SC18 Network Research Exhibition: Demonstration Publishable Submission

Poseidon: A Machine Learning Approach to Network Device Role and Behavior Identification

Charles Lewis, Cyber Reboot (An IQT Lab), clewis@iqt.org

Abstract

Effective network security relies heavily upon the situational awareness of the network operator. At minimum, addressing risks requires the operator to answer two basic questions in near real-time: what is on the network; and what it is doing?

In practice, answering these questions accurately has proven to be a challenge in and of itself. Doing so at scale in a reasonable, automated fashion can be debilitating to all but the smallest of organizations.

Poseidon[1] is an open-source project focused on leveraging software-defined networks (SDNs) and machine learning (ML) techniques to answer these two questions. Poseidon approaches this problem by sitting "on top of" SDN controllers – which includes Faucet[2] – effectively deployed as a northbound application. It then automates actions such as mirroring and analyzing traffic using SDN interfaces.

Poseidon takes cues from Faucet to discover new devices on the network, both at Layers 2 and 3. It then takes those cues to mirror traffic from those devices for a period of time in order to feed trained ML models.

The ML models only require that the packet headers in the traffic captures be retained, thereby alleviating most privacy and encryption challenges. In addition, the payloads are stripped away during capture. Poseidon currently has two models, the first model classifies the device as a given role (e.g. Developer Laptop, Active Directory Controller, Printer, etc.) and the second is designed to determine if the behavior of the device is normal for that role. The second model is still in its infancy, and will not be a focus for SC18.

Finally, the results are fed back to Poseidon, which a network operator can use for situational awareness or to take defensive action. Additionally, this automated feedback loop of learning, analyzing, and device recognition progresses over time to provide a historical view of the network.

Goals

1. Connect Poseidon to a Faucet controller running a large network without adversely impacting it.
2. Use Poseidon to scale the number of devices it can handle from hundreds to thousands.
3. Automatically visualize the network Poseidon has learned with CRviz[3], an open-source network visualization tool.
4. Be able to classify learned devices into roles, with less than 30% yielding an 'unknown' label.

Resources

Poseidon deployments can be distributed, based on where the SDN controller(s) reside. Poseidon only learns and analyzes traffic it has access to, which is ultimately up to the controller or the configuration of the network and the location where Poseidon is plugged in.

For full visibility, Poseidon needs a dedicated mirror port on any switches controlled by Faucet. Alternatively, Poseidon can be selectively deployed on specific segments or via a downstream trunk port that may limit the traffic it is allowed to see.

For this deployment to be successful, ideally Poseidon should be permitted to analyze traffic from as many devices on the network as possible. Since it only relies on read/write access to a Faucet

[1] https://github.com/CyberReboot/poseidon
[2] https://github.com/faucetsdn/faucet
[3] https://github.com/CyberReboot/CRviz

instance, it can work with any networking gear that is interoperable with Faucet. Additionally, Poseidon requires basic connectivity to the internet for the initial start-up.

Power for the Poseidon server in the vicinity of the switch ports dedicated for mirroring would also be necessary for this deployment.

1. Connecting Poseidon to Faucet on a Large Network

Deploying Poseidon requires not only connecting it to an SDN controller, but also being able to control the configuration settings of that SDN controller. It may not be desirable to connect Poseidon to the SDN controller that also manages the entire network for a variety of reasons such as increased risk, ownership, and access.

Since Faucet is interoperable with Open vSwitch[4] it allows Poseidon to be deployed in a way that is both safe and still useful on networks where it might not be reasonable to give Poseidon full control.

For this deployment there is a main Faucet controller that manages a number of switches of which Poseidon is neither aware of, nor does it need to be. That main Faucet controller has allocated one mirror port on a switch it manages, where traffic from other devices on that network is mirrored to.

That mirror port is a 100GbE port that will receive traffic from various devices on various VLANs and is connected to the server running Poseidon.

The Poseidon server for this deployment is a Dell R420 1U rack mount server that has 128GB DDR3 memory, 2 Intel Xeon ES-2400 processor sockets, and a Mellanox MCX516A-CCAT ConnectX-5 EN Network Interface Card 100GbE Dual-Port QSFP28.

