end) {
        return XDP_ABORTED;
    }
    if (ether->h_proto != htons(ETH_P_IP)) {
        return XDP_PASS;
    }
    data += sizeof(*ether);
    struct iphdr *ip = data;
    if (data + sizeof(*ip) > data_end) {
        return XDP_ABORTED;
    }
    // Solo TCP
    if (ip->protocol != IPPROTO_TCP)
        return XDP_PASS;

    struct tcphdr *tcp = (struct tcphdr *)(ip + 1);
    if ((void *)(tcp + 1) > data_end)
        return XDP_ABORTED;
```

Come controlli preliminari, vado a verificare che il pacchetto in arrivo sia integro e privo di errori, inoltre vado a verificare che la tipologia di pacchetto sia TCP, in caso contrario lascerò passare a priori tramite la regola `XDP_PASS`

Verifica della blacklist

```
__u32 src_ip = ip->saddr;

    // Controlla IP presente nella blacklist
    __u64 *blocked = ip_blocked_map.lookup(&src_ip);
    if (blocked) {
        bpf_trace_printk("Already blocked IP: %x\\n", src_ip)
   ;
        return XDP_DROP;
    }
```

Verifico che l'ip non sia presente nella blacklist, se presente applico `XDP_DROP` per scartare il pacchetto.

Gestione dei pacchetti TCP-SYN

```
// Gestisci pacchetti SYN
    if (tcp->syn && !tcp->ack) {
        __u64 *syn_count = ip_syn_count.lookup(&src_ip);

        if (syn_count) {
            // Se stato inviato un SYN, blocca l'IP
            __u64 val = 1;
            ip_blocked_map.update(&src_ip, &val);
            bpf_trace_printk("Blocked IP: %x for multiple
   SYNs\\n", src_ip);
            return XDP_DROP;
        } else {
            // Se primo SYN, genera SYN cookie
            __u64 initial_count = 1;
            ip_syn_count.update(&src_ip, &initial_count);

            __u32 cookie_seq = bpf_tcp_raw_gen_syncookie_ipv4
   (ip, tcp, sizeof(*tcp));
            if (cookie_seq == 0) {
                bpf_trace_printk("NOT generated SYN cookie
   for IP: %x\\n", src_ip);
                return XDP_PASS;  // Nessun cookie generato,
   passa il pacchetto
            }

            bpf_trace_printk("Generated SYN cookie for IP: %x
   \\n", src_ip);
```

Nel seguente frammento di codice gestisco l'arrivo di pacchetti TCP con Syn: all'inizio vado a controllare se l'host interessato ha già inviato pacchetti TCP con Syn, in caso affermativo, vado ad inserire l'IP nella blacklist in modo tale da bloccare tentativi di richieste di connessione future per sospetto attacco. Se invece è la prima volta che viene inviato il pacchetto TCP-Syn, l'host viene segnato nella mappa di controllo preventiva e inoltre viene generato un Syn-Cookie per andar verificare l'identità dell'host. Questa gestione è

fondamentale per proteggere il sistema da attacchi di tipo SYN flood, che tentano di saturare le risorse del server inviando una grande quantità di pacchetti SYN senza completare il processo di handshake TCP.

3.2.1 Generazione del SYN Cookie

Nel frammento di codice riportato, quando viene ricevuto un pacchetto TCP con il flag `SYN` impostato e l'host inviante non ha già inviato altri pacchetti SYN, viene generato un **SYN cookie**. Questo cookie è un meccanismo di sicurezza che permette di verificare se una richiesta di connessione è legittima, senza dover allocare subito risorse sul server.

La funzione `bpf_tcp_raw_gen_syncookie_ipv4(ip, tcp, sizeof(*tcp))` è utilizzata per generare il SYN cookie in modo raw, ovvero senza fare uso di un vero socket associato alla connessione. Il cookie generato è un numero sequenziale, creato a partire da informazioni critiche come l'indirizzo IP sorgente e la porta del client, che consente di evitare attacchi SYN flood, in cui gli attaccanti inviano grandi quantità di pacchetti SYN con indirizzi IP sorgente falsificati .

Se la funzione di generazione del SYN cookie restituisce 0, significa che il cookie non è stato generato e, pertanto, il pacchetto SYN viene lasciato passare attraverso il filtro XDP (`XDP_PASS`). Altrimenti, viene registrata la generazione del cookie con un messaggio di trace tramite la funzione `bpf_trace_printk`, che può essere visualizzata per il debug [14].

Differenza tra la generazione raw del SYN cookie e quella classica

- **Generazione raw del SYN cookie:** Quando si utilizza la funzione `bpf_tcp_raw_gen_syncookie_ipv4`, la generazione del SYN cookie avviene in modo diretto, senza il coinvolgimento di un socket associato alla connessione. Questo approccio è particolarmente utile in contesti come XDP, dove l'elaborazione del pacchetto avviene a livello di kernel e non c'è un vero socket aperto con l'host inviante. Il cookie è generato

unicamente sulla base dei dati del pacchetto TCP e delle informazioni del livello IP, rendendo il processo estremamente veloce ed efficiente.

- **Generazione classica del SYN cookie:** Nel caso tradizionale, la generazione del SYN cookie avviene all'interno di uno stack TCP completo, con un socket effettivo associato alla connessione. Questo metodo consente una maggiore interazione con lo stato della connessione, ma richiede più risorse, poiché il socket viene allocato per mantenere il contesto della connessione. Lo svantaggio di questo approccio è che può essere meno efficiente in termini di utilizzo delle risorse, specialmente durante un attacco DoS di tipo SYN flood, dove il server potrebbe essere sopraffatto dall'elevato numero di connessioni in sospeso.

Invio pacchetto TCP con frammento SYN-ACK

