

Cirries-Diverter – Universal Network Mirroring Agent

Overview:

Cirries-Diverter mirrors network or application-level traffic using Linux tc and VXLAN encapsulation. Deployable across EC2, ECS, EKS, or hybrid environments to send mirrored packets to a remote collector (e.g., DART AI AWS PacketSensor, Wireshark, or any VXLAN receiver).

Automated test procedure for DART AI VPB 10Gbps

On successful subscription user need to download below scripts according to their environment for automated testing:

EC2 environment:

- Download setup script from:
https://download.cirries.com/cirries_div_aws_mp/ec2_setup_10gbps.sh
- give executable permission: `chmod +x ec2_setup_10gbps.sh`
- run script from AWS CLI: `./ec2_setup_10gbps.sh <collector_IP>`

Description:

This script automatically creates a temporary EC2 instance using the latest Ubuntu 22.04 AMI, installs Docker, and runs the Cirries Diverter container from ECR.

What it does:

1. Creates a minimal IAM role, security group, and EC2 instance.
2. Installs Docker and AWS CLI inside the instance.
3. Pulls and runs your ECR image:
4. `709825985650.dkr.ecr.us-east-1.amazonaws.com/cirries/dart_ai_vpb_agent_10gbps:0.0.1`
5. Verifies container status (`docker ps`, `docker logs`) and displays public IP.
6. Prompts to optionally clean up resources.

EKS environment:

- Download setup script from:
https://download.cirries.com/cirries_div_aws_mp/eks_setup_10gbps.sh
- Download daemonset yaml file from:
https://download.cirries.com/cirries_div_aws_mp/AWSEKS10Gbps.yaml
- give executable permission: `chmod +x eks_setup_10gbps.sh`
- run script from AWS CLI: `./eks_setup_10gbps.sh <collector_IP>`
- Once done apply yaml using kubectl: (note: change collector_ip in yaml)
 - `kubectl apply -f AWSEKS10Gbps.yaml`

Description:

This script creates a temporary EKS cluster.

What it does:

- Creates (or reuses) an EKS cluster and node group.
- Sets up kubectl and configures cluster access.
- Deploys a Deployment manifest referencing the Cirries ECR image.
- Waits until the pod reaches Running state and fetches container logs.
- Cleans up EKS resources (optional).

Post Deployment verification:

On collector node (DART AI PacketSensor, or VXLAN-capable host):

- REMOTE_IP=<collector-public/private-ip> # e.g. 3.91.45.200/192.168.x.x
 - SRC_IFACE=ens5/eth0
 - VXLAN_ID=100
 - VXLAN_PORT=4789
 - VXLAN_IFACE=vxlan100
 - sudo ip link add \$VXLAN_IFACE type vxlan id \$VXLAN_ID dev \$SRC_IFACE remote \$REMOTE_IP dstport \$VXLAN_PORT
2. ip link set vxlan100 up
 3. tcpdump -i vxlan100 -nn

```
[root@mf_analyzer ec2-user1# tcpdump -i vxlan100 -nn
tcpdump: verbose output suppressed, use -v(v)... for full protocol decode
listening on vxlan100, link-type EN10MB (Ethernet), snapshot length 262144 bytes
12:38:38.732140 IP 192.168.5.43.45321 > 3.227.59.45.4789: VXLAN, flags [I] (0x08), vni 100
IP 192.168.5.43.45321 > 3.227.59.45.4789: VXLAN, flags [I] (0x08), vni 100
IP 192.168.80.108.443 > 192.168.27.150.35412: Flags [P.], seq 96630460:96630558, ack 1759286563, win 501, options [nop,nop,TS val 1085705300 ecr 1932303293], length 98
12:38:38.732165 IP 192.168.80.108.443 > 192.168.27.150.35412: Flags [P.], seq 0:98, ack 1, win 501, options [nop,nop,TS val 1085705300 ecr 1932303293], length 98
12:38:38.732207 IP 192.168.5.43.22800 > 3.227.59.45.4789: VXLAN, flags [I] (0x08), vni 100
IP 192.168.27.150.35412 > 192.168.80.108.443: Flags [.], ack 98, win 592, options [nop,nop,TS val 1932305786 ecr 1085705300], length 0
12:38:38.732236 IP 192.168.5.43.45321 > 3.227.59.45.4789: VXLAN, flags [I] (0x08), vni 100
IP 192.168.5.43.45321 > 3.227.59.45.4789: VXLAN, flags [I] (0x08), vni 100
IP 192.168.80.108.443 > 192.168.27.150.35412: Flags [P.], seq 98:196, ack 1, win 501, options [nop,nop,TS val 1085705300 ecr 1932303293], length 98
12:38:38.732221 IP 192.168.5.43.22800 > 3.227.59.45.4789: VXLAN, flags [I] (0x08), vni 100
IP 192.168.27.150.35412 > 192.168.80.108.443: Flags [.], ack 196, win 592, options [nop,nop,TS val 1932305786 ecr 1085705300], length 0
12:38:38.732245 IP 192.168.80.108.443 > 192.168.27.150.35412: Flags [P.], seq 98:196, ack 1, win 501, options [nop,nop,TS val 1085705300 ecr 1932303293], length 98
```

Manual test procedure:

Prerequisites (All Platforms)

4. Supported Platforms:

- Amazon EC2 (Linux)
- Amazon EKS (Kubernetes DaemonSet)
- Amazon EKS Anywhere
- ✗ Not supported on AWS Fargate, Lambda, or Windows.

