The Inspektor Gadget Project

An eBPF systems inspection tool and framework

CNCF Sandbox Project
Goals

eBPF
You have a high-level understanding of eBPF and its superpowers

Inspektor Gadget
You learn about Inspektor Gadget and how it “supercharges” eBPF

Fun
You are engaged during this talk and enjoy learning about these items
1. Introduction
2. What is eBPF?
3. What is Inspektor Gadget?
4. Why Inspektor Gadget?
5. How to leverage IG
6. What’s next?
Hello!

Maya Singh
Product Manager
@ Microsoft

Mauricio Vasquez Bernal  Chris Kuehl
Disclaimer:

Under Active development!

+ Some features are behind an experimental flag or in progress
+ Others still in development branches
+ Will stabilize over the next few releases/months
+ Eager to have your feedback on functionality and UX
What do you think of when you hear “eBPF?”
What is eBPF?
What is eBPF?

eBPF is in-kernel bytecode runtime used for tracing, security, networking etc…

**eBPF Capabilities**

+ Brings flexibility to the kernel
+ Low strain from a performance perspective
+ Won’t crash your kernel
Examples of eBPF use cases

**Tracing**
eBPF can be used to measure CPU usage, memory allocation, and similar metrics which can be used for performance troubleshooting.

**Security**
eBPF can be used to enforce access control policies, you can whitelist/blacklist specific system calls, network connection etc…

**Networking**
eBPF allows for packet filtering and modification within the Linux kernel (Firewall rules)
eBPF Hooks

- eBPF is event driven, when “hooks” are passed, eBPF programs are executed
**eBPF Maps**

- Key/Value structures to share information between eBPF programs and user space applications

Source: Mauricio Vasquez Bernal & Chris Kuehl
What is eBPF?

eBPF is in-kernel bytecode runtime used for tracing, security, networking etc…

**eBPF Capabilities**

+ Brings flexibility to the kernel

+ Low strain from a performance perspective

+ Safe way to access the kernel

**eBPF Challenges**

+ Steep learning curve

+ Requires deep level of low-level systems troubleshooting

+ Limited higher level context
eBPF programs attach to kernel primitives (hooks) like sockets, syscalls, Tracepoints, etc. and run when an event occurs.

Source: Mauricio Vasquez Bernal & Chris Kuehl
What is Inspektor Gadget?
What is Inspektor Gadget?

**Tool**
A set of tools (gadgets) that empower users to inspect Linux and Kubernetes systems using eBPF programs in an accessible way.

**Framework**
A method through which eBPF developers can easily build, package, deploy, and run “gadgets”.

**Community**
Bridging the gap between highly technical eBPF concepts and the everyday developer who wants visibility into Linux and Kubernetes systems.
eBPF programs attach to kernel primitives like sockets, syscalls, Tracepoints, etc. end run when an event occurs.

- Enrichment
- Filtering
- Userspace processing
- Data export
- Sharing & distribution
- Many modes of use

Source: Mauricio Vasquez Bernal & Chris Kuehl
Event enrichment and filtering

+ Problem: events from eBPF give low-level data:
  + Kernel namespaces
  + cgroups

+ Solution: event enrichment adds high-level data:
  + Kubernetes pods, containers
  + Domain names or Kubernetes services from IP
  + Container information

+ Event filtering: showing a subset of events
  + From selected containers, Kubernetes pod, namespace, labels
  + Filtered in eBPF for performance, but abstracted for gadget authors

Source: Mauricio Vasquez Bernal & Chris Kuehl
Event enrichment and filtering:
Abstracted from the eBPF code

---

**eBPF program**

**Kernel**
- Mount namespace
- Cgroup id
- Other data

**Userspace**
- Kubernetes API server
- Container managers & runtimes
- ...other

**eBPF maps**

**IG (userspace)**
- Kubernetes namespace, pod, container
- Systemd unit
- From IP address to Kubernetes Service

---

**Event enrichment and filtering**

Source: Mauricio Vasquez Bernal & Chris Kuehl
Anatomy of a gadget

- Information about
  - The gadget
  - Capabilities
  - Output formatting
  - Build information
- One or more eBPF programs
- Userspace modules for post-processing of eBPF data.
  - Can be in any language WASM supports

* Also looking to include
  - Documentation
  - Source code
  - Logo
  - etc.