```
// Prepara il pacchetto SYN-ACK
            unsigned char tmp_mac[ETH_ALEN];
            // Inverti gli indirizzi MAC
            __builtin_memcpy(tmp_mac, ether->h_source, ETH_ALEN);   // Salva MAC di origine temporaneamente
            __builtin_memcpy(ether->h_source, ether->h_dest, ETH_ALEN); // MAC di destinazione in origine
            __builtin_memcpy(ether->h_dest, tmp_mac, ETH_ALEN);     // MAC salvato come destinazione

            tcp->ack = 1;
            tcp->ack_seq = htonl(ntohl(tcp->seq) + 1);  // ACK del SYN ricevuto
            tcp->seq = htonl(cookie_seq);  // Imposta il cookie generato come numero di sequenza

            // Inverti gli indirizzi IP
            __u32 tmp_ip = ip->saddr;
            ip->saddr = ip->daddr;
            ip->daddr = tmp_ip;

```

```
17              // Inverti le porte TCP
18              __u16 tmp_port = tcp->source;
19              tcp->source = tcp->dest;
20              tcp->dest = tmp_port;
21
22              // Calcolo del checksum TCP
23              tcp->check = 0;  // Azzera il checksum TCP
24              __u32 tcp_csum = bpf_csum_diff(0, 0, (__be32 *)
    tcp, sizeof(*tcp), 0);
25              tcp->check = tcp_csum;
26
27              // Calcolo del checksum IP
28              ip->check = 0;  // Azzera il checksum IP
29              __u32 ip_csum = bpf_csum_diff(0, 0, (__be32 *)ip,
     sizeof(*ip), 0);
30              ip->check = ip_csum;
31
32              return XDP_TX;  // Trasmetti il pacchetto SYN-ACK
33          }
34      }
```

Nel seguente frammento di codice, vado a creare manualmente il pacchetto di risposta TCP contenente SYN-ACK, andando ad inserire il valore generato precedentemente del Syn-Cookie. Tramite la funzione `bpf_csum_diff()` vado a ricalcolare il checksum: il ricalcolo del checksum è necessario ogni volta che si modifica un pacchetto IP, in quanto il checksum serve a garantire l'integrità dell'header del pacchetto durante la trasmissione. Il checksum IP è un campo dell'header IP che verifica la correttezza dei dati presenti nell'header stesso. Se i dati vengono modificati, come nel caso di una manipolazione o generazione di un SYN cookie, è necessario aggiornare anche il checksum per evitare che il pacchetto venga scartato dai dispositivi di rete lungo il percorso.

Verifica SYN-Cookie

```
// Gestisci pacchetti ACK
    if (tcp->ack && !tcp->syn) {
        if (!bpf_tcp_raw_check_syncookie_ipv4(ip, tcp)) {
            __u64 val = 1;
            ip_blocked_map.update(&src_ip, &val);
            bpf_trace_printk("Blocked IP: %x for invalid SYN
    cookie\\n", src_ip);
            return XDP_DROP;
        }

        // SYN cookie valido, azzera il conteggio SYN inviati
     per questo IP
        __u64 *syn_count = ip_syn_count.lookup(&src_ip);
        if (syn_count) {
            __u64 zero_count = 0;
            ip_syn_count.update(&src_ip, &zero_count);  //
    Reset SYN count
        }

        bpf_trace_printk("Valid ACK for IP: %x\\n", src_ip);
        return XDP_PASS;
    }