See Figure 1 for how the Poseidon server is connected to the SCinet network.

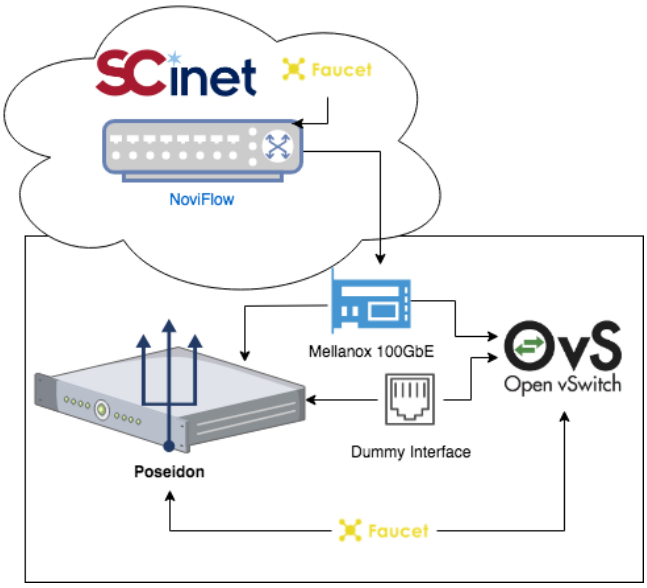

Figure 1. Diagram of how Poseidon will be connected to SCinet for this demonstration.

The Poseidon server will be running Faucet, Open vSwitch, and Poseidon itself. Open vSwitch will be configured to have a virtual bridge with 2 ports. The first port is attached to the Mellanox physical 100GbE port that is attached to the mirror port on the physical switch. The second port is mapped to a dummy device that will be used for Poseidon to run investigations of devices that are learned on the Mellanox port. That virtual bridge is then connected to the instance of Faucet that is running on the Poseidon server. This allows Poseidon to have full control over this virtual switch and the two attached ports without having to give Poseidon access to the main Faucet that controls the real network.

Separating Poseidon from controlling the real network in this way allows a number of benefits. Firstly, it ensures that if Poseidon crashes, has an outage, or does something unexpected to the Faucet instance running on the Poseidon server it does not adversely affect the real network or the main Faucet controller and has no operational impact to any of the other devices on the network. Secondly, it allows the main Faucet to decide what traffic Poseidon gets to see. There may be reasons why some of the traffic should never be sent to

[4] https://www.openvswitch.org/

Poseidon, and this ensures that the data owners can enforce these boundaries should they choose to.

2. Scaling Poseidon

Poseidon hooks into learn events generated by Faucet. When Faucet sees a new device on the network it sends an event and Poseidon decides what it wants to do with it.

Poseidon has several gates in place to help it perform under the stresses of very large networks - both in density of network traffic and density of devices:

- One of the gates strips off the payloads of every packet, allowing Poseidon to capture at higher rates. That not only improves performance, but also allows Poseidon to work in privacy sensitive networks or on encrypted traffic. This also reduces the amount of data that needs to be processed down the pipeline, since it only needs and captures the headers of each packet.
- The second major gate that Poseidon has is the ability to set a limit of concurrent investigation threads. This allows Poseidon to keep a queue of new device events provided by Faucet without having to process all of them at once. Since Poseidon is simply a lens into the network it does not need to see everything from everywhere all the time. It can just take slices as it has the resources to do so.
- Finally, as Poseidon is processing investigation events and gathering network traffic captures, it puts the subsequent work down the pipeline into queues that are managed based on the system load. Those queues effectively throttle the system should it start to get overwhelmed with work.

With these gates in place, Poseidon is built with the intent to withstand thousands of devices on a network and large network pipes in the range of 10-100G. While it may not be able to process everything on very high bandwidth networks, it is designed to process samples of it at a frequency that won't overwhelm whatever system resources Poseidon happens to be running on.

