# Summary

## Meeting time and location

Madrid conference room @ Advance

## Participants

* Advance
* Dark nITes
  + Jeandre
  + Muhammed
  + Sisa

## Decisions/Amendments made/adopted

* Logging system will form part of the system. A log entry will include *time*, *username*, and the *action taken*
* User accounts/Log ins will be included as a security feature
* IPv6 support not to be added now; peg as a possible future feature
* Implementation of “unique algorithm per pool” requirement to be added later

# Overview

The purview of this meeting was to touch base and reconcile our progress with the client’s expectations.

## Metrics

* Use a blocking call to initiate Prometheus, and remove sleeps
  + Make a socket
  + Poll the server
  + Once a response has been received, continue
* Each resource typically exposes their own metrics, so when exposing backend metrics place the exporters on the resources themselves instead of centralising metrics exposure to the Eureka server
* Look at reading the eBPF maps in Python instead, instead of making calls to C prog.
* Alert Manager:
  + Can be used to send alerts out based on rules set on data from the metrics
    - Employ linear regression to try make predictions on status of system resources/network traffic behaviour
    - Include an alert rule that notifies on an increase in traffic based on a ratio
* Demo purposes:
  + Implement a Grafana playlist to cycle through dashboard panels on demo day

## Interfaces

* Implement confirmatory toasts
  + Info – confirms an action such as when an IP is manually whitelisted
  + Warning – A threshold of connections/packets per second is being reached, and an attack may ensue
  + Error –
* Also include a notice of actions, e.g. when an IP is dynamically blacklisted

## Logging system

* Store a statistical sample of our packets, instead of all of them
  + Perform statistical analyses on the sample to get meaningful data

## Load-balancing

* What happens if someone spoofs a whitelisted IP?:
  + ISPs tend not to service requests if a source IP is not one from their IP range

## Auditing

* Instead of row entries, consider making an Audit object:
  + Each object is titled after some identifying feature, e.g. the IP address
  + Payload of the packet is stored in object, and object is logged as an audit entry

# To-do

* Metrics
  + Implement blocking calls for Metrics initialisation
  + Look at reading eBPF maps directly in Python
  + Alert manager alert rules
  + Playlist for demo
* Interfaces
  + Confirmatory (automatic action) toasts for Info, Warning and Error
  + Include a feedback/confirmation of (manual) actions
* Auditing/Logging
  + Statistical sampling of data
  + Look towards implementing Audit objects