    return XDP_PASS;
}
```

Infine, se il client è legittimo, andrà a completare la connessione al server (Three-Way handshake) tramite la ritrasmissione del pacchetto TCP contenente l'ACK. Tramite la funzione `bpf_tcp_raw_check_syncookie_ipv4()` si può controllare se il valore ritrasmesso dal client è il Syn-Cookie corretto, in caso affermativo allora il client è legittimo; se invece il client ritrasmette un valore del Syn-Cookie non valido, quel pacchetto e tutte le richieste di connessione successive verranno scartate (l'IP del client verrà inserito nella blacklist).

3.3 Risultati ottenuti

3.3.1 Client non malevolo

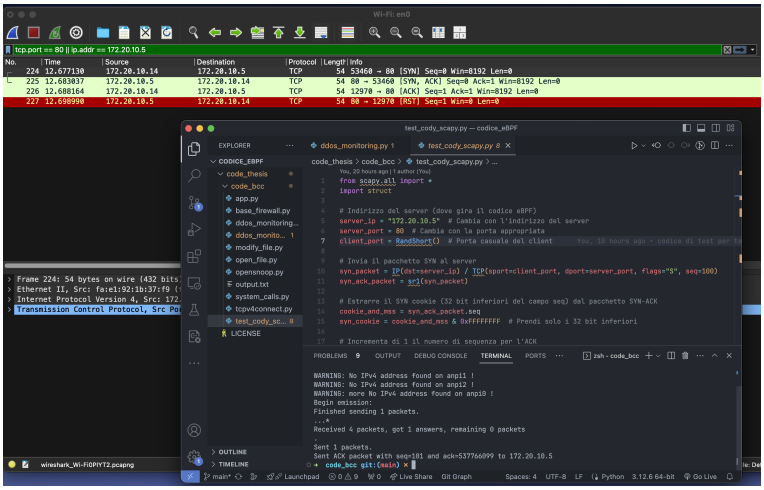

Figura 3.2: Tentativo richiesta connessione client

Le seguenti figure rappresentano il tentativo 3.2 e la buona riuscita 3.3 di creazione di connessione tra client e server: il client (simulato dal codice Scapy) genera il pacchetto TCP contenente Syn, per poi inviarlo al server. Il server, dopo i controlli preliminari, genera il syn-cookie e re-invia il pacchetto TCP con Syn-Ack al client, ricalcola l'ack e invia in risposta al server. In conclusione il server controlla la correttezza del Syn-Cookie restituito, e in caso positivo autorizza il client per la connessione ad esso

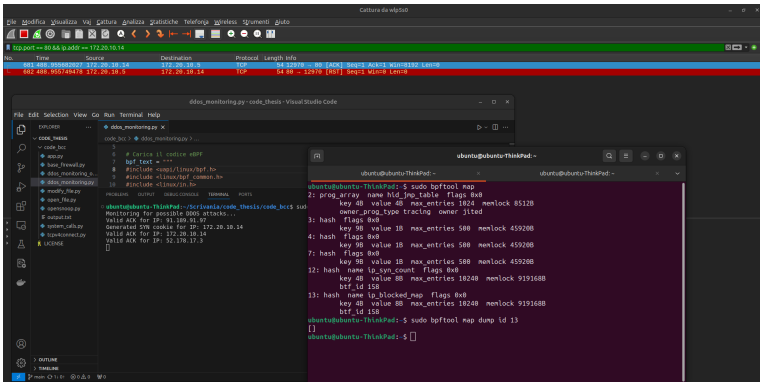

Figura 3.3: Connessione accettata dal server

3.3.2 Client malevolo - Attacco DoS Syn Flood

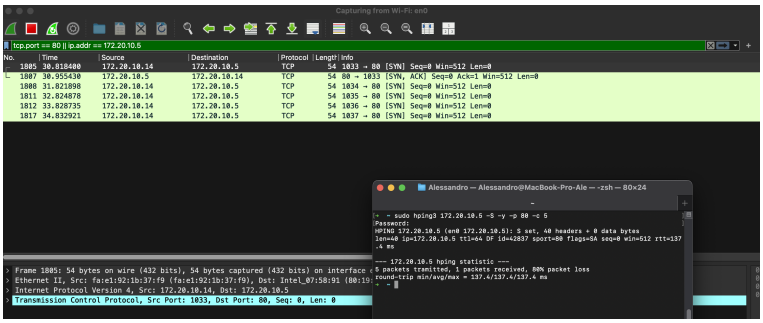

Figura 3.4: Tentativo attacco DoS Syn Flood attaccante

Simulando un attacco DoS Syn-Flood tramite hping3 3.4, il client invia molteplici pacchetti TCP contenenti Syn, ma poichè l'attaccante non rielabora il pacchetto TCP contenente Syn-Ack, viene subito bloccato dal server grazie al controllo sulle mappe contenenti gli IP già richiedenti di connessione. L'immagine sottostante 3.5 dimostra come funziona il codice XDP, scartando i pacchetti ancora prima che entrino nello stack di rete applicativo.

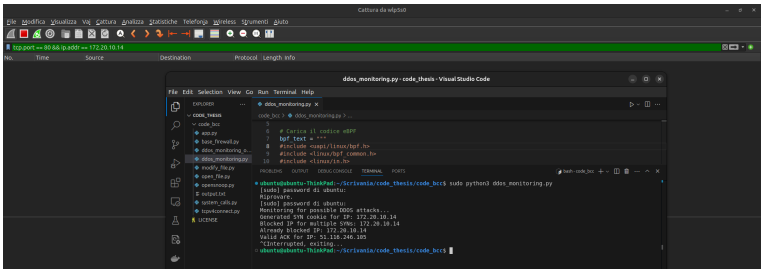

Figura 3.5: Schermata di blocco del server

3.4 NetFilter

Netfilter è un framework integrato nel kernel Linux utilizzato per l'elaborazione dei pacchetti di rete, offrendo una potente infrastruttura per il filtraggio, il NAT (Network Address Translation) e la gestione delle connessioni. Una delle sue principali applicazioni è la difesa da attacchi DoS e DDoS (Distributed Denial of Service), attraverso meccanismi di filtraggio avanzato e gestione efficiente delle risorse di sistema, come la memoria [18].

3.4.1 Integrazione di Netfilter con eBPF

Un approccio efficace per mitigare attacchi DoS e DDoS consiste nell'integrare Netfilter con programmi eBPF, sfruttando la flessibilità e le elevate prestazioni di eBPF per implementare logiche di filtraggio dinamico. Questa integrazione consente di definire regole che possono essere eseguite direttamente nel kernel, riducendo drasticamente l'overhead dovuto alla gestione del traffico sospetto.

Quando viene rilevato un attacco DDoS, Netfilter può intervenire in diversi modi:

- **Bloccare pacchetti malevoli**: attraverso regole di filtraggio, Netfilter può bloccare pacchetti provenienti da fonti sospette, riducendo il carico di lavoro del sistema.

- **Limita il numero di connessioni simultanee**: configurazioni specifiche in Netfilter possono prevenire che una singola fonte saturi le risorse di rete.

- **Rate limiting**: l'integrazione di Netfilter con tc (traffic control) consente di applicare il rate limiting sui pacchetti, impedendo che il traffico eccessivo provochi il sovraccarico della rete.

3.4.2 Deallocazione della Memoria tramite Netfilter

