Captive Portals Hackathon

IETF 100
The Problem

• How can the API server identify a captive device?
Planning

• Some rough discussions over slack on Saturday
• Decided to build on the work from the IETF 98 hackathon
Technology and Code

• Lots of virtual machines
• [https://github.com/coova/coova-chilli](https://github.com/coova/coova-chilli) (Enforcement device)
• [https://github.com/darshakthakore/capport-detection](https://github.com/darshakthakore/capport-detection) (API server using Flask)
• [https://github.com/klarose/capport_98](https://github.com/klarose/capport_98) (Client side code)
Achievements

• Proof via working code of the feasibility of different deployment models
• Feedback for discussion at the capport meeting on Tuesday.
Adjacent Device
Adjacent Device

- Tested Explicit L2 Login to API
- Tested Explicit L3 Login to API
Remote Device

- Tested Explicit L2 Login to API
- Tested Explicit L3 Login to API
- Tested inferred L3 login to API
Findings

• API server co-located with enforcement device was easier than a remote one
• Inferring the identity made for a simpler, less stateful API.
• Inferring the MAC didn’t seem easy – didn’t try. Would probably try to use the arp table.
• For inferring the identity, the API server needs to be on the same route as the portal
Participants

- Kyle Larose (Sandvine)
- Tommy Pauly (Apple)
- Alexander Roscoe (Comcast)
- Donald Eastlake (Huawei)