Prometheus Network Monitor

Savannah Malo, Madison Norton, Jacob Skidmore, Jack Spitzer
Computer Science, University of New Hampshire, Durham, NH 03824

Introduction

Our sponsor, Scott Kitterman, aims to monitor the current UNH CS Department network using Prometheus to oversee and monitor traffic and performance. The goal is to enhance network security by utilizing the information provided by the Prometheus program. We researched Prometheus, along with with Net-SNMP, Docker, and Grafana, then installed them on the Drake server. The value provided to the IT admin is an ability to understand the traffic coming in and out of the network, scan for possible issues, give high-level overview of network health, and send alerts when issues are detected. These goals are evaluated by testing the program to see if it will properly respond to any anomalies that occur by successfully alerting the admin of the issue. Penetration tests and vulnerability tests are conducted to verify the application's accuracy.

Important Terminology

SNMP - This is the protocol that network devices use to share data and statistics.

Prometheus - Powerful data collection and analysis tool.

Grafana - Customizable web dashboard which can be configured to show data in a variety of ways.

Drake - The primary server provided by our project sponsor for implementing this project.

Docker- Application virtualization tool, placing each program into a box.

SSH - Tool to remotely access the command line for a server

Requirements

Functional requirements are as follows: a network admin must be able to collect data from switches and network appliances, and view them displayed in a dashboard. SNMP data sent by the CS network will be monitored by Prometheus and displayed on the Grafana Dashboard. Drake will host these three elements (SNMP gathering, Prometheus, and Grafana). The network admin shall receive an alert whenever there is an anomaly, triggered by rules encoded in Prometheus. This will instruct an email server to send the network admin an email reporting any deviations.

Non-functional requirements are as follows: the system will be designed with security measures in place to ensure the Prometheus system remains isolated from external access. This will be implemented by restricting access to the Prometheus system to only authorized entities. Additionally, comprehensive documentation and a network behavior log will be maintained for the benefit of future project teams. The system will be designed with flexibility in mind, to accommodate future changes, enhancements, and contributions.

Security requirements are as follows: if an attacker were to gain access to the Prometheus system, they would potentially have the ability to intercept, view, and manipulate sensitive server data from the CS network. Furthermore, there is the risk of denial of service, elevation of privileges, and other potential vulnerabilities. One possible scenario is that an attacker could exploit stolen credentials to illicitly modify the network, steal or destroy data, or disrupt the availability of the department's network by slowing it down or causing it to go offline. These risks should be reduced, if not outright prevented. These attacks will be mitigated by the access restrictions mentioned above and other security measures that will be implemented.

Network Topology

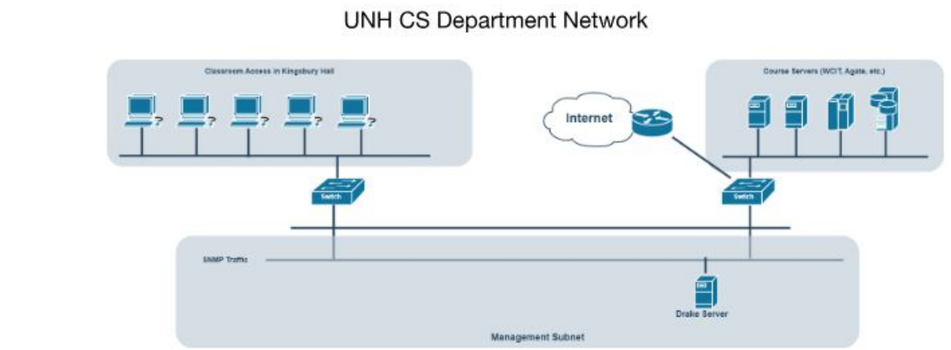

Figure 1: The UNH CS network being monitored. Essentially, there are several switches that can either connect to each other or connect to the internet via external servers.

