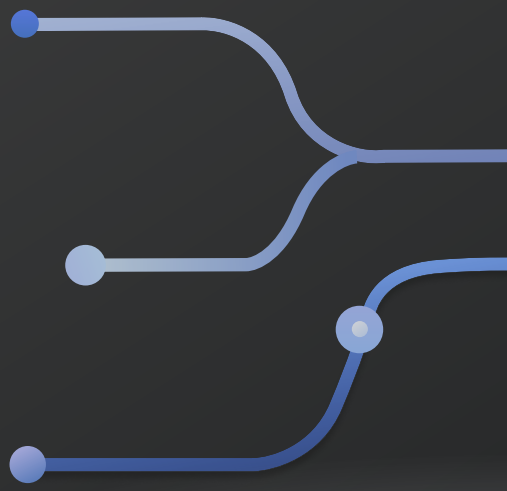
eBPF: The Future of API Security and Observability

This solutions brief explains and explores how extended Berkeley Packet Filter (eBPF) works for API observability, and how eBPF can unlock deep application and API insight.

Introduction

A new shift has occurred within the software development industry, one that hasn't been addressed fully by the API security tools available today. Cloud-native, distributed, and API-based applications create new risks and new opportunities for hackers to steal data and attack your system.



As applications become more distributed, you no longer have visibility or the necessary understanding of what is happening across your environment. Monolithic applications stored their logs in one place. On the other hand, each microservice has its own monitoring and logging capabilities. Monitoring each service individually isn't a problem, but learning global system behavior is. Blind spots within the interaction between services could develop due to data processing limitations imposed by most legacy systems.

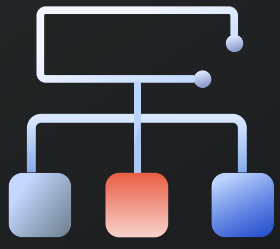
This specifically brings the added challenge of knowing what kinds of data to analyze that allows security teams to make better decisions in protecting their APIs. There are literally thousands of data points that each give relevant information and context, and are needed to create a holistic view of your API attack surface.

This paper is intended for security professionals who want a deeper understanding of how Traceable collects data for context-aware API security.

Traceable is the only API security vendor that provides an eBPF-based solution for capturing API security-related data from application environments.

What is eBPF?

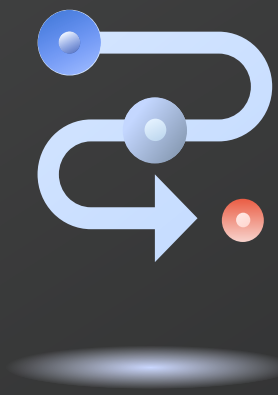
eBPF (extended Berkeley Packet Filter) is a kernel feature that has been shipped with Linux Kernels since 2014, the same year the first Kubernetes commit was made [1, 2, 3]. Unlike most developer code that gets written in user space, using eBPF requires writing code in the kernel space which brings distinct advantages in terms of performance and resource consumption.



eBPF is very popular with teams that need to operate in high-performance environments. For example, Netflix has about 15 eBPF programs running on every server instance, Facebook, in contrast, has about 40 eBPF programs that are active on every server with another 100 eBPF programs that get spawned and killed as needed [4].

eBPF: How it works

eBPF provides a low-level event-driven framework that enables programmers to run code in the kernel's privileged context & capture how the kernel reacts to defined triggers like network events, system calls, function entries, and kernel tracepoints. Effectively, this means that eBPF enables the ability for user space programs to read and react to data from the actions of the kernel.



eBPF ensures the safety of the kernel and other processes running on it by requiring authorization and validation before it runs programs in a kernel sandbox. eBPF is native in all modern-day Linux kernels and is also available in Windows, so the framework is already widespread, especially in cloud-based applications.

Applications in the industry

eBPF was initially used as a way to increase observability and security when filtering network packets. However, over time, it has become a way to make the implementation of user-supplied code safer, more convenient, and better performing. eBPF has become increasingly popular and is now used for many applications.

Big cloud companies like Netflix, Facebook, AWS, Google, and Microsoft are providing new cloud capabilities and tools using eBPF. And there are several new management tools from companies such as Isovalent's Cilium for cloud-native networking, security, and observability, New Relic's Pixie for Kubernetes observability, and Meta's Katran, a high performance layer 4 load balancer. Examples of other projects using eBPF can be found on the eBPF site at https://ebpf.io/projects.