Source: Mauricio Vasquez Bernal & Chris Kuehl
Anatomy of a gadget

OCI registry

Gadget (OCI Image)
- Metadata (yaml file)
- eBPF programs+maps (ELF file)
- Userspace module (Wasm file)

Linux host

Image push

Image pull

Source: Mauricio Vasquez Bernal & Chris Kuehl
“Official” Gadgets

+ **Advise:** Recommend system configurations based on collected information
  + seccomp-profile, network-policies
+ **Audit:** Audit a subsystem
  + seccomp
+ **Profile:** Profile different subsystems
  + block-io, cpu
+ **Snapshot:** Take a snapshot of a subsystem and print it
  + process, socket
+ **Top:** Gather, sort and periodically report events according to a given criteria
  + file, tcp
+ **Trace:** Trace and print system events
  + bind, dns, exec, mount, oomkill, tcp{drop, retrans}, open, few more…
Sample Gadgets

**trace DNS**
Prints information about DNS queries and responses sent and received by different pods.

**top block io**
Used to visualize containers generating the most block device input/output.

**snapshot process**
Gets a list of running processes on the host.
Which type of gadget seems the most useful to you?

Menti.com | 2209 6641
Why Inspektor Gadget?
Background

+ Started in 2019
+ Wanted to bring eBPF and BCC tools to Kubernetes
  + …meaning it was Kubernetes only
+ Now supports…
  + Linux hosts with the ig cli tool
  + Kubernetes with the **kubectl-gadget** kubectl plugin
+ Discovered we had essentially built a framework
+ Has been transitioning from a collection of “built-in” **gadgets** to tool for building, packaging and running **gadgets** packaged as OCI images
Why Inspektor Gadget?

+ eBPF is an extremely powerful tool for gathering system information
+ But eBPF is hard – technically and intuitively
+ Once you have data, it’s still not immediately useful
  + How does this kernel data relate to my system as I understand it?
  + Where do I send the data?
+ Lots of additional tooling needed for…
  + Managing eBPF programs
  + Mapping kernel data to higher-level resources (K8s, container runtimes, etc.)
  + Doing userspace processing
  + Exporting data / providing data via API
<table>
<thead>
<tr>
<th>ARMO</th>
<th>MS Defender</th>
<th>Amazon EKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARMO and the Opensource project Kubescape use IG to enhance detecting vulnerabilities in containers</td>
<td>Inspektor Gadget is used in MS Defender for Containers to collect security events, generate insights and real-time threat detection alerts.</td>
<td>Amazon EKS Users leverage Inspektor Gadget to inspect their Kubernetes environment with eBPF tools</td>
</tr>
</tbody>
</table>

Source: [Empowering Kubernetes Observability with eBPF on Amazon EKS](link) | [Containers](link) | [CNCF On demand webinar: eBPF-based Kubernetes Security](link) | [CNCF](link) |
How you can leverage IG
**Inspektor Gadget Modes of Operation**

+ **Linux host**
  + Ig binary
  + Ig inside a container

+ **Client-server setup**
  + Ig runs as a service inside the host
  + We use a Client called gadgetctl to control the service (via API call)

+ **Kubernetes**
  + Ig is deployed via daemon set
  + Kubectl-gadget plugin used to control the daemon set

+ **Go library API**
  + Work in progress
Inspektor Gadget Data Export Options

**Raw Data**
We provide a json file of data that you can then do with what you please

**Prometheus**
We are working towards a more efficient exporting of metrics to Prometheus

**OpenTelemetry**
Ultimately, we want to support logs, metrics, and traces through OTel
Demo – trace DNS and top TCP
## Demo – trace DNS and top TCP