Durante un attacco DoS, uno degli obiettivi principali è garantire che il sistema non esaurisca risorse cruciali, come la memoria, a causa del numero elevato di pacchetti in entrata. Netfilter può contribuire alla difesa del sistema implementando meccanismi di deallocazione della memoria per pacchetti e connessioni non necessari, proteggendo il kernel da eventuali sovraccarichi.

Le principali tecniche di gestione della memoria tramite Netfilter includono:

- **Conntrack**: Netfilter utilizza il modulo conntrack per gestire le connessioni di rete. In caso di attacco DoS, connessioni sospette o inattive vengono eliminate dal connection tracking table, liberando memoria occupata da stati di connessione non necessari.

- **Timeout delle connessioni**: configurando valori di timeout più aggressivi per le connessioni incomplete (ad esempio SYN flood), è possibile garantire che la memoria utilizzata per tracciare queste connessioni venga liberata rapidamente, riducendo il rischio di esaurimento della memoria.

- **Limite massimo di connessioni tracciate**: Netfilter permette di impostare un limite massimo per il numero di connessioni tracciate. Una volta raggiunto il limite, le nuove connessioni sospette possono essere automaticamente scartate, liberando memoria.

3.4.3 Ottimizzazione delle Risorse tramite XDP

Quando si utilizza XDP (eXpress Data Path) insieme a Netfilter, è possibile scartare pacchetti indesiderati direttamente al livello della scheda di rete, prima che entrino nello stack del kernel. Questo permette di ridurre il carico sulla CPU e la memoria allocata per il trattamento dei pacchetti, migliorando così la resilienza del sistema contro attacchi DoS e DDoS di ampia scala.

In conclusione, Netfilter, combinato con eBPF e XDP, fornisce un potente strumento per la gestione efficiente della memoria e la protezione delle risorse di sistema in scenari di attacco DoS, consentendo la deallocazione tempestiva di connessioni e pacchetti non necessari e garantendo una risposta rapida e scalabile agli attacchi di rete.

Capitolo 4

Analisi e confronto dei risultati

4.1 Confronto prestazionale tra la soluzione kernel space e user space per la raccolta delle informazioni

1. **Flessibilità nell'analisi degli eventi**:

   - **eBPF**: eBPF (Berkeley Packet Filter esteso) offre un livello di flessibilità senza precedenti nell'analisi degli eventi grazie alla sua capacità di eseguire programmi personalizzati direttamente nel kernel Linux. Questi programmi possono essere scritti per monitorare, filtrare e analizzare una vasta gamma di eventi di sistema in tempo reale. Con eBPF, gli sviluppatori possono scrivere programmi specifici per catturare eventi rilevanti per le loro esigenze, consentendo una visione dettagliata e personalizzata dello stato del sistema. Ciò offre una flessibilità senza precedenti nell'analisi degli eventi, consentendo di identificare e risolvere problemi di prestazioni, sicurezza e altro ancora.
   - **auditd**: Sebbene auditd offra una serie di funzionalità di monitoraggio e registrazione degli eventi, la sua flessibilità è limitata rispetto a eBPF. Mentre auditd è utile per il monitoraggio degli

eventi di sistema standard e delle attività degli utenti, potrebbe non essere in grado di soddisfare le esigenze di analisi degli eventi più complessi e personalizzati come può fare eBPF.

2. **Profondità di analisi**:

   - **eBPF**: Grazie alla sua capacità di operare direttamente nel kernel Linux, eBPF offre una visione più dettagliata e granulare degli eventi di sistema rispetto a auditd. Poiché i programmi eBPF possono essere eseguiti in modo efficiente nel kernel, sono in grado di accedere a informazioni più dettagliate sui processi, sulla rete, sulle chiamate di sistema e su altri aspetti del sistema operativo.
