Chaos Engineering in the Fast Lane: Accelerating Resilience with AI and eBPF

February 2024

Francesco Sbaraglia
Michele Dodič

Speakers

Francesco Sbaraglia

SRE Tech Lead ASG
AIOPS & Observability Lead EMEA
Accenture

Michele Dodič

SRE Co-Lead ASG
AIOPS & Observability SME
Accenture

Speakers

Francesco Sbaraglia

SRE Tech Lead ASG
AIOPS & Observability Lead EMEA
Accenture

Speakers

Michele Dodič

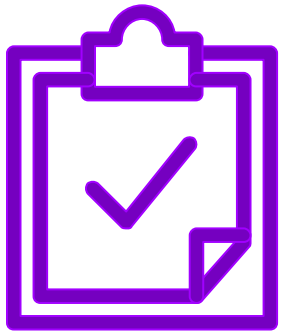
SRE Co-Lead ASG
AIOPS & Observability SME
Accenture

Agenda

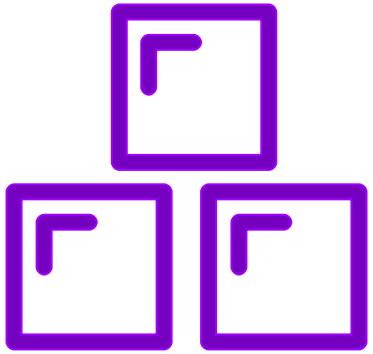
1. The current state of Chaos Engineering
2. Augmenting Chaos Engineering with eBPF
3. Augmenting Chaos Engineering with AI
4. Target Architecture & Live Demo
5. Conclusion & Takeaways

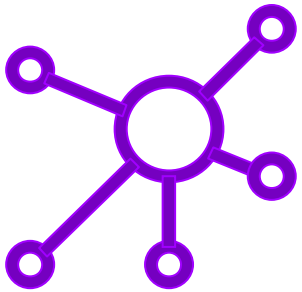
The Current State of Chaos Engineering

Lessons Learned – Key Benefits for SREs and Platform Engineers

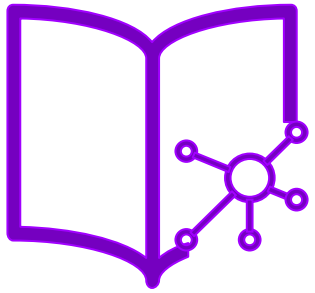


Bring up to speed new SRE resources

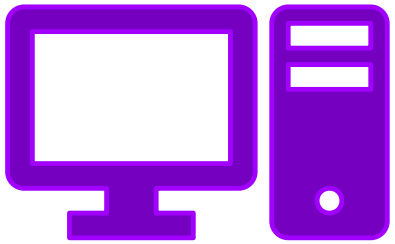




Improve communication inside the SRE team



Discover blind spots and reduce unknown behaviours



Improve MTTD/MTTR using new technologies

The Current State of Chaos Engineering

Lessons Learned – Key Challenges & Trends

CHALLENGES	HOW WE SOLVED IT
Complex connectivity troubleshooting of distributed Kubernetes clusters	?
Better visibility and control during chaos experimentation	?
Design enhanced security chaos experiments within Kubernetes	?
Getting started with the very first Chaos Engineering Experiment	?

The Current State of Chaos Engineering

Lessons Learned – Key Challenges & Trends

CHALLENGES	HOW WE SOLVED IT
Complex connectivity troubleshooting of distributed Kubernetes clusters	?
Better visibility and control during chaos experimentation	?
Design enhanced security chaos experiments within Kubernetes	?
Getting started with the very first Chaos Engineering Experiment	?

The Current State of Chaos Engineering

Lessons Learned – Key Challenges & Trends

CHALLENGES	HOW WE SOLVED IT
Complex connectivity troubleshooting of distributed Kubernetes clusters	Introduction of eBPF
Better visibility and control during chaos experimentation	?
Design enhanced security chaos experiments within Kubernetes	?
Getting started with the very first Chaos Engineering Experiment	?

source: *https://pixabay.com/photos/honey-bees-insects-hive-bee-hive-401238/*

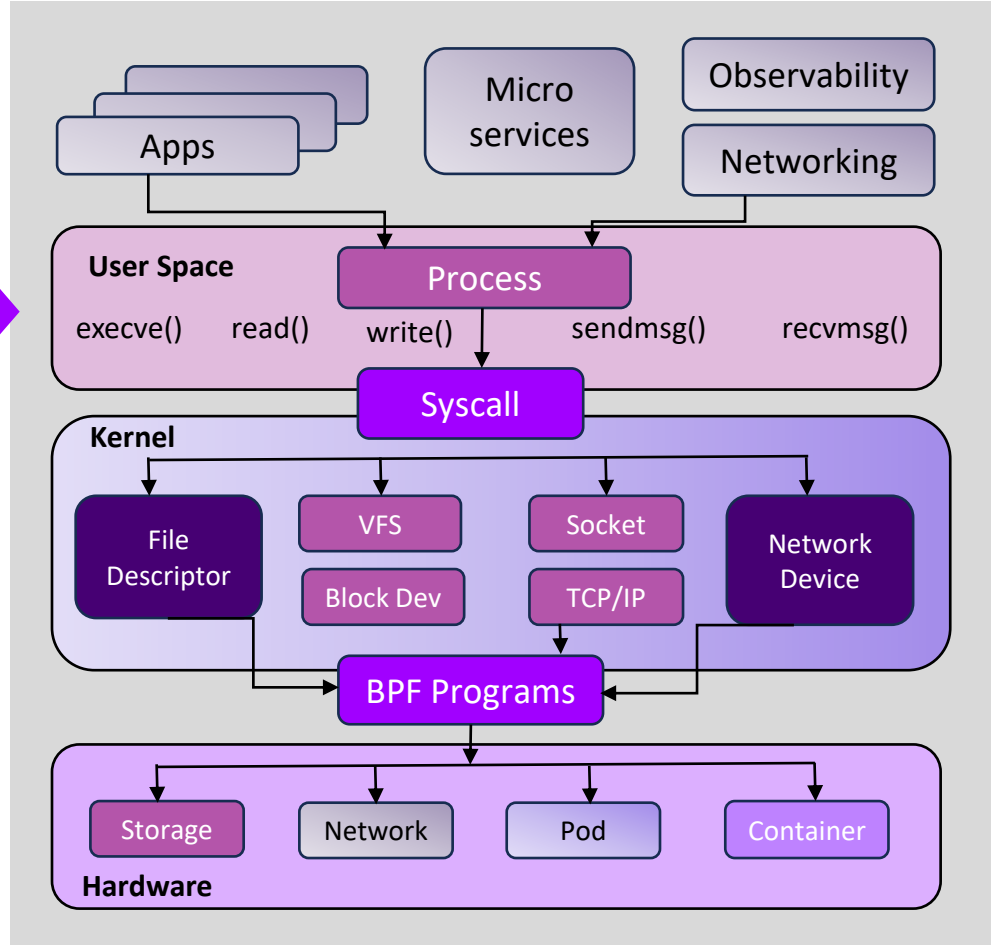
What is eBPF?