Design & Implementation

In this project, we are working with two servers: The main server “Drake”, which is the project sponsor’s machine, and the test server. Each of these servers are accessed by connecting to a VPN, then using SSH to send commands to the server. We have used Docker as a way to efficiently run containers which each have only one software component for our project. This allows for better security, and means that a component can be removed or upgraded on it’s own without impacting any of the other software.

For the software that actually powers the project, we are using **Prometheus** as the core of our service, which operates as a data collection and organization system. It collects data from the network devices using the SNMP protocol, and gives it to Grafana for easy viewing, and an email notification system for alerting network admins to network failures.

Design Diagrams

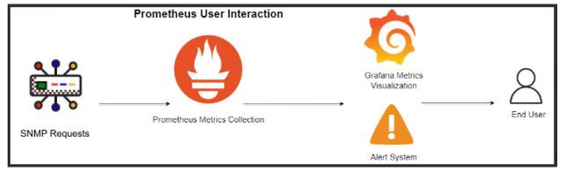

Figure 2: User interaction with the system. The end user only sees graphs or email alerts. However, the underlying infrastructure is Prometheus receiving data collected from SNMP requests on network technologies.

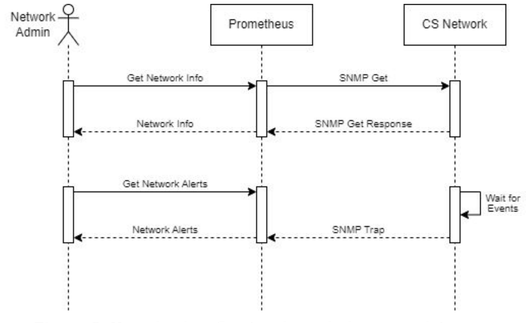

Figure 3: How the data flowing through the network is monitored. SNMP GET requests respond back to the Prometheus software with information.

Testing & Results

A testing server, with a similar software configuration to the production server, provides the ability to change the variables, or roll back the server to a previous state if something goes wrong. This allows us to see if the features we are implementing are functional, and if they do contribute to the MOV.

The results of this project consist of data graph visuals on the Grafana web interface. Network devices are monitored under various metrics, including inbound and outbound traffic, bandwidth, HTTP requests, uptime and downtime. Because of this, we come to understand potential security concerns when the data are reported beyond what is normal or expected.

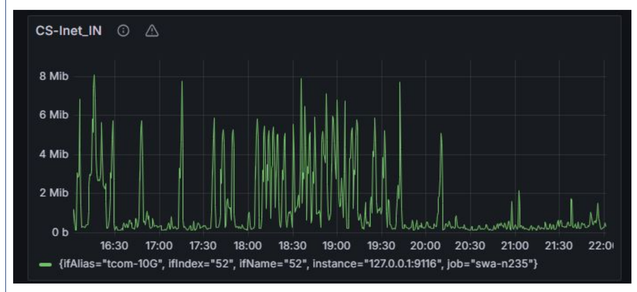

Figure 4: Example graph on Grafana measuring inbound network traffic.

Evaluation & Conclusion

It is important to determine which types of anomalous behavior are to be considered for concern. Manually sorting the mass amounts of data flowing through the network is a difficult task. Visual data can allow IT admin to easily determine outliers, which can be security risks. While the current state of the server can vary, the tools can be adjusted according to needs.

For example, spikes in data of inbound traffic represent an abnormal amount of incoming connections (see Figure 4). Excessive traffic can put strain on the server. Alternatively, the threshold for strain can be improved with updated technologies.

We have written a document outlining the security precautions associated with Prometheus and Grafana applications. It lists the scans needed to be performed on a consistent basis in order to keep security tight. Future work includes continuation of data collection on the server, by adding more network devices. This includes adjusting configurations as needed to meet security requirements.

Acknowledgements

Sponsor: Scott Kitterman
Advisor: Craig Smith