How Traceable uses eBPF

Traceable is the only API security vendor that provides an eBPF-based solution for capturing API security-related data from application environments. Due to its agentless nature and low resource requirements, it's a popular option with our customers in production, especially in industries such as FinTech that have high-performance requirements.

The eBPF-based data collection has the ability to show deep API traffic data (request/response headers and bodies/payloads) for both North-South and East-West traffic. This data collection is out-of-band, non-invasive, and fast and highly efficient because it's running at the kernel level. What's more, this high eBPF efficiency, combined with Traceable technology, results in a near-zero overhead (< 1 ms latency difference) on instrumented applications.

The data collected by the eBPF instrumentation is processed by the same Traceable platform (AI/ML, data processing, analysis, etc) as the data from all the other Traceable data collectors, enabling the same set of capabilities: API discovery and risk management; API protection; and API security analytics.

Traceable is the only API security vendor that provides an eBPF-based solution for capturing API security-related data from application environments.

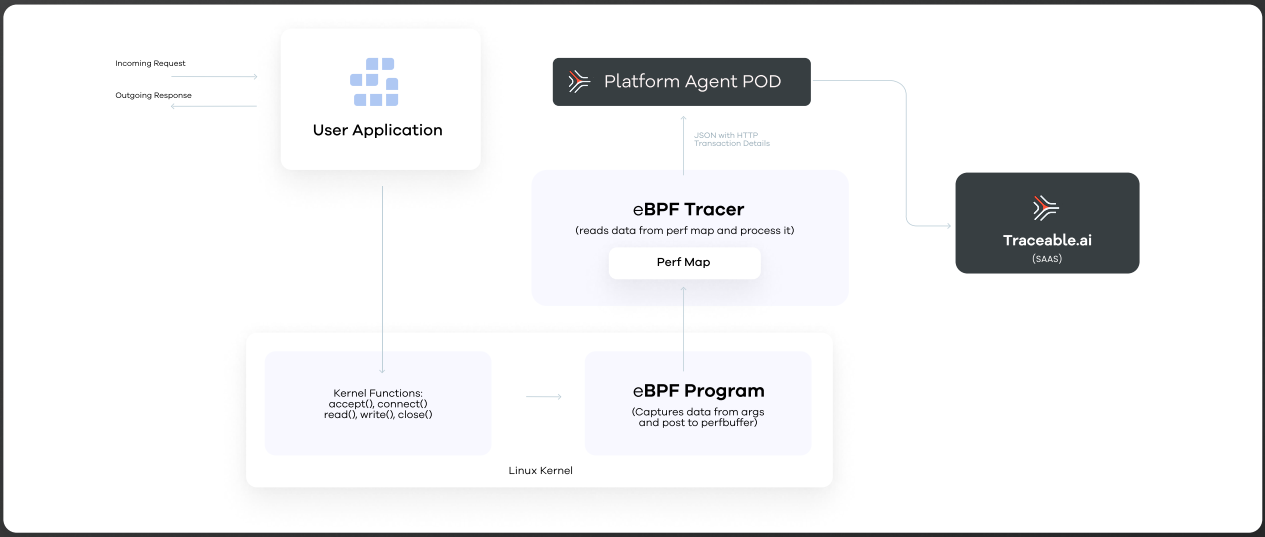
eBPF: How it works

Traceable eBPF instrumentation is one of our agentless deployment options that can be deployed via helm charts/scripts as Kubernetes daemonsets. It attaches probes to kernel functions and collects the data on function execution. The types of functions the probes are attached to are network socket transactions like open, connect, read, write, and close calls.

The eBPF deployment can be configured to capture data from all or specific pods/containers, based on deployment time configuration (annotations). For instance, Traceable can choose to capture data only from nginx-ingress containers, or from backend services. Traceable can additionally choose to collect ingress or egress data depending on whether the container in question is a gateway or backend service.

Unlike pod mirroring solutions or sidecar-based data capture solutions, which need to be deployed in each pod, ebpf-based daemonset can be run 1 pod per node, thereby significantly reducing the resource footprint across the cluster. For the same reason, it also reduces network chattiness.

The following diagram shows a high-level flow of how Traceable's eBPF-based data collection works:



Where is this headed?

Traceable is continuing our leadership in leveraging eBPF to increase the effectiveness and efficiency of API security. We are continuing to work closely with our customers using eBPF in their production and pre-production environments to continue to push the envelope on how eBPF can help improve their API security posture.

In general, as eBPF continues to become more and more mainstream, and the industry learns more about how to create value on top of it, the list of innovations will continue to grow. With eBPF, the future of infrastructure, application, and security management will be very exciting.