„eBPF, which stands for extended Berkeley Packet Filter, is an extraordinary technology with origins in the Linux Kernel, that can run sandboxed programs in a privileged context."

source: *https://ebpf.io/what-is-ebpf/*

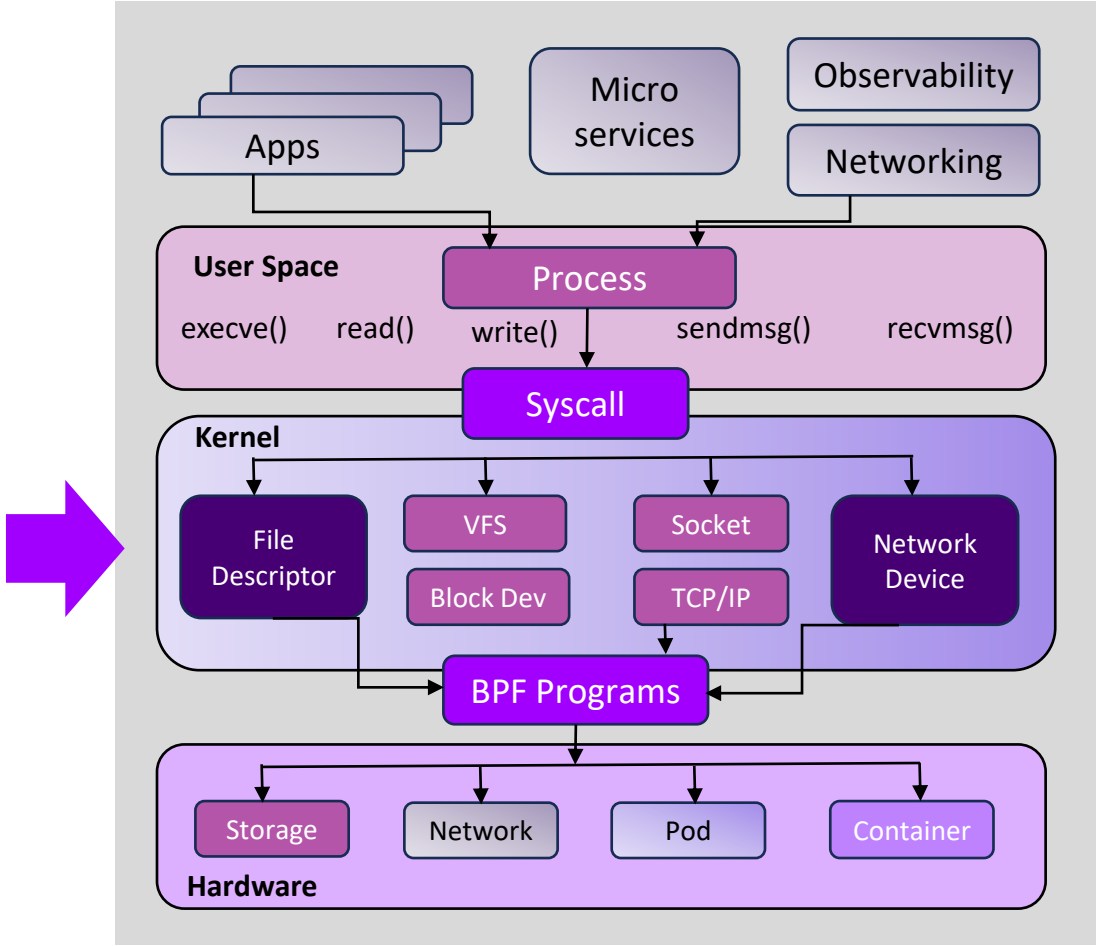
How does eBPF work? (I)

eBPF makes the kernel programmable



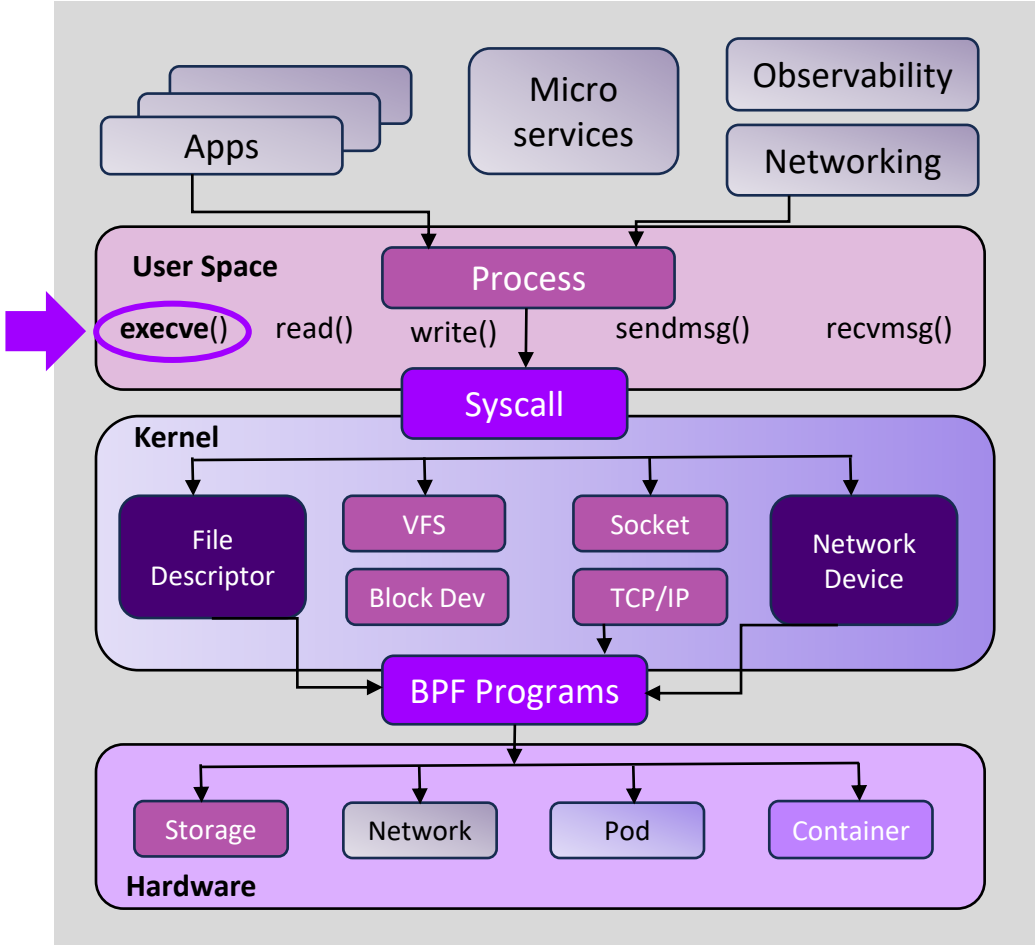
How does eBPF work? (I)