   - **auditd**: Sebbene auditd fornisca una panoramica generale degli eventi di sistema, potrebbe non offrire la stessa profondità di analisi di eBPF. Auditd registra principalmente eventi di alto livello come modifiche ai file, accessi alla rete e operazioni di autenticazione. Mentre questo fornisce una visione generale delle attività di sistema, potrebbe non essere sufficiente per identificare e risolvere problemi più complessi o per ottenere una comprensione dettagliata del comportamento del sistema. Di conseguenza, la profondità di analisi offerta da auditd potrebbe risultare limitata in confronto a quella di eBPF.

3. **Overhead del sistema**:

   - **eBPF**: In generale, eBPF tende ad avere un overhead inferiore rispetto a auditd in termini di utilizzo delle risorse di sistema. Poiché i programmi eBPF sono eseguiti direttamente nel kernel, non c'è la necessità di passare attraverso il livello user-space, il che riduce significativamente il carico sul sistema. Inoltre, poiché eBPF offre la possibilità di filtrare e analizzare gli eventi di sistema in modo efficiente, può ridurre il carico complessivo del sistema rispetto a meccanismi più tradizionali come auditd.

- **auditd**: auditd, essendo un processo che opera a livello user-space, può richiedere risorse aggiuntive del sistema (come mostrato in figura 4.1), specialmente quando si occupa di grandi volumi di dati o quando vengono applicate regole di audit complesse. Il passaggio dei dati tra lo spazio utente e il kernel può comportare un overhead significativo, soprattutto in scenari ad alta intensità di eventi. Di conseguenza, auditd potrebbe richiedere più risorse di sistema rispetto a eBPF per gestire lo stesso carico di lavoro.

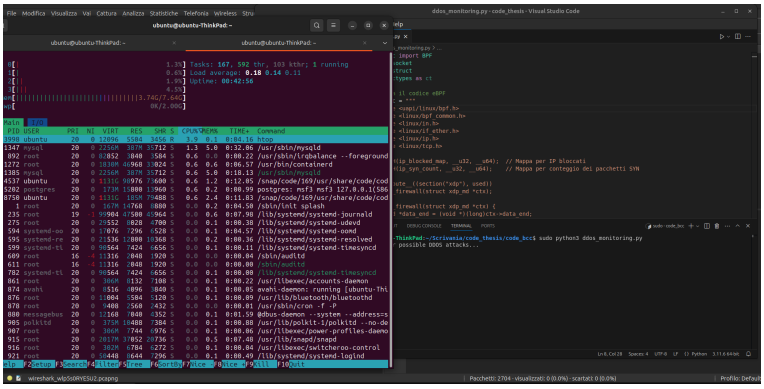

Figura 4.1: Test overhead tramite Htop

In conclusione, mentre sia eBPF che auditd offrono soluzioni valide per il monitoraggio degli eventi di sistema, ciascuna ha i suoi vantaggi e le sue limitazioni. La scelta tra i due dipenderà dalle esigenze specifiche del caso d'uso, dalle risorse disponibili e dalla complessità del sistema da monitorare.

4.2 Strumenti di testing

4.2.1 Hping3

hping3 è uno strumento di linea di comando open-source utilizzato per generare e manipolare pacchetti di rete. Originariamente creato come un'alternativa avanzata a `ping`, `hping3` offre una vasta gamma di funzionalità

per testare la sicurezza e le prestazioni di una rete, simulare attacchi o semplicemente esaminare il comportamento di specifici protocolli. A differenza di `ping`, che invia solo pacchetti ICMP, `hping3` può inviare pacchetti TCP, UDP, ICMP o RAW-IP [16].

Caratteristiche principali di hping3

- **Supporto a diversi protocolli:** hping3 può generare pacchetti di vari protocolli di rete, inclusi TCP, UDP, ICMP e RAW-IP, permettendo quindi di eseguire test specifici su vari livelli della pila di protocolli.
- **Modifica dettagliata dei pacchetti:** L'utente può manipolare i pacchetti generati in modo dettagliato, modificando il numero di sequenza, i flag TCP (SYN, ACK, RST, PSH, FIN, URG), la dimensione dei pacchetti, il TTL (Time To Live), i campi IP e le opzioni di protocollo.
- **Flooding:** È possibile utilizzare `hping3` per inviare rapidamente un elevato numero di pacchetti al fine di simulare attacchi di tipo Denial of Service (DoS) o Distributed Denial of Service (DDoS), come un attacco SYN flood.
- **Traceroute TCP:** hping3 può eseguire una traceroute utilizzando pacchetti TCP al posto dei tradizionali ICMP, utile per esaminare percorsi di rete con filtri o firewall che bloccano pacchetti ICMP.
- **Scanning di porte:** hping3 può essere utilizzato come uno strumento di port scanning TCP avanzato, testando l'apertura o la chiusura di porte su un host remoto.

Esempi di utilizzo di hping3

- **Invio di un pacchetto TCP SYN:**