### trace DNS output

```
 Demo gadget trace dns --podname my-app

<table>
<thead>
<tr>
<th>K8S.NODE</th>
<th>K8S.NAMESPACE</th>
<th>K8S.POD</th>
<th>PID</th>
<th>TID</th>
<th>COMM</th>
<th>QR</th>
<th>TYPE</th>
<th>QTYPE</th>
<th>NAME</th>
<th>RCODE</th>
<th>NUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>aks-node...s000001</td>
<td>default</td>
<td>my-app</td>
<td>27554</td>
<td>27554</td>
<td>nslook up Q OUTGOING A</td>
<td>wrong-url.test.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>aks-node...s000001</td>
<td>default</td>
<td>my-app</td>
<td>27554</td>
<td>27554</td>
<td>nslook up R HOST A</td>
<td>wrong-url.test. Non-Existent</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>aks-node...s000001</td>
<td>default</td>
<td>my-app</td>
<td>27555</td>
<td>27555</td>
<td>nslook up Q OUTGOING A</td>
<td>example.com.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>aks-node...s000001</td>
<td>default</td>
<td>my-app</td>
<td>27555</td>
<td>27555</td>
<td>nslook up R HOST A</td>
<td>example.com.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
```

### top TCP output

```
<table>
<thead>
<tr>
<th>K8S.NODE</th>
<th>K8S.NAMESPACE</th>
<th>K8S.POD</th>
<th>K8S.CONTAINER</th>
<th>PID</th>
<th>COMM</th>
<th>IP</th>
<th>SRC</th>
<th>DST</th>
<th>SENT</th>
<th>RECV</th>
</tr>
</thead>
<tbody>
<tr>
<td>aks-no...000002</td>
<td>default</td>
<td>pod-1</td>
<td>pod-1</td>
<td>11315</td>
<td>proxy</td>
<td>4</td>
<td>p/default/pod-1:8080</td>
<td>p/default/pod-2:52338</td>
<td>554.6...</td>
<td>535B</td>
</tr>
<tr>
<td>aks-no...000001</td>
<td>kube-system</td>
<td>konnc...-k77p4</td>
<td>konnc...-konnc...</td>
<td>19106</td>
<td>proxy</td>
<td>4</td>
<td>p/kube-system/konnc...</td>
<td>p/kube-system/pod:8080</td>
<td>12.76...</td>
<td>535B</td>
</tr>
<tr>
<td>aks-no...000002</td>
<td>gadget</td>
<td>gadget-blshd</td>
<td>gadget</td>
<td>10488</td>
<td>gadget</td>
<td>4</td>
<td>r/127.0.0.1:8080</td>
<td>r/127.0.0.1:60262</td>
<td>5.496...</td>
<td>30B</td>
</tr>
<tr>
<td>aks-no...000000</td>
<td>kube-system</td>
<td>konnc...-v6xhn</td>
<td>konnc...-konnc...</td>
<td>18786</td>
<td>proxy</td>
<td>4</td>
<td>p/kube-system/konnc...</td>
<td>p/kube-system/pod:8080</td>
<td>4.259...</td>
<td>304B</td>
</tr>
<tr>
<td>aks-no...000000</td>
<td>gadget</td>
<td>gadget-vqrv6</td>
<td>gadget</td>
<td>18859</td>
<td>gadget</td>
<td>4</td>
<td>r/127.0.0.1:8080</td>
<td>r/127.0.0.1:5352</td>
<td>4.08K...</td>
<td>43B</td>
</tr>
<tr>
<td>aks-no...000002</td>
<td>gadget</td>
<td>gadget-dgktv</td>
<td>gadget</td>
<td>9943</td>
<td>gadget</td>
<td>4</td>
<td>r/127.0.0.1:8080</td>
<td>r/127.0.0.1:57992</td>
<td>3.985...</td>
<td>30B</td>
</tr>
</tbody>
</table>
What’s Next?
Looking ahead

+ Support declarative way to run gadgets
  + Configuration file
+ Support of various export options
+ Golang API full support of image based gadgets
+ Gadgets in Artifacthub.io
+ Understand community priorities
+ Proper documentation for all this ;-)
Think about how eBPF could be used to enhance the projects you’re working on and see if we have a gadget that could help you!

Call to Action
Web: inspektor-gadget.io
Slack: #inspektor-gadget on the Kubernetes Slack
Github: github.com/inspektor-gadget/inspektor-gadget

Hope to see you around SCaLE!

Thank you