eBPF makes the kernel programmable



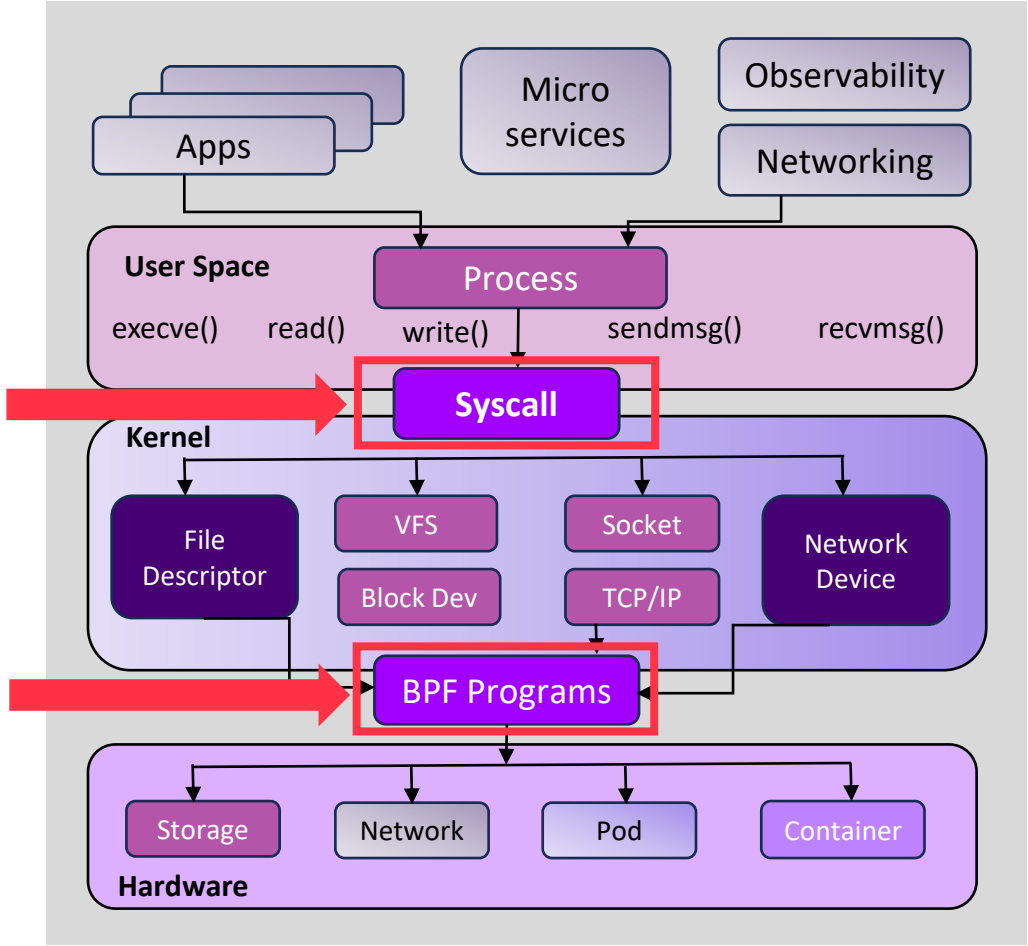
How does eBPF work? (I)

eBPF makes the kernel programmable



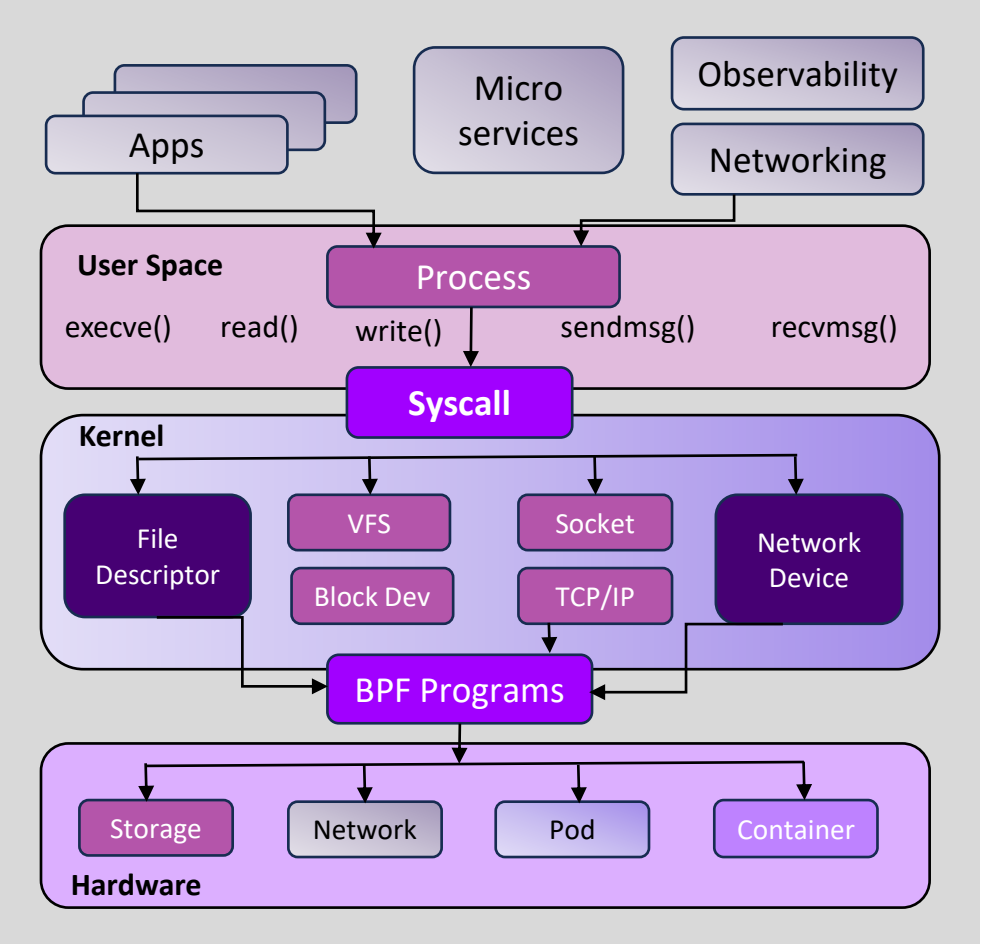
How does eBPF work? (I)