```
1    hping3 -S -p 80 <indirizzo IP>
2
```

Questo comando invia un pacchetto TCP con il flag `SYN` impostato (indicando l'inizio di una richiesta di connessione TCP) alla porta 80 dell'indirizzo IP specificato.

- **Attacco SYN flood:**

```
hping3 -S --flood -p 80 <indirizzo IP>
```

Con l'opzione `--flood`, hping3 inonda la porta 80 dell'host di destinazione con pacchetti SYN. Questo tipo di attacco, noto come SYN flood, è utilizzato per tentare di saturare le risorse del server, obbligandolo a gestire un elevato numero di richieste di handshake TCP non complete.

- **Traceroute TCP:**

```
hping3 --traceroute -V -p 80 <indirizzo IP>
```

Questo comando esegue una traceroute utilizzando pacchetti TCP con destinazione la porta 80 dell'host di destinazione, mostrando il percorso seguito dai pacchetti per raggiungere l'host remoto.

- **Scansione di porte TCP:**

```
hping3 -S -p ++50 <indirizzo IP>
```

In questo esempio, hping3 esegue una scansione incrementale delle porte TCP sull'indirizzo IP specificato, partendo dalla porta 50 e incrementandola ad ogni pacchetto inviato. È possibile così determinare quali porte sono aperte o chiuse sull'host di destinazione.

Utilizzo di hping3 per simulare attacchi DoS

hping3 è spesso usato per simulare attacchi DoS, come SYN flood o UDP flood. In questi scenari, lo strumento invia pacchetti malformati o un volume elevato di richieste per testare la resilienza di un sistema agli attacchi di tipo

Denial of Service. L'opzione `--flood` è utile per generare rapidamente un grande numero di pacchetti, saturando le risorse di rete o server target. È tuttavia importante usare questi comandi solo in ambienti controllati o autorizzati, in quanto un uso improprio di `hping3` può compromettere i servizi di rete e violare le leggi su sicurezza informatica.

4.2.2 WireShark

Wireshark è uno degli strumenti più utilizzati per l'analisi del traffico di rete. Si tratta di un software open-source che permette di catturare e visualizzare in tempo reale i pacchetti che transitano su una rete. Grazie alla sua interfaccia grafica user-friendly e alle numerose funzionalità avanzate, Wireshark è utilizzato da amministratori di rete, sviluppatori e professionisti della sicurezza per diagnosticare problemi di rete, monitorare il traffico e analizzare attacchi.

Funzionalità principali di Wireshark

Wireshark offre una vasta gamma di funzionalità per l'analisi dettagliata del traffico di rete. Alcune delle sue caratteristiche più importanti includono:

- **Cattura di pacchetti in tempo reale:** Wireshark è in grado di catturare pacchetti direttamente dalle interfacce di rete disponibili sul sistema, consentendo agli utenti di osservare e analizzare il traffico in tempo reale.
- **Supporto a vari protocolli di rete:** Wireshark supporta centinaia di protocolli di rete, come TCP, UDP, ICMP, HTTP, HTTPS, DNS, e molti altri. Ogni protocollo è decodificato e presentato in modo dettagliato.
- **Filtri di cattura e di visualizzazione:** Gli utenti possono definire filtri sia durante la fase di cattura dei pacchetti che durante l'analisi. I filtri di cattura limitano i pacchetti che vengono catturati, mentre

i filtri di visualizzazione permettono di selezionare specifici pacchetti dalla cattura per un'analisi più dettagliata.

- **Analisi dei pacchetti:** Wireshark consente una visualizzazione dettagliata dei pacchetti catturati, mostrando le informazioni a livello di collegamento dati, rete, trasporto e applicazione. È possibile esplorare l'header di ogni livello per osservare campi specifici, come indirizzi IP, numeri di porta, flag TCP, checksum e altro ancora.
- **Seguire flussi TCP:** Wireshark è in grado di ricostruire i flussi TCP, rendendo più facile capire lo scambio di dati tra client e server. Questa funzione è particolarmente utile per analizzare la comunicazione tra due host o per diagnosticare problemi di connettività.
- **Ricerca e filtraggio avanzato:** Wireshark offre potenti strumenti di ricerca e filtraggio. Gli utenti possono cercare pacchetti specifici per indirizzi IP, porte, protocolli o contenuti, facilitando l'analisi di grandi volumi di traffico.

Utilizzi principali di Wireshark

Wireshark è estremamente versatile e può essere utilizzato in molti contesti diversi, come illustrato di seguito:

- **Diagnostica dei problemi di rete:** Gli amministratori di rete utilizzano Wireshark per identificare problemi di latenza, pacchetti persi o altre anomalie nel traffico di rete. Ad esempio, è possibile individuare dove si verificano ritardi nella trasmissione di pacchetti TCP o UDP, o scoprire se un pacchetto viene frammentato.
- **Analisi di sicurezza:** Wireshark è uno strumento prezioso per identificare attacchi informatici. Ad esempio, permette di rilevare attacchi di tipo DoS e DDoS, come SYN flood, osservando un anomalo incremento del traffico SYN. Può anche aiutare a scoprire pacchetti malformati, tentativi di scansione di porte o tentativi di spoofing.

- **Monitoraggio della rete:** Wireshark consente di monitorare l'intero traffico di rete in tempo reale, visualizzando ogni pacchetto che transita su un'interfaccia. Questo è utile in ambienti dove è necessario avere una visione completa di ciò che accade a livello di rete.

4.2.3 Codice Scapy

Scapy è una potente libreria Python utilizzata per la manipolazione e l'analisi dei pacchetti di rete. Permette di costruire, inviare, ricevere e interpretare pacchetti a diversi livelli di rete (livello link, rete, trasporto e applicazione). Grazie alla sua flessibilità, Scapy è ampiamente utilizzata per test di rete, simulazioni di traffico, analisi di sicurezza e per generare pacchetti personalizzati [19].

Funzionalità principali di Scapy

Scapy offre una vasta gamma di funzionalità che la rendono uno strumento molto utile per sviluppatori, ricercatori e professionisti della sicurezza. Alcune delle sue capacità principali includono:

- **Creazione di pacchetti:** Scapy consente di creare pacchetti a partire da zero, permettendo di definire tutti i campi e gli header necessari. È possibile costruire pacchetti per numerosi protocolli come Ethernet, IP, TCP, UDP, ICMP, DNS, HTTP, ecc.
- **Manipolazione dei pacchetti:** Gli utenti possono modificare i campi di un pacchetto, aggiungere nuovi strati, alterare checksum, modificare indirizzi IP o porte, e simulare vari scenari di traffico di rete.
- **Cattura del traffico:** Scapy è in grado di catturare pacchetti di rete in tempo reale, funzionando come uno sniffer, per poi analizzarli e manipolarli ulteriormente.
- **Invio di pacchetti:** Una delle caratteristiche più potenti di Scapy è la possibilità di inviare pacchetti personalizzati attraverso la rete,

simulando il comportamento di un client o server. Questa capacità è utile per testare la robustezza delle applicazioni di rete.

- **Tracciamento e risposta automatica:** Scapy può essere programmata per inviare risposte automatiche ai pacchetti in arrivo, simulando il comportamento di un sistema completo o di un client specifico.

Simulazione di un client non malevolo con Scapy

Per simulare un client TCP che esegue una semplice richiesta di connessione a un server, si può utilizzare Scapy per costruire un pacchetto SYN per avviare la connessione, inviare un pacchetto ACK per completare il three-way handshake.

```
from scapy.all import *
import struct

# Indirizzo del server (dove gira il codice eBPF)
server_ip = "{indirizzo del server}"
server_port = 80
client_port = RandShort()

# Invia il pacchetto SYN al server
syn_packet = IP(dst=server_ip) / TCP(sport=client_port, dport
    =server_port, flags="S", seq=100)
syn_ack_packet = sr1(syn_packet)

# Estrarre il SYN cookie (32 bit inferiori del campo seq) dal
    pacchetto SYN-ACK
cookie_and_mss = syn_ack_packet.seq
syn_cookie = cookie_and_mss & 0xFFFFFFFF  # Prendi solo i 32
    bit inferiori

