
OWASP 2024
AppSec DAYS
SINGAPORE
TM

OWASP 2024
AppSec DAYS
SINGAPORE

OCT 1-2 2024
OCT 1 TRAINING
OCT 2 CONFERENCE

OWASP 2024
AppSec DAYS
SINGAPORE

OCT 1-2 2024
OCT 1 TRAINING
OCT 2 CONFERENCE

Enhancing Kubernetes Security: File Integrity Monitoring with eBPF

Abhijit Chatterjee

What is eBPF (extended Berkeley Packet Filter)?

A technology for safely running custom programs within the Linux kernel.

Allows monitoring and responding to events like file access, network packets, system calls, and more.

Provides real time visibility.

Dynamically attach programs without changing kernel code or rebooting the system.

It was released in version 3.18 of the Linux kernel.

History and evolution of eBPF

1992: Originl BPF (Berkeley Packet Filter (BPF)) was introduced.

2014: eBPF Introduced: Extended BPF (eBPF) introduced in the Linux kernel, enhancing original BPF capabilities and expanding its functionality beyond packet filtering.

2015-2016: Enhanced Observability: eBPF begins to support tracing and observability; added features allow users to attach programs to kernel events (e.g., system calls, tracepoints).

2017: Performance Monitoring: Adoption of eBPF grows for performance analysis, allowing real-time insights into system performance without significant overhead.

2018: Security Use Cases: being utilized for intrusion detection and network security policies.

2019: Cloud-Native Adoption: eBPF becomes more prominent in cloud-native environments like Kubernetes for container-level monitoring and networking (e.g., Cilium project).

2020-Present: Advanced Applications: eBPF supports advanced observability, security, and networking functionalities, gaining adoption across major platforms and use cases in performance tuning, real-time security monitoring, and traffic control.

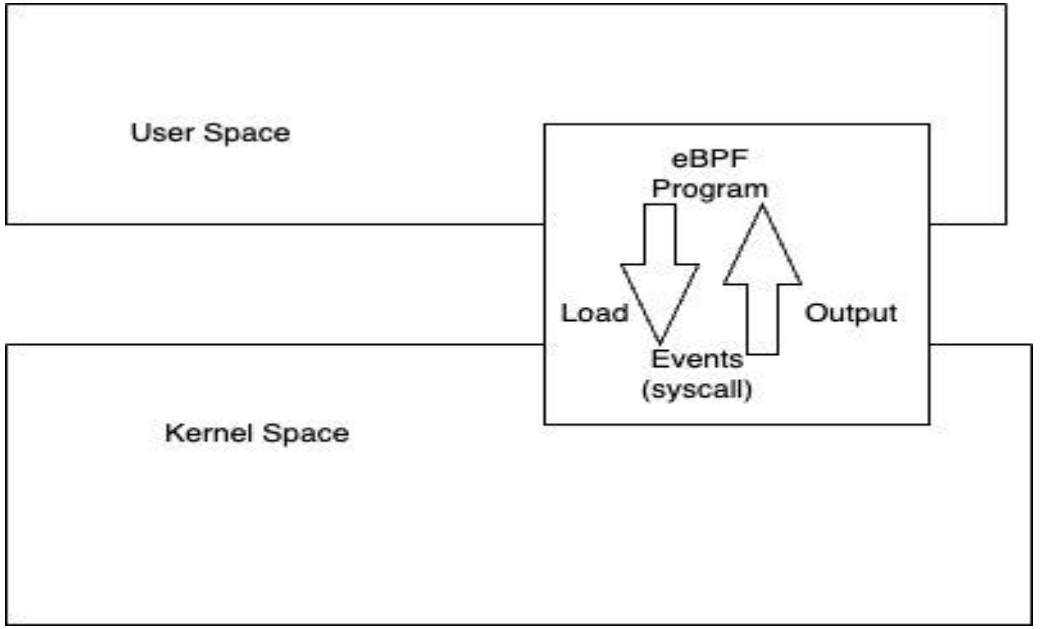
How eBPF works

eBPF Programs Hook into Kernel Events - can be dynamically inserted into various points in the Linux kernel.

Execution in the Kernel Space - run safely in the kernel space with minimal overhead.
eBPF Maps for Data storage - Data is stored in "maps," which are efficient, which are efficient key-value stores.

User-Space and Kernel-Space Communication - User-space applications can load and read data from eBPF maps.

eBPF Diagram



What is Linux System Calls?

It is programmatic way in which a program requests a service from the operating system.

A program from user space trying to access a web page triggers an event in the Kernel.

tracepoint/syscalls/sys_enter_fchownat
tracepoint/syscalls/sys_enter_unlinkat

What is FIM ?

A security process that tracks and detects changes to files, directories, and system configurations.
Identify unauthorized or unexpected modifications of files.
Security layer to prevent data breaches, malware, and other potential threats to system integrity.
Example :- auditd, Tripwire etc.

Traditional File Integrity Monitoring Challenges

High Performance Overhead - constant scanning and resource intensive logging.
Lack of Real-Time Analysis - Work in a batch which means periodically scan files.
Complexity in Dynamic Environments - Not suitable for containerized environment such as Kubernetes.

Traditional FIM Limitation in Kubernetes Environment

Ephermal workloads - Short lived containers. They can be spun up or down in seconds based on demand, failures, or updates. Traditional FIM tools that require manual configuration or agents installed on each node can struggle to keep up with these frequent changes.

Multi Tenant Environments - Monitor without exposing sensitive information across tenants is complex.

Resource Constraints and Scalability - Containers are often with limited CPU and memory resources. Resource heavy FIM agent can impact container performance.

Lack of Granular Visibility - A host can have multiple containers running. Traditional FIM tools often focus on the host level checking.

Deploying eBPF for FIM in Kubernetes

Hosts should be having the Linux Kernel 3.18 or above.

eBPF-based FIM solution can be deployed by using Daemonset or sidecar containers.

Linux headers installation in the host - require to compile the eBPF program.

Security considerations - eBPF Requires Privileged Access. It is risky as compromised users could use eBPF to gain elevated access to the kernel.

Complexity of eBPF Programs - eBPF programs requires in-depth knowledge of kernel behavior. There is a risk of creating programs that unintentionally open security holes or cause system instability.

Setup RBAC policy for a secure eBPF-based File Integrity Monitoring (FIM) deployment

Dedicated Service Account

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: ebpf-service-account
  namespace: kubefim-ns
```

Restricted Role

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: ebpf-privileged-role
rules:
  - apiGroups: [""]
    resources: ["pods"]
    verbs: ["get", "list", "watch", "create", "delete"]
  - apiGroups: [""]
    resources: ["pods/exec", "pods/log"]
    verbs: ["get", "create"]
  - apiGroups: [""]
    resources: ["nodes"]
    verbs: ["get", "list"]
```

ClusterRoleBinding

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: ebpf-privileged-binding
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: ebpf-privileged-role
subjects:
  - kind: ServiceAccount
    name: ebpf-service-account
    namespace: kubefim-ns
```

DaemonSets Vs Sidecar

A DaemonSet ensures that all Nodes run a copy of a Pod.

Sidecar containers are the secondary containers that run along with the main application container within the same Pod.

Init Containers

Containers that run before app containers in a Pod.

It helps to install the system dependencies that the application container requires.

If init containers fails, the kubelet repeatedly restarts that init container until it succeed.

Demo

OWASP KubeFIM

https://github.com/abhijitio/OWASP-KubeFIM/tree/main/kubernetes_deployment


OWASP 2024
AppSec DAYS
SINGAPORE
THANK
YOU!