eBPF makes the kernel programmable

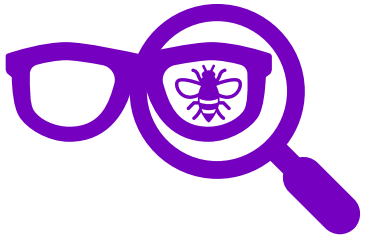


How does eBPF work? (I)

eBPF makes the kernel programmable



- **eBPF** tools instrument the system **without any app or config changes**
- The **kernel** is with eBPF like the **Big Brother** now. It sees everything!



How does eBPF work? (II)

How we technically load the eBPF program into the kernel?

Python code that compiles my eBPF program:

```
#!/usr/bin/python3
from bcc import BPF

program = eBPF_PROGRAM

b = BPF(text=program)
syscall = b.get_syscall_fnname("execve")
b.attach_kprobe(event=syscall, fn_name="hello")

b.trace_print()
```

source: *https://ebpf.io/what-is-ebpf/*

eBPF Hello World

```
int hello(void *ctx)
{
   bpf_printk("I'm alive!");
   return 0;
}
```

Info about process that called execvesyscall

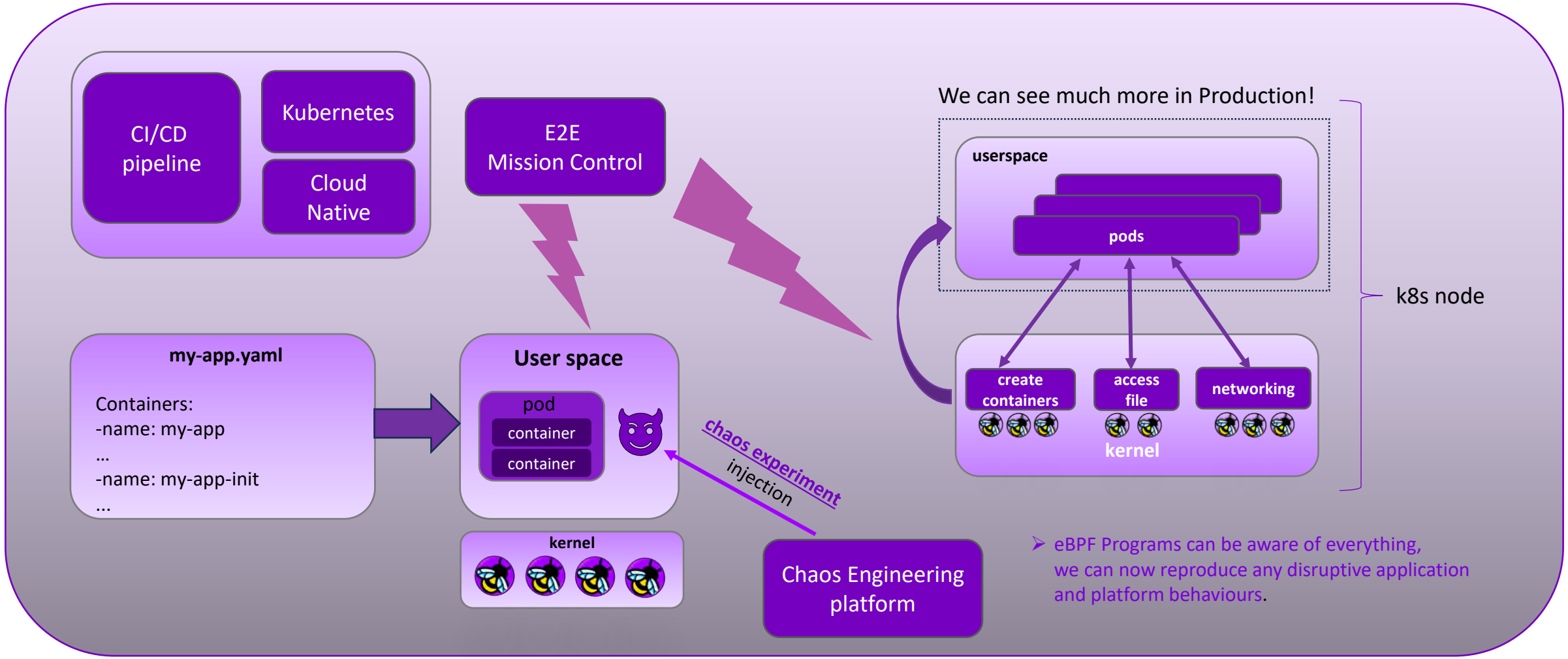
```
$ sudo ./hello
   bash-20241   [004] d... 84210.752785: 0: I'm alive!
   bash-20242   [004] d... 84216.321993: 0: I'm alive!
   bash-20243   [004] d... 84225.858880: 0: I'm alive!
```

source: *https://ebpf.io/what-is-ebpf/*

Every time a new program runs on this virtual machine, the `hello()`eBPF program will be triggered.

Augmenting Chaos Engineering with eBPF

Improved Chaos Experimentation observability: eBPF programs can be aware of everything



- eBPF Programs can be aware of everything, we can now reproduce any disruptive application and platform behaviours.

The Current State of Chaos Engineering

Lessons Learned – Key Challenges & Trends

CHALLENGES	HOW WE SOLVED IT
☑ Complex connectivity troubleshooting of distributed Kubernetes clusters	Introduction of eBPF
Better visibility and control during chaos experimentation	?
Design enhanced security chaos experiments within Kubernetes	?
Getting started with the very first Chaos Engineering Experiment	?

The Current State of Chaos Engineering

Lessons Learned – Key Challenges & Trends

CHALLENGES	HOW WE SOLVED IT
Complex connectivity troubleshooting of distributed Kubernetes clusters	Introduction of eBPF
Better visibility and control during chaos experimentation	?
Design enhanced security chaos experiments within Kubernetes	?
Getting started with the very first Chaos Engineering Experiment	?

