THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

University of Waikato
Network Research Group

OpenLI

RADIUS IP Intercepts

OpenLI Training: Chapter Fifteen

Shane Alcock
University of Waikato
New Zealand
shane.alcock@waikato.ac.nz

RADIUS

- AAA protocol for managing network access
  - Authentication, Authorization, Accounting

- Commonly used by network operators
  - But not the **only** option, so may not apply to everyone

RADIUS

- From an IP interception perspective, RADIUS gives us:
  - IP address assignments for our target users
    - Can follow users who change addresses
  - Session state updates to produce IRIs

RADIUS

- RADIUS is handled much like SIP for VoIP intercepts
  - RADIUS traffic must be mirrored into the collector
    - Authentication AND Accounting messages required
  - RADIUS server IP and port must be pre-configured

RADIUS

- RADIUS traffic is processed separately
  - Maintain IP session status for all users
    - Including users that are NOT targets
  - Inform capture threads which IPs are used by the targets
    - Traffic for those IPs will then be intercepted
  - RADIUS messages for targets must generate IRIs

Identity in RADIUS

- User-Name
  - The most conventional method

- Calling-Station-Id
  - Used as an alternative in some networks
  - User-Name is then set to a default value for all customers

Sessions in RADIUS

- Acct-Session-Id
  - Unique per RADIUS session
  - If this changes, then the CIN must change as well

RADIUS Intercept Scenario

- We operate a RADIUS server at 203.122.255.140
  - Access on UDP port 1645
  - Accounting on UDP port 1646
  - We use User-Name for identity in RADIUS

- We have received an IP intercept warrant
  - Target is the user: `b4CPidYn7u8Vesbo`
  - User is an ADSL customer

Configuring the RADIUS Server

- This is exactly like configuring a SIP server
  - Ideally, you would have configured this when you deployed OpenLI

- POST the RADIUS server details to the provisioner via REST API
  - http://<PROVIP>:<RESTAPIPORT>/radiusserver

Configuring the RADIUS Server

- RADIUS server objects look just like SIP server objects
  - Our server listens on two ports, so we'll need two API requests

```
{
    "ipaddress": "203.122.255.140",
    "port": "1645"
}

{
    "ipaddress": "203.122.255.140",
    "port": "1646"
}
```

Configuring the RADIUS Server

- Using curl to push the request to the provisioner
  - Don’t forget to repeat for port 1646!

```
curl -X POST -H "Content-Type: application/json"
  -d ‘{
        “ipaddress”: “203.122.255.140”,
        “port”: “1645”
      }’
  http://172.19.0.3:8080/radiusserver


<html><body>OpenLI provisioner configuration was successfully
updated.</body></html>
```

Configuring the RADIUS Server

- In the collector logs, we should now see

```
openlicollector[110]: OpenLI: collector has added 203.122.255.140-1645 to
its RADIUS core server list.
openlicollector[110]: OpenLI: collector has added 203.122.255.140-1646 to
its RADIUS core server list.
```

Removing a RADIUS Server

- Again, just like disabling a SIP server
  - Server must be expressed as <address> - <port>

```
curl -X DELETE http://172.19.0.3:8080/radiusserver/203.122.255.140-1645
```

REST API for RADIUS IP Intercepts

- Adding a RADIUS IP intercept via REST
  - Same API as other IP intercepts
  - Slightly different set of JSON object properties

http://<PROVIP>:<RESTAPIPORT>/ipintercept

REST API for RADIUS IP Intercepts

- The JSON object for our RADIUS IP intercept

```
{
    “liid”: “RADIUS003”,
    “authcc”: “NZ”,
    “delivcc”: “NZ”,
    “mediator”: 1,
    “agencyid”: “mocklea”,
    “starttime”: 0,
    “endtime”: 0,
    “user”: “b4CPidYn7u8Vesbo”,
    “accesstype”: “adsl”
}
```

REST API for RADIUS IP Intercepts

- The JSON object for our RADIUS IP intercept

```
{
    “liid”: “RADIUS003”,
    “authcc”: “NZ”,
    “delivcc”: “NZ”,
    “mediator”: 1,
    “agencyid”: “mocklea”,
    “starttime”: 0,
    “endtime”: 0,
    “user”: “b4CPidYn7u8Vesbo”,
    “accesstype”: “adsl”
}
```

REST API for RADIUS IP Intercepts

- The JSON object for our RADIUS IP intercept

```
{
    “liid”: “RADIUS003”,
    “authcc”: “NZ”,
    “delivcc”: “NZ”,
    “mediator”: 1,
    “agencyid”: “mocklea”,
    “starttime”: 0,
    “endtime”: 0,
    “user”: “b4CPidYn7u8Vesbo”,
    “accesstype”: “adsl”
}
```

REST API for RADIUS IP Intercepts

- Use `radiusident` to configure user identity matching
  - Set to `user` to match on User-Name only
  - Set to `csid` to match on Calling-Station-Id only
  - If not set, then OpenLI will match on either field
    - This is more work for OpenLI, so try to be specific if you can

REST API for RADIUS Intercepts

- Using curl to add the intercept on the provisioner

```
curl -X POST -H "Content-Type: application/json"
  -d '{
        "liid": "RADIUS003",
        "authcc": "NZ",
        "delivcc": "NZ",
        "mediator": 1,
        "agencyid": "mocklea",
        "starttime": 0,
        "endtime": 0,
        "user": "b4CPidYn7u8Vesbo",
        "accesstype": "adsl",
        "radiusident": "user"
    }'
  http://172.19.0.3:8080/ipintercept
```

REST API for RADIUS Intercepts

- We see an entry in the collector logs...

```
openlicollector[110]: OpenLI: received IP intercept for target
b4CPidYn7u8Vesbo from provisioner (LIID RADIUS003, authCC NZ, start time 0,
end time 0)
```

REST API for RADIUS Intercepts

- And also in the mediator logs

```
openlimediator[9279]: OpenLI Mediator: received "Activated" HI1 Notification
from provisioner for LIID RADIUS003 (target agency = mocklea)
```

REST API for RADIUS Intercepts

- Other REST methods work the same as with static IP intercepts
  - PUT, DELETE and GET
  - Refer to previous lesson if you've forgotten the syntax

One Last Thing

- To avoid confusion, remove the previous static IP intercept
  - Overlap in addressing in the static and RADIUS test traffic

```
curl -X DELETE http://172.19.0.3:8080/ipintercept/STATIC002
```

Next Time

- Perform a RADIUS intercept

- Inspect the resulting IRI records