eBPF: la sicurezza, e anche l’attacco, si spostano nel kernel



eBPF è la tecnologia che ha reso il cuore di Linux programmabile senza doverlo riscrivere, e nel farlo ha spostato il baricentro della sicurezza dove finora era difficile arrivare: dentro il kernel. La sigla sta per *extended Berkeley Packet Filter*, e descrive la capacità di eseguire piccoli programmi in uno spazio protetto del sistema operativo, senza modificare il codice del kernel, senza caricare moduli e senza riavviare la macchina. Nato per filtrare pacchetti di rete, eBPF è diventato una macchina virtuale generalista che oggi serve per la rete, l’osservabilità e, sempre più, la difesa.

Il motivo per cui interessa chi si occupa di sicurezza è una questione di posizione. Un programma eBPF vede ogni *system call*, ogni esecuzione di processo, ogni accesso a un file e ogni pacchetto di rete prima che qualunque applicazione o container li elabori. È il punto di osservazione più vicino alla verità che un difensore possa avere, e per giunta a costo bassissimo. La stessa posizione, però, è preziosa anche per chi attacca, ed è questa ambivalenza a rendere eBPF uno dei temi più rilevanti, e meno raccontati, della sicurezza dei sistemi.

Cos’è eBPF, e perché il kernel è il posto giusto

Il problema storico di chi voleva sorvegliare un sistema dall’interno era il rischio: un errore in un modulo del kernel manda in crash l’intera macchina. eBPF aggira questo pericolo

con un meccanismo che è la sua vera innovazione, il verificatore. Prima di lasciar girare un programma, il kernel ne dimostra la sicurezza: i cicli devono essere limitati, così da garantire che il programma termini in un numero prevedibile di istruzioni, e ogni accesso alla memoria viene validato. Un programma che non supera la verifica non viene eseguito. È questa garanzia a permettere di iniettare logica nel kernel senza trasformarlo in una mina.

Da qui nasce il vantaggio per la sicurezza, descritto bene dalla documentazione ufficiale del progetto su cosa sia eBPF. Raccogliere dati su processi, file e rete direttamente dal kernel significa avere una visibilità completa su tutto ciò che accade, senza la frammentazione e i punti ciechi degli strumenti che lavorano nello spazio utente. Significa anche un dato più difficile da manomettere: un *malware* che inganna o disattiva un agente di *logging* in spazio utente ha vita molto più dura contro un sensore che osserva dal kernel. E significa basso impatto, perché la raccolta avviene là dove gli eventi nascono, senza i costi di intermediazione che gravano sugli approcci tradizionali.

Dalla visibilità all'enforcement: Falco e Tetragon

Su questa base è cresciuto un ecosistema di strumenti che ha cambiato il modo di fare sicurezza a runtime, soprattutto in ambienti containerizzati. Il capostipite è Falco, progetto della CNCF che ha aperto la strada al rilevamento comportamentale basato su eBPF: osserva gli eventi del kernel e segnala le anomalie sulla base di una vasta libreria di regole, lavorando soprattutto sul versante del rilevamento. È il sensore che dice al centro operativo di sicurezza cosa sta accadendo nel momento in cui accade.

Il passo successivo è non limitarsi a vedere, ma intervenire. Tetragon, progetto CNCF nato all'interno di Cilium, porta la

logica di sicurezza dentro i programmi eBPF stessi, e questo gli consente non solo di rilevare ma di applicare: può accorgersi di un comportamento ostile e terminare il processo che lo sta eseguendo prima che la *system call* si completi, con un impatto contenuto anche sotto grandi volumi di eventi. La differenza tra rilevare e bloccare nel kernel non è cosmetica: è la distanza tra sapere che qualcosa è andato storto e impedire che vada a termine. È anche la ragione per cui la difesa cloud-native si sta spostando dalla sola visibilità alla prevenzione in tempo reale, e per cui i pesanti agenti a *sidecar* di un tempo cedono il passo a sensori nativi del kernel ben più leggeri.

eBPF è un'arma a doppio taglio

Qui arriva la parte che si tende a non raccontare. Lo stesso potere che rende eBPF prezioso per i difensori lo rende attraente per gli attaccanti, e la frontiera del kernel è terreno conteso. Una nuova generazione di *rootkit* sfrutta eBPF per agganciare le *system call* e intercettare gli eventi del kernel senza caricare un modulo tradizionale, l'artefatto che gli strumenti di difesa hanno sempre cercato. Con programmi eBPF malevoli un attaccante può nascondere processi, socket e file, sottrarre credenziali e manomettere la telemetria che dovrebbe tradirlo, restando invisibile a chi cerca solo moduli sospetti.

Non è teoria. Famiglie come BPFDoor e Symbiote, emerse dal 2021, sono state tra le prime a usare questa tecnologia per canali di comando e controllo furtivi a livello di kernel, e analisi più recenti come quella del *rootkit* LinkPro mostrano una sofisticazione crescente. Le ricerche dei vendor sulle capacità offensive di eBPF descrivono un repertorio ormai maturo, dall'intercettazione delle *system call* alla persistenza nascosta. Il paradosso è netto: il miglior punto di osservazione per difendere un sistema è anche il miglior nascondiglio per chi lo ha compromesso.

Monitorare chi monitora il kernel

La conseguenza pratica è che eBPF aggiunge una nuova superficie da sorvegliare, e cambia la domanda di sicurezza. Non basta più chiedersi cosa accade nel sistema: bisogna chiedersi anche quali programmi eBPF sono caricati, da chi e con quali permessi. La difesa matura monitora l'uso stesso di eBPF, controlla chi ha la capacità di caricare programmi nel kernel e tratta quel privilegio con la stessa cura riservata agli accessi più sensibili. La tecnologia che dà la visibilità più profonda richiede, per non ritorcersi contro, una governance altrettanto profonda di chi può usarla.

Perché conta adesso

La spinta non è teorica, è operativa. Negli ambienti cloud-native, dove i carichi sono effimeri e gli attacchi sempre più spesso non lasciano *malware* da riconoscere, la capacità di osservare il comportamento reale a runtime è diventata decisiva, e eBPF è la tecnologia che la rende praticabile su larga scala e a basso costo. Un worm cloud-native che si muove tra container compromessi lascia tracce comportamentali che un sensore nel kernel può cogliere anche quando non esiste un file malevolo da analizzare. È la risposta tecnica all'evoluzione degli attacchi verso l'uso di strumenti e credenziali legittimi.

Il quadro che ne emerge ridefinisce il confine della sicurezza dei sistemi. Per anni il kernel è stato una scatola quasi opaca, difficile da osservare e rischiosa da modificare. eBPF lo ha aperto, dando ai difensori una visibilità e una capacità di intervento prima impensabili, e dando agli attaccanti un nuovo livello dove operare e nascondersi. La domanda non è più se usare eBPF, perché lo useranno comunque entrambe le parti, ma chi controlla cosa gira nel kernel. In quello spazio, prima inaccessibile e ora programmabile, si sta giocando una parte crescente della partita tra difesa e attacco, e ignorarlo non

è più un’opzione per chi prende sul serio la sicurezza dei propri sistemi.

Condividi sui Social Network:
https://www.ictsecuritymagazine.com/cyber-security/ebpf-sicurezza/