The Current State of Chaos Engineering

Lessons Learned – Key Challenges & Trends

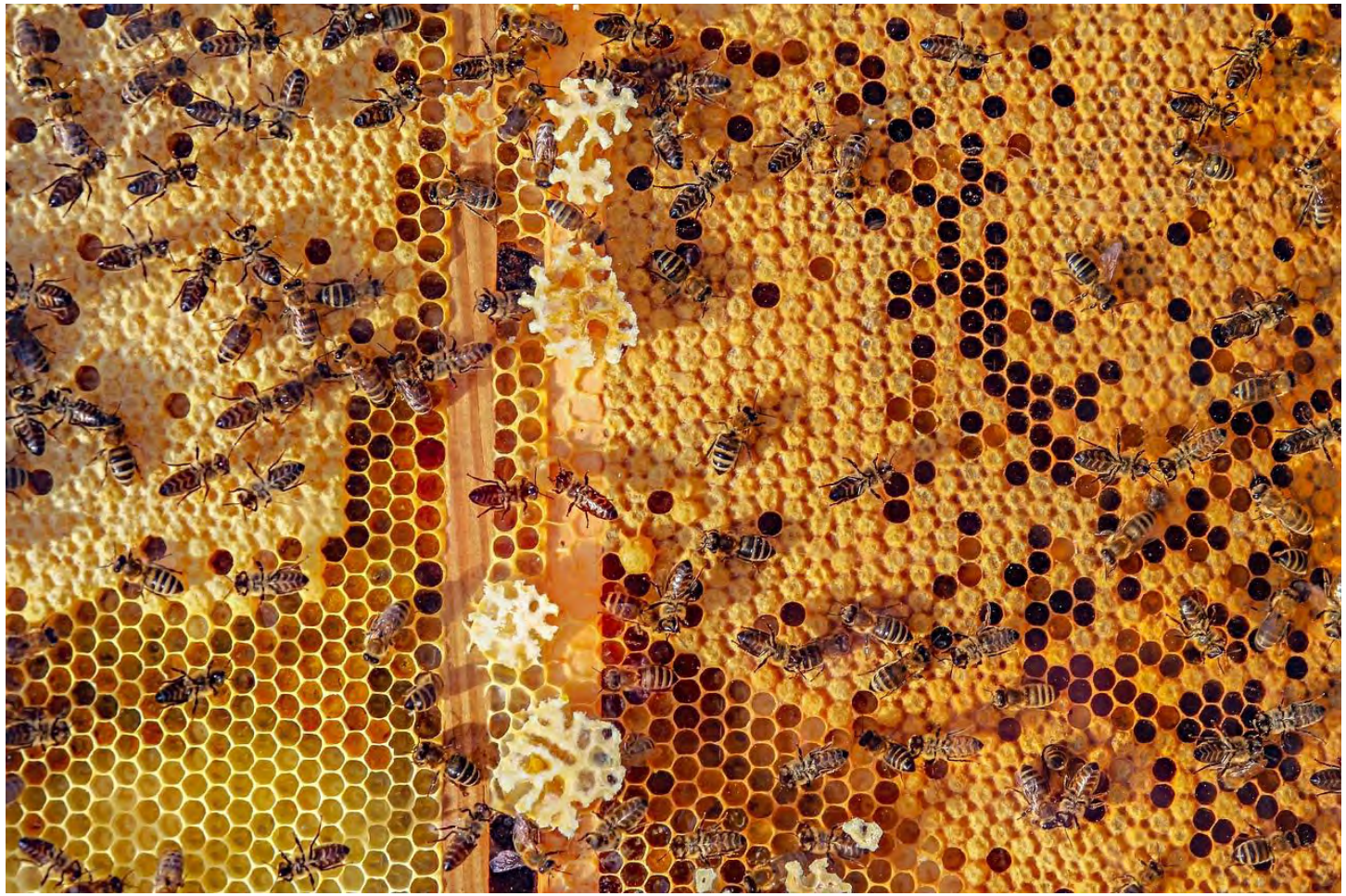
CHALLENGES	HOW WE SOLVED IT
Complex connectivity troubleshooting of distributed Kubernetes clusters	Introduction of eBPF
Better visibility and control during chaos experimentation	eBPF data can be leveraged to generate events and metrics, which can be used as the 'Big Red Stop Button'
Design enhanced security chaos experiments within Kubernetes	?
Getting started with the very first Chaos Engineering Experiment	?

Augmenting Chaos Engineering with eBPF

Improved security context-awareness on a more Cloud-Native level

Benefits?

- eBPF does **not** need any **app changes**
- eBPF can **see all activities** on the node
- eBPF is applied to enable **security observability**
- eBPF data can be used to generate **metrics & events**, which are used as input for our **AI** prediction



source: *https://pixabay.com/photos/bees-insects-macro-honey-bees-4126065/*

The Current State of Chaos Engineering

Lessons Learned – Key Challenges & Trends

	CHALLENGES	HOW WE SOLVED IT
☑	Complex connectivity troubleshooting of distributed Kubernetes clusters	Introduction of eBPF
☑	Better visibility and control during chaos experimentation	eBPF data can be leveraged to generate events and metrics, which can be used as the 'Big Red Stop Button'
	Design enhanced security chaos experiments within Kubernetes	?
	Getting started with the very first Chaos Engineering Experiment	?

The Current State of Chaos Engineering

Lessons Learned – Key Challenges & Trends

	CHALLENGES	HOW WE SOLVED IT
☑	Complex connectivity troubleshooting of distributed Kubernetes clusters	Introduction of eBPF
☑	Better visibility and control during chaos experimentation	eBPF data can be leveraged to generate events and metrics, which can be used as the 'Big Red Stop Button'
➡	Design enhanced security chaos experiments within Kubernetes	?
	Getting started with the very first Chaos Engineering Experiment	?