# Incrementa di 1 il numero di sequenza per l'ACK
ack_number = syn_ack_packet.ack

```

```
20 # Invia il pacchetto ACK con il SYN cookie come numero di
       sequenza
21 ack_packet = IP(dst=server_ip) / TCP(sport=client_port, dport
       =server_port, flags="A", seq=syn_ack_packet.ack, ack=
       syn_cookie + 1)
22 send(ack_packet)
23
24 print(f"Sent ACK packet with seq={syn_ack_packet.ack} and ack
       ={syn_cookie + 1} to {server_ip}")
```

In questo esempio, viene simulato il comportamento di un client TCP che:

- Invia un pacchetto `SYN` per richiedere l'apertura di una connessione TCP.
- Riceve e risponde con un pacchetto `SYN-ACK` per completare il three-way handshake (il server gli ritorna all'interno del pacchetto il Syn-Cookie).
- Rielabora il pacchetto TCP con l'ACK corretto di risposta per creare una connessione con il server. Bisogna ricordare che in questo passaggio, il client in caso scarti il campo contenente il SYN-Cookie inviato dal server non potrà creare la connessione e i suoi pacchetti verranno scartati.

Conclusioni

Conclusioni

Nel corso di questa tesi è stato presentato uno snippet di codice eBPF/-XDP per la gestione e mitigazione degli attacchi DoS/DDoS, con particolare attenzione agli attacchi di tipo SYN Flood. Sebbene lo snippet proposto mostri un'efficace capacità di identificare e bloccare pacchetti sospetti provenienti da singole fonti (attacchi DoS), presenta delle difficoltà nel fronteggiare attacchi DDoS provenienti da numerose fonti distribuite.

Un attacco DDoS SYN Flood, infatti, comporta la ricezione simultanea di pacchetti SYN da innumerevoli indirizzi IP, rendendo inefficiente la gestione delle connessioni malevole tramite una semplice mappa. L'utilizzo di una mappa risulta infatti poco pratico in quanto la memoria necessaria per memorizzare un numero elevato di IP può diventare rapidamente insostenibile. Inoltre, l'operazione di ricerca e aggiornamento all'interno di una mappa potrebbe comportare un sovraccarico in termini di tempo di esecuzione.

Per affrontare tale sfida, una possibile soluzione potrebbe essere l'utilizzo di una coda, che permetterebbe di gestire in maniera più efficiente le richieste e le fonti di attacco. Tuttavia, l'integrazione di una coda in un programma eBPF presenta ancora delle limitazioni significative. eBPF, pur essendo una tecnologia avanzata e versatile, non è attualmente ottimizzato per la gestione delle code o dei loop di controllo che sarebbero necessari per scansionare una coda alla ricerca di un determinato IP. In particolare, manca una funzione dedicata per la ricerca efficiente all'interno di una coda, il che rende complesso

implementare una soluzione scalabile per attacchi di grande entità come i DDoS distribuiti.

In conclusione, sebbene l'approccio eBPF/XDP utilizzato offra una soluzione valida per mitigare attacchi DoS, ulteriori ottimizzazioni e sviluppi sono necessari affinché possa diventare una tecnologia competitiva per la gestione di attacchi di vasta portata come, ad esempio, attacchi DDoS Syn Flood. In particolare, l'implementazione di meccanismi avanzati per la gestione delle code e delle risorse di sistema potrebbe rappresentare un passo fondamentale verso una protezione più efficace e scalabile nei confronti degli attacchi distribuiti.

Quanto specificato soprastante non deve essere interpretato come un vincolo tale per cui l'utilizzo di eBPF diventi effimero. Infatti, nonostante le limitazioni menzionate, numerose aziende come Netflix, Cloudflare, Facebook e Datadog utilizzano snippet eBPF come strumenti di analisi del traffico di rete e metriche di sicurezza contro attacchi hacker, ciò è dovuto alla capacità di eBPF di operare direttamente a livello del kernel, permettendo un monitoraggio altamente granulare e a basso overhead del traffico di rete, senza richiedere modifiche al codice sorgente del kernel.

Inoltre, eBPF consente un intervento rapido e personalizzato nella gestione del traffico sospetto, migliorando la capacità di rilevare e mitigare attacchi in tempo reale, come nel caso presentato dei DoS - Syn flood. La flessibilità nella definizione di regole specifiche per filtrare pacchetti, la sua integrazione con altri strumenti di sicurezza, e l'ottimizzazione delle performance di rete lo rendono una scelta ideale per ambienti su larga scala che richiedono sicurezza e prestazioni elevate.

Anche se l'assenza di un supporto nativo per queue e loop complessi rappresenta una sfida per il controllo avanzato di attacchi distribuiti, il vantaggio di avere una visione completa delle operazioni a livello di kernel e di poter applicare politiche di sicurezza avanzate continua a rendere eBPF un componente fondamentale nella difesa delle infrastrutture aziendali.

Il codice XDP sviluppato contro attacchi DoS Syn Flood è un esempio di

come l'eBPF possa essere utilizzato per creare soluzioni leggere e altamente performanti contro attacchi di rete. Nonostante le limitazioni nella gestione di attacchi provenienti da numerose fonti, senza l'ausilio di strutture come code o algoritmi di ricerca più complessi, la sua integrazione in ambienti aziendali rappresenta un'efficace misura di protezione. Grazie alla velocità e alla flessibilità di XDP, le aziende possono adottare una difesa proattiva contro minacce sempre più sofisticate, garantendo prestazioni elevate e una maggiore sicurezza per le loro reti, alcuni esempi di applicazione industriale potrebbero essere:

Protezione delle Infrastrutture Critiche