5. System Requirements:

- **Linux Kernel 4.14+** (eg. Amazon Linux 2, RHEL 8+, Ubuntu 20+).
- **VXLAN & tc support:**
 - **Check: `sudo lsmod | grep -E "vxlan|clsact|mirred|u32|ingress"`**

1. Expected output:

```
cls_u32          20480 0
act_mirred       16384 0
sch_ingress     16384 1
vxlan           57344 0
ip6_udp_tunnel  16384 1 vxlan
udp_tunnel      16384 1 vxlan
```

If not loaded you can load with below commands

2. `sudo modprobe vxlan`
 3. `sudo modprobe sch_clsact`
 4. `sudo modprobe act_mirred`
 5. `sudo modprobe cls_u32`
 6. `sudo modprobe sch_ingress`
1. **Docker 20.10+** installed and running:

2. Networking:

- UDP port **4789** open between diverter and collector.
- Ensure network connectivity via VPC Peering, VPN, or Cloud WAN.
- Use private IP if in same VPC; otherwise use public or routable IP.

6. Collector Setup:

On collector node (DART AI PacketSensor, or VXLAN-capable host):

- REMOTE_IP=<collector-public/private-ip> # e.g. 3.91.45.200/192.168.x.x
- SRC_IFACE=ens5/eth0
- VXLAN_ID=100
- VXLAN_PORT=4789
- VXLAN_IFACE=vxlan100

- `sudo ip link add $VXLAN_IFACE type vxlan id $VXLAN_ID dev $SRC_IFACE remote $REMOTE_IP dstport $VXLAN_PORT`
 - 7. `ip link set vxlan100 up`
 - 8. `tcpdump -i vxlan100 -nn`
-

Default cfg file: (user can copy paste this file and run in EC2 with modified configuration)

Interface to capture packets from (auto-detected if empty)

SRC_IFACE=

Interface to send mirrored traffic (default: vxlan100)

MIRROR_IFACE=vxlan100

0 = ingress only, 1 = egress only, 2 = both

MIRROR_TYPE=2

Remote collector (VXLAN destination)

#REMOTE_IP=

VXLAN_ID=100

VXLAN_PORT=4789

Optional filters (leave empty to mirror all)

FILTER_SRC_CIDR=

FILTER_DST_CIDR=

FILTER_PROTO=ip

FILTER_SRC_PORT=

FILTER_DST_PORT=

Deployment Options

Amazon EC2 (Direct Docker Run)

```
sudo docker run -d --rm --privileged \  
--name cirries-diverter \  
-e REMOTE_IP=<Collector_IP> \  
-e MIRROR_TYPE=2 -e VXLAN_ID=100 -e VXLAN_PORT=4789 \  
709825985650.dkr.ecr.us-east-1.amazonaws.com/cirries/dart_ai_vpb_agent_10gbps:0.0.1
```

Amazon EKS (DaemonSet)

Before deployment:

EKS cluster created and nodes joined

```
$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
ip-192-168-20-23.ec2.internal	Ready	<none>	2m16s	v1.32.6
ip-192-168-39-237.ec2.internal	Ready	<none>	2m23s	v1.32.6

Node security groups allow UDP 4789

Nodes support privileged pods (kubelet --allow-privileged=true) also

- **Check: `sudo lsmod | grep -E "vxlan|clsact|mirred|u32|ingress"`**

1. Expected output:

```
cls_u32          20480 0  
act_mirred       16384 0  
sch_ingress      16384 1  
vxlan            57344 0  
ip6_udp_tunnel  16384 1 vxlan  
udp_tunnel       16384 1 vxlan
```

If not loaded you can load with below commands

2. `sudo modprobe vxlan`
3. `sudo modprobe sch_clsact`
4. `sudo modprobe act_mirred`
5. `sudo modprobe cls_u32`
6. `sudo modprobe sch_ingress`

Download DaemonSet yaml:

wget https://download.cirries.com/cirries_div_aws_mp/AWSEKS10Gbps.yaml

Change REMOTE_IP and then Apply: `kubectl apply -f AWSEKS10Gbps.yaml`

\$ kubectl get pods -n cirries -o wide

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED
NODE	READINESS	GATES					
cirries-diverter-tvgzd	1/1	Running	0	3m13s	192.168.39.237	ip-192-168-39-237.ec2.internal	<none>
cirries-diverter-wd2j9	1/1	Running	0	3m13s	192.168.20.23	ip-192-168-20-23.ec2.internal	<none>

\$ kubectl logs -f -n cirries cirries-diverter-tvgzd

2025-10-31T13:22:34Z - No config file found, using defaults

2025-10-31T13:22:34Z - Auto-detected primary interface: ens5

2025-10-31T13:22:34Z - Creating VXLAN interface vxlan100 → 3.227.59.45 via ens5

2025-10-31T13:22:34Z - Applying mirroring: src=ens5 → vxlan100, mode=2

2025-10-31T13:22:34Z - No filters — mirroring all traffic

2025-10-31T13:22:34Z - Mirroring setup complete

2025-10-31T13:22:34Z - No config file found, using defaults

2025-10-31T13:22:34Z - Auto-detected primary interface: ens5

2025-10-31T13:22:34Z - Creating VXLAN interface vxlan100 → 3.227.59.45 via ens5

2025-10-31T13:22:34Z - Applying mirroring: src=ens5 → vxlan100, mode=2

2025-10-31T13:22:34Z - No filters — mirroring all traffic

2025-10-31T13:22:34Z - Mirroring setup complete