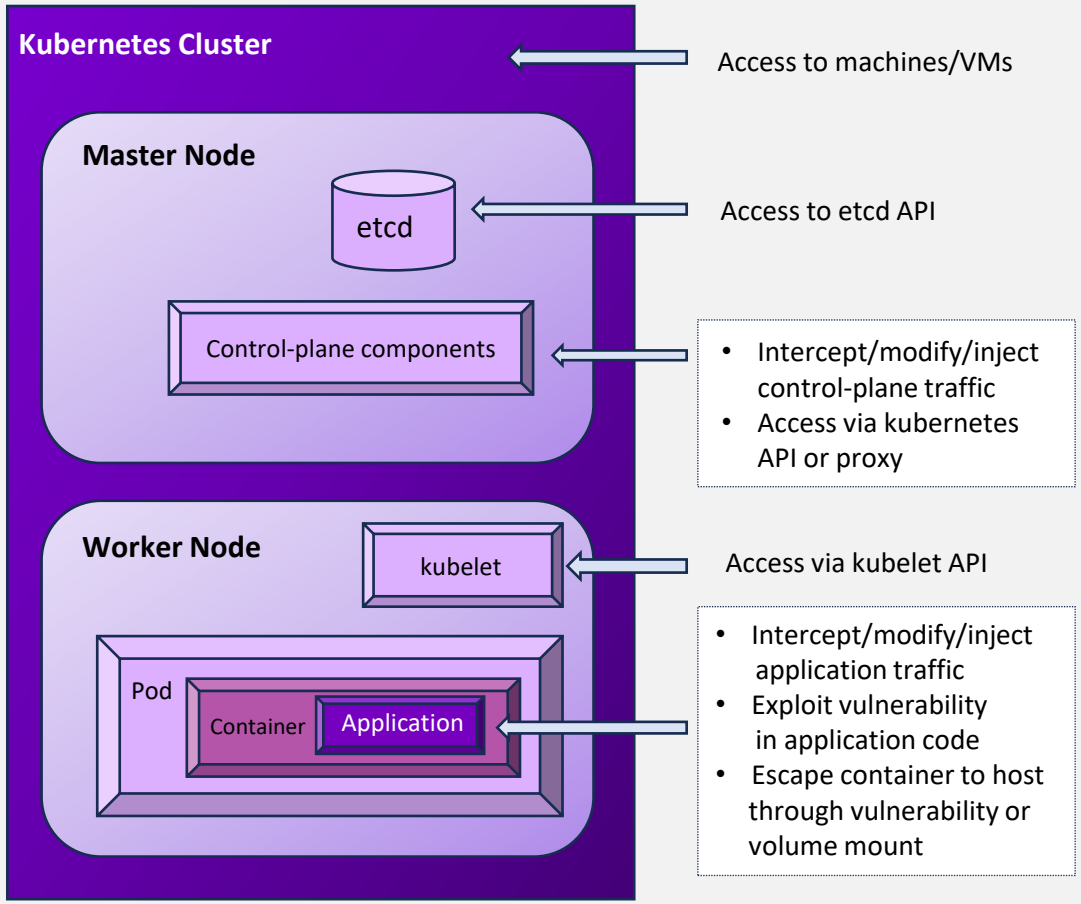
The Current State of Chaos Engineering

Lessons Learned – Key Challenges & Trends

	CHALLENGES	HOW WE SOLVED IT
☑	Complex connectivity troubleshooting of distributed Kubernetes clusters	Introduction of eBPF
☑	Better visibility and control during chaos experimentation	eBPF data can be leveraged to generate events and metrics, which can be used as the 'Big Red Stop Button'
➡	Design enhanced security chaos experiments within Kubernetes	eBPF helps to design new Security Chaos experiments
	Getting started with the very first Chaos Engineering Experiment	?

Potential security k8s vulnerabililties

Typical attacking surface



Access to machines/VMs

Access to etcd API

- Intercept/modify/inject control-plane traffic
- Access via kubernetes API or proxy

Access via kubelet API

- Intercept/modify/inject application traffic
- Exploit vulnerability in application code
- Escape container to host through vulnerability or volume mount

source: https://www.oreilly.com/library/view/kubernetes-security/9781492039075/ch01.html

What can we expect from a cyber attack ?

- Run another instance of kubelet on the same nodes
- Privileges pods
- Host network-enabled pod network direct attack
- Malicious webhook
- Pod network can be used for exploitation
- Using insecure root mounts
- Kubernetes cronjob/jobs
- Pod default tokens
- Kubernetes External-IPs

Top 3 vulnerabilities for **Kubernetes** attacks:

- Misconfigured Container
- **Unintentional Cluster Misconfiguration**
- Malicious Container Image

Advanced Chaos Experiments with eBPF

Leverage Cilium's Advanced Network Policy

Benefits

- Create better network Experiments
- Isolate pods
- Service Mesh Experiments
- Multi-Cluster Experiments

Future proof

- Node IO resource exhaustion
- Network resource exhaustion
- Pod resource exhaustion

Cilium EBPF

source: *https://cilium.io/use-cases/network-policy/*

eBPF removes the need to kill or delete the pod and to deploy any new tools.

The Current State of Chaos Engineering

Lessons Learned – Key Challenges & Trends

	CHALLENGES	HOW WE SOLVED IT
☑	Complex connectivity troubleshooting of distributed Kubernetes clusters	Introduction of eBPF
☑	Better visibility and control during chaos experimentation	eBPF data can be leveraged to generate events and metrics, which can be used as the 'Big Red Stop Button'
☑	Design enhanced security chaos experiments within Kubernetes	eBPF helps to design new Network and Security Chaos Experiments
	Getting started with the very first Chaos Engineering Experiment	?

The Current State of Chaos Engineering

Lessons Learned – Key Challenges & Trends

	CHALLENGES	HOW WE SOLVED IT
☑	Complex connectivity troubleshooting of distributed Kubernetes clusters	Introduction of eBPF
☑	Better visibility and control during chaos experimentation	eBPF data can be leveraged to generate events and metrics, which can be used as the 'Big Red Stop Button'
☑	Design enhanced security chaos experiments within Kubernetes	eBPF helps to design new Network and Security Chaos Experiments
➡	Getting started with the very first Chaos Engineering Experiment	?