In ambienti aziendali con server esposti a Internet, come data center, piattaforme cloud, o servizi online ad alta disponibilità, il codice XDP può essere utilizzato per prevenire attacchi hacker. La capacità di intercettare e bloccare i pacchetti sospetti a livello di rete, prima che raggiungano il protocollo TCP, rende lo snippet particolarmente utile per proteggere infrastrutture critiche senza aggiungere carico sui server.

Mitigazione a Basso Overhead

Un altro vantaggio del codice XDP è che opera nel livello più basso dello stack di rete, bypassando il percorso completo di rete del kernel. Ciò riduce il carico sul sistema e in contesti aziendali dove le prestazioni sono cruciali (ad esempio servizi di borsa, banche o e-commerce), questo permette di difendersi da attacchi DoS senza introdurre rallentamenti significativi. Le aziende possono mantenere l'operatività dei loro servizi garantendo uptime e reattività.

Ambienti Cloud e DevOps

Le aziende che gestiscono servizi su larga scala, come piattaforme SaaS (Software as a Service) o IaaS (Infrastructure as a Service), possono sfruttare

lo snippet XDP per proteggere i propri sistemi dal traffico malevolo in entrata. Il codice può essere implementato in ambienti containerizzati (ad esempio Kubernetes) o in architetture cloud-native, bloccando pacchetti SYN prima che raggiungano i servizi o i pod, migliorando così la resilienza del sistema.

Applicazione nei Firewall di Front-End

Le aziende che offrono soluzioni di firewall di rete potrebbero integrare lo snippet XDP nei loro sistemi per fornire una protezione avanzata contro gli attacchi DoS. XDP, operando a una velocità molto elevata, può essere utilizzato come livello di protezione "di front-end", bloccando traffico sospetto prima che entri nei firewall tradizionali o nei sistemi di intrusion detection/prevention (IDS/IPS).

Miglioramento della Sicurezza nei Settori Finanziari e di Telecomunicazioni

Le reti finanziarie e di telecomunicazioni sono particolarmente soggette a minacce informatiche. In questi settori, attacchi DoS possono compromettere servizi critici come piattaforme di trading o infrastrutture di telecomunicazioni. Implementare lo snippet XDP in questi contesti fornisce un livello aggiuntivo di sicurezza, prevenendo disservizi su larga scala e riducendo la possibilità di downtime.

Protezione dei Gateway IoT e Edge Computing

Con la crescita dell'Internet of Things (IoT) e dell'edge computing, il numero di endpoint vulnerabili ad attacchi Syn Flood è aumentato. Lo snippet XDP può essere utilizzato per proteggere gateway IoT o nodi edge, bloccando gli attacchi prima che possano saturare le risorse limitate di questi dispositivi, migliorando così la sicurezza dell'intera rete aziendale.

Servizi di CDN e Content Delivery

Aziende che offrono servizi di content delivery (CDN) possono trarre vantaggio dallo snippet XDP per proteggere i loro server edge da attacchi DoS. Bloccando il traffico malevolo a livello di XDP, la latenza si riduce, migliorando la disponibilità dei servizi di distribuzione contenuti e garantendo un'esperienza utente più fluida anche durante tentativi di attacco.

Bibliografia

[1] CVE-2021-3493, . URL https://ubuntu.com/security/CVE-2021-3493.

[2] NVD - CVE-2021-3560, . URL https://nvd.nist.gov/vuln/detail/CVE-2021-3560.

[3] NVD - CVE-2022-26352, . URL https://nvd.nist.gov/vuln/detail/CVE-2022-26352.

[4] NVD - CVE-2022-48703, . URL https://nvd.nist.gov/vuln/detail/CVE-2022-48703.

[5] NVD - CVE-2023-3446, . URL https://nvd.nist.gov/vuln/detail/CVE-2023-3446.

[6] NVD - CVE-2023-39325, . URL https://nvd.nist.gov/vuln/detail/CVE-2023-39325.

[7] NVD - CVE-2023-50269, . URL https://nvd.nist.gov/vuln/detail/CVE-2023-50269.

[8] NVD - CVE-2023-51385, . URL https://nvd.nist.gov/vuln/detail/CVE-2023-51385.

[9] NVD - CVE-2024-25617, . URL https://nvd.nist.gov/vuln/detail/CVE-2024-25617.

[10] NVD - CVE-2024-29040, . URL https://nvd.nist.gov/vuln/detail/CVE-2024-29040.

[11] Linux eBPF Tracing Tools. URL https://www.brendangregg.com/ebpf.html.

[12] ply. URL https://wkz.github.io/ply/.

[13] System call, accessi e modifiche ai file con auditd | EXTRAORDY - La Formazione ufficiale Red Hat. URL https://www.extraordy.com/auditd/.

[14] bpf-helpers(7) - Linux manual page. URL https://man7.org/linux/man-pages/man7/bpf-helpers.7.html.

[15] eBPF - Introduction, Tutorials & Community Resources. URL https://ebpf.io/what-is-ebpf/.

[16] hping3 | Kali Linux Tools. URL https://www.kali.org/tools/hping3/.

[17] Kernel Korner. URL https://dl.acm.org/doi/fullHtml/10.5555/1103050.1103058.

[18] netfilter/iptables project homepage - The netfilter.org project. URL https://www.netfilter.org/.

[19] Scapy. URL https://scapy.net/.

[20] iovisor/bcc, Oct. 2024. URL https://github.com/iovisor/bcc.

[21] Che cos'è l'eBPF? | IBM, May 2024. URL https://www.ibm.com/it-it/topics/ebpf.

[22] amadabhu. Configure Linux system auditing with auditd, Oct. 2021. URL https://www.redhat.com/sysadmin/configure-linux-auditing-auditd.

[23] F. Risso and M. Tumolo. Towards a faster Iptables in eBPF.

[24] A. Tolkachova, D. Zhuravchak, A. Piskozub, and V. Dudykevych. *Monitoring ransomware with Berkeley Packet Filter (BPF)*. Nov. 2023.

[25] H. V. Wieren. Signature-Based DDoS Attack Mitigation: Automated Generating Rules for Extended Berkeley Packet Filter and Express Data Path.

[26] M. I. Z. Zam. Auditd: Rule Writing for better Threat Detection on *nix Devices. 2021. doi: 10.13140/RG.2.2.16085.76003. URL `http://rgdoi.net/10.13140/RG.2.2.16085.76003`.