3. Visualizing Poseidon

CRviz, an open-source network visualization tool, was specifically built to be able to visualize large networks in a scalable way. While Poseidon has not been run on a network with thousands of devices yet, CRviz has been tested with datasets in the thousands to verify that it will work as expected when Poseidon is run on a network with thousands of devices. See Figures 2 and 3 for samples of what a network with 4000 devices on it could look like. For an interactive demonstration version, go to: https://cyberreboot.github.io/CRviz

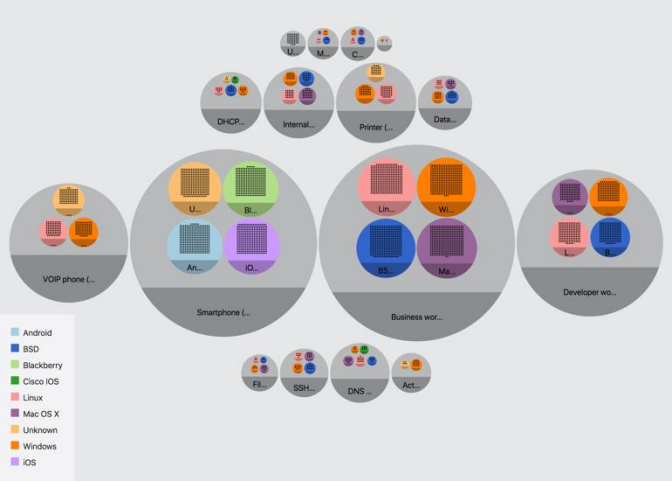

Figure 2. CRviz showing a network of 4000 devices broken up into outer circles by the type of device it was classified into, and then further broken down into the inner circles and colored by the Operating System of those devices.

Figure 3. The same network as shown in Figure 2, but zoomed into a specific inner circle and hovered over a specific device to view additional metadata that Poseidon learned about it.

4. Classifying Learned Devices into Roles

Poseidon leverages a plugin system called Vent[5] that provides the ability to swap in or out various tools that can operate on the network traffic that Poseidon sees in an automated fashion. For the purposes of this deployment, Poseidon will be configured with four such tools. These four tools work in a pipeline that feed inputs and outputs to one another.

Since the traffic for this deployment is VLAN tagged, the first tool simply takes the captured traffic and rewrites it to not be tagged. This ensures that all of the sessions and packets across captures that may or may not be using VLAN tagged traffic will be processed in a consistent way.

The output of the first tool feeds the input of the second tool which takes the captured traffic and breaks it up into server/client sessions. Each of these sessions are then passed on as inputs to both the third and fourth tools.

The third tool is p0f[6] which is used for passive OS fingerprinting and allows Poseidon to understand the operating system of the device it is learning, giving Poseidon additional insight into the decision making process.

The fourth tool is PoseidonML[7] which is the machine learning portion of Poseidon. It takes the output of sessions from the second tool and runs a classifier on the traffic to determine the role of the device that was being investigated. Once the device is classified the traffic is then used to evaluate against a second model that determines if the device is behaving normally or abnormally based on the role it was classified into. As previously mentioned, the focus for this deployment is on the role classification, rather than the behavior.

5. Conclusion and Future Work

Being able to run Poseidon on a production network with a variety of devices and large amounts of traffic is a big step forward for the project. Validating that the role classification works reasonably well on a real network will help get Poseidon one step closer to building a more mature behavior detection.

Successfully deploying Poseidon on SCinet will prove out that it can be realistically used in enterprise environments and open up the conversation for automating network security by leveraging SDN.

Poseidon already has the mechanisms in place to make configuration changes to Faucet based on information it receives. Ideally the types of configuration changes should be expanded beyond mirroring traffic to more interesting actions such as throttling, quarantining, automated ACLs, and other mitigations that might be appropriate when a device on the network is detected to not be behaving in an expected way for the role it was classified into.

Involved Parties

- Charles Lewis, Cyber Reboot Lab, clewis@iqt.org
- Alice Chang, Cyber Reboot Lab, alice@cyberreboot.org
- Josh Bailey, Google, joshb@google.com

[5] https://github.com/cyberreboot/vent
[6] http://lcamtuf.coredump.cx/p0f3/#
[7] https://github.com/cyberreboot/poseidonml