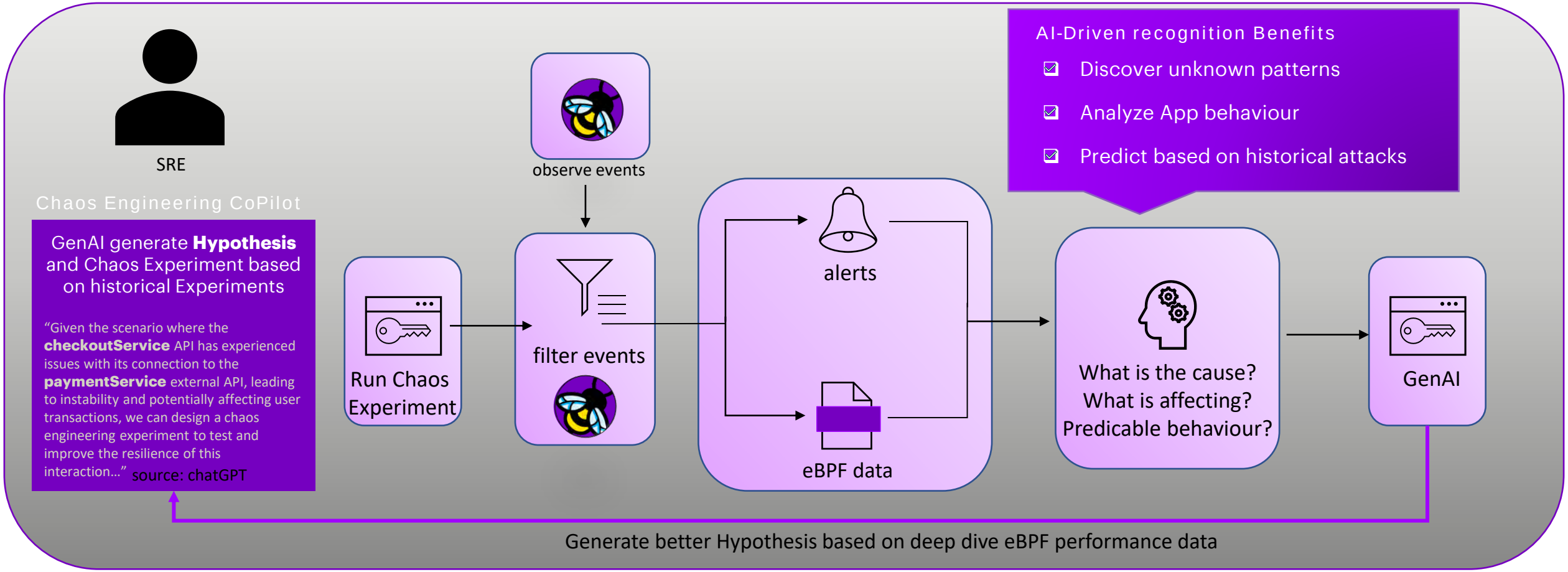
The Current State of Chaos Engineering

Lessons Learned – Key Challenges & Trends

	CHALLENGES	HOW WE SOLVED IT
☑	Complex connectivity troubleshooting of distributed Kubernetes clusters	Introduction of eBPF
☑	Better visibility and control during chaos experimentation	eBPF data can be leveraged to generate events and metrics, which can be used as the 'Big Red Stop Button'
☑	Design enhanced security chaos experiments within Kubernetes	eBPF helps to design new Network and Security Chaos Experiments
➡	Getting started with the very first Chaos Engineering Experiment	Chaos Engineering CoPilot powered by GenAI

Augmenting Chaos Engineering with AI and GenAI

eBPF goes beyond traditional networking in enhancing the Chaos Engineering landscape



The Current State of Chaos Engineering

Lessons Learned – Key Challenges & Trends

	CHALLENGES	HOW WE SOLVED IT
☑	Complex connectivity troubleshooting of distributed Kubernetes clusters	Introduction of eBPF
☑	Better visibility and control during chaos experimentation	eBPF data can be leveraged to generate events and metrics, which can be used as the 'Big Red Stop Button'
☑	Design enhanced security chaos experiments within Kubernetes	Cilium/eBPF helps to create Security Chaos experiments
☑	Getting started with the very first Chaos Engineering Experiment	Chaos Engineering CoPilot powered by GenAI

Demo: AI-drive Chaos Engineering Platform

Enhanced Target Architecture

fsbaraglia Create chaos_experiment_ai.py 8d72815 · now History

Code Blame 36 lines (27 loc) · 2.18 KB Code 55% faster with GitHub Copilot Raw

```python
import os

from openai import OpenAI

client = OpenAI(
  organization=os.environ.get("ORG_ID")
  # This is the default and can be omitted
  api_key=os.environ.get("OPENAI_API_KEY")
)

#Prompt Engineering #system role
messages = [{"role": "system", "content": "
            Chaos Engineering Expert with Kubernetes and Cloud Background
description: |
  This role embodies an expert specialized in chaos engineering within Kubernetes environments and cloud infrastructure. The individual possesses a comprehensive understanding of des

expertise:
  - Chaos Engineering Principles: Profound knowledge in chaos engineering concepts, methodologies, and best practices to assess and enhance system resilience.
  - Kubernetes: Advanced experience in deploying, managing, and troubleshooting Kubernetes clusters, including familiarity with Kubernetes architecture, ecosystem, and orchestration.
  - Cloud Platforms: Extensive experience with major cloud service providers (e.g., AWS, Azure, Google Cloud) and their managed Kubernetes services like EKS, AKS, and GKE.
  - Automation and Tools: Proficiency in using chaos engineering tools and automation frameworks to conduct experiments in a controlled and automated manner.
  - Observability and Monitoring: Skilled in implementing observability and monitoring "}]

content = "Create an hypothesis and experiment (step-by-step) to improve the stability of the (checkoutservice) api, there were a lot of trouble in the past with the connection to th

messages.append({"role": "user", "content": content})

completion = client.chat.completions.create(
    messages=messages,
    model="gpt-3.5-turbo",
)

chat_response = completion.choices[0].message.content
print(f'SRE Pipeline: {content}')

print(f'Chaos Engineering CoPilot: {chat_response}')
```

System Role creation

Improve the result by adding Context based on historical data (observability and incident post-mortem reports).

Chaos Experiment YAML is ready for execution. Hypothesis saved into the Chaos Engineering backlog.

DEMO

Starting in 3.... 2.... 1....

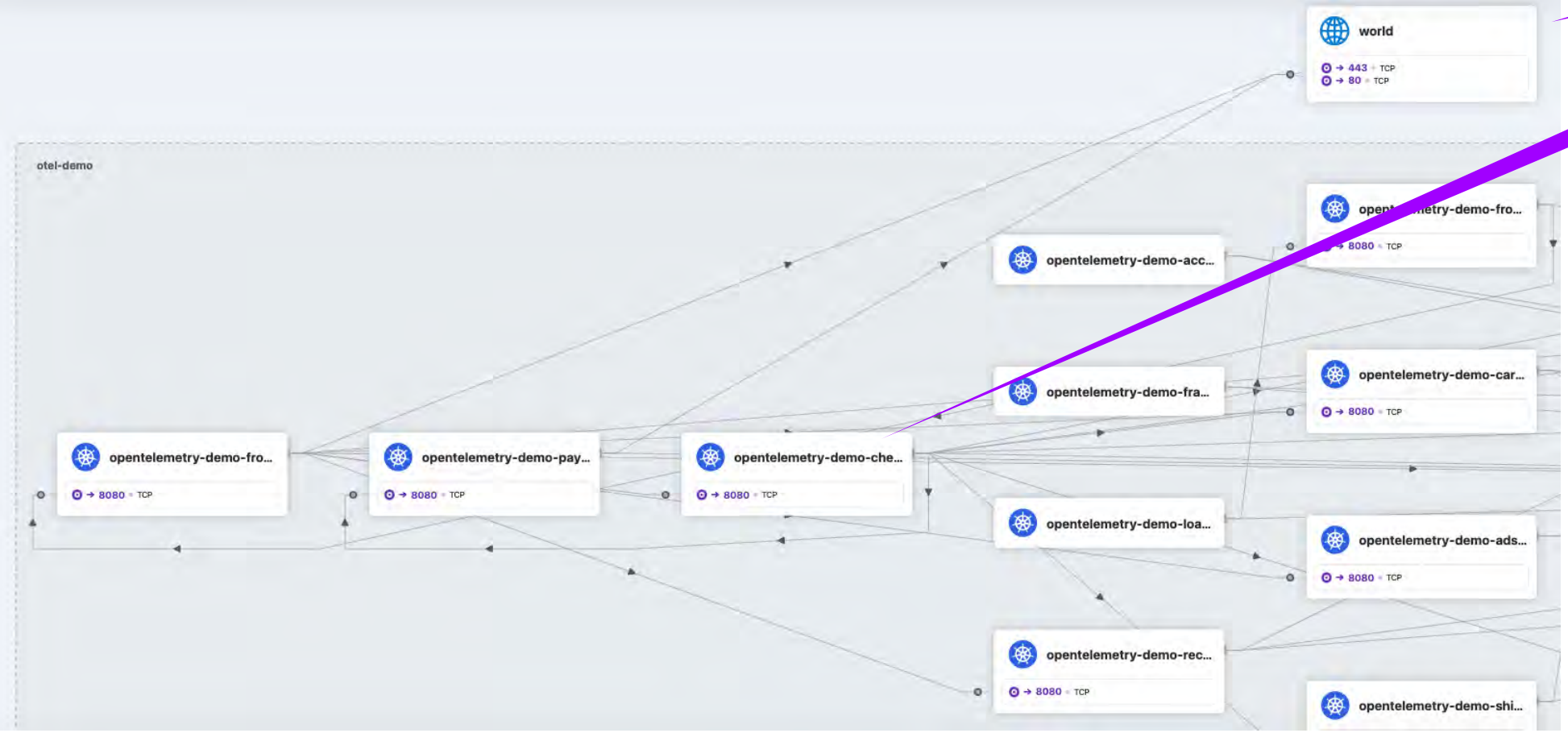
Photo by Alex Kondratiev on Unsplash

Demo: AI-drive Chaos Engineering Platform

Enhanced Target Architecture

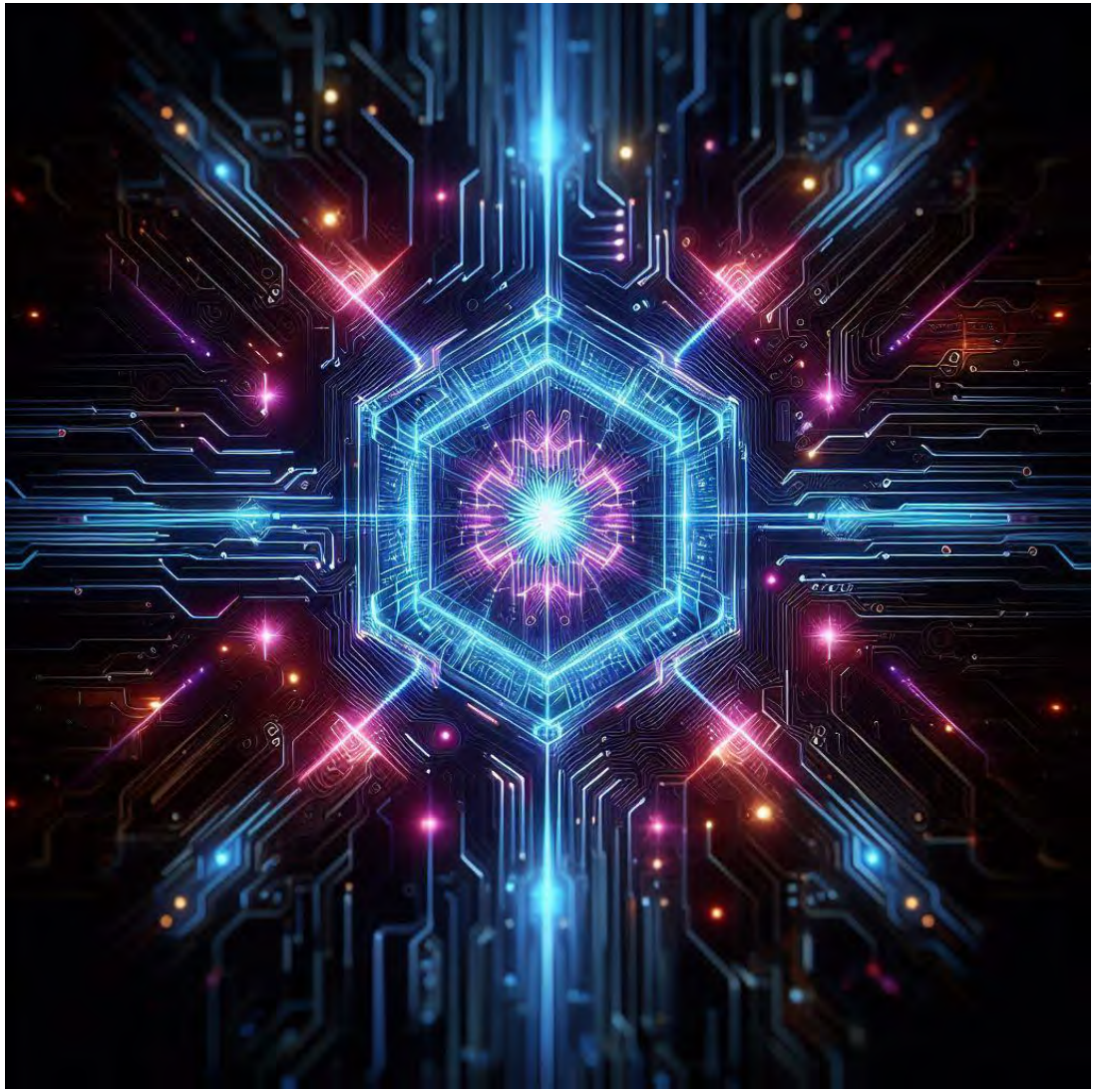
hubble-ui view

Target service



Conclusion & Takeaways

- **AI** , **GenAI** and **eBPF** can be leveraged to better detect **chaos experiments**
- **eBPF** goes beyond classical observability
- **AI** can significantly enhance threat detection capabilities by analyzing real-time data and identifying patterns indicative of security risks
- **Start small, scale fast**



Source: generated with GenAI on Bing.com

Thank You