NANOG 76
HACKATHON RECAP

Syed Ahmed
Hackathon Recap

• 8th hackathon organized by NANOG
• Roughly had 50 participants
• Working in 10 different teams
• A lot of college students
• Bunch of first timers
Hackathon Goals

• Theme of hackathon was deploying active monitoring solution
• Extract topology info using some kind of automation
• Build network map using topology info
• Calculate all possible best path between end points
• Probe all possible best paths and account for failure as it happens
Experience

• Day started with explaining hack, lab topology
• We gave a quick of overview of tools and protocols that might help to come up with solution
• It was interesting to notice a lot of participants have heard about exabgp but never got chance to use it.
• Some of the participants have not heard about BGP – LS and SCAPY.
• Teams worked on hack from 10:30 to 6pm
• At 6pm each team presented their work
• Winner was decided based on voting
Experience (contd.)

• We also encourage participants to bring their own project
• Project that got resurrected from NANOG67. PiCon: Console servers from RPi’s
• Winner of this hackathon are
  • NetBuffs
  • picon: Console servers from RPi's
Team: **NETBUFFS**
University of Colorado Boulder
Thank you!
Problem Statement

• Build a system to automate network monitoring and failure detection

• Reduce time to identify network and application problems

• Automate reporting of the issue based on the detected failure

• Make use of open source tools
Hackathon Goals

- Extract topology information (number of all possible paths and hops)
- Active monitoring of network paths for failures
- Automatically report failures
Below information of each link was parsed from routes.txt:

- Neighbor ID
- Neighbor address
- Interface ID
- Interface Address

Tools Used:
- Python
- Json
- ExaBGP
Topology

- Fetched all possible paths from end points
- Tool used: NetworkX
Create and send Probes → Receive Probes → Calculate Time Difference ($tdiff$) → Create a ticket (Freshdesk)

- Delay > threshold
- Delay < threshold

Continuous monitoring
Probe creation

- Create IP GRE headers to trace all the paths.
- Used SCAPY to create the packets.
- Send the packet from source to destination.
import smtplib

def print_return_packet_details(x):
    # sendTime, seq, probe, total = unpack('dlll', x.load)
    # print (x.Time, sendTime, seq, probe, total)

    print(x.load)
    print(x.time)
    threshold = 1
    difference = float(x.time) - float(x.load)
    if (float(x.time) - float(x.load)) < threshold:
        print("Delay in this route is higher than threshold. Creating a new ticket for debugging!")

fromaddr = 'animeshgupta720@gmail.com'
toaddr = 'support@animeshgupta720help.freshdesk.com'
msg = 'There is a high latency on the R1-R2-R4-R6 route. Please take necessary action.'

server = smtplib.SMTP("smtp.gmail.com:587")
server.starttls()
username='animeshgupta720@gmail.com'
password=********
server.login(username, password)
server.sendmail(fromaddr, toaddr, msg)
#server.quit()
server.login(username, password)
File "/usr/lib/python2.7/smtplib.py", line 623, in login
raise SMTPAuthenticationError(code, resp)
smtplib.SMTPAuthenticationError: (534, '5.7.14 <https://account.vGdSrJrRnsJuYoI_JUsAHrvVQP6D4FF7628Alc6n9Huzg6o5ijj5swHGS787-yQh> Please\n5.7.14 log in via your web browser and then /mail/answer/78754 g41sm2326496uah.12 - gsmtp')
tesuto@dev1:~$ sudo python test4.py
1560115679.91
1560115679.98
Delay in this route is higher than threshold. Creating a new
1w ticket for debugging!
tesuto@dev1:~$ cat testing.py ^C
tesuto@dev1:~$
tesuto@dev1:~$
tesuto@dev1:~$
tesuto@dev1:~$
tesuto@dev1:~$ sudo vi test4.py
tesuto@dev1:~$ sudo python test4.py
1560115968.87
1560115968.93
Delay in this route is higher than threshold. Creating a new
1w ticket for debugging!
tesuto@dev1:~$
Delay in this route is higher than threshold. Creating a new ticket for debugging!

tesuto@dev1:~$ sudo python test4.py
1560115968.87
1560115968.93

Delay in this route is higher than threshold. Creating a new ticket for debugging!

tesuto@dev1:~$ sudo python test4.py
1560116123.1
1560116123.15

Delay in this route is higher than threshold. Creating a new ticket for debugging!

tesuto@dev1:~$ sudo python test5.py
1560115679.91
1560115679.98

Delay in this route is higher than threshold. Creating a new ticket for debugging!

tesuto@dev1:~$ sudo python test3.py
Report Failure

- Raised ticket and reported issue by integrating our script with a ticketing tool (Freshdesk)
Future Scope

- **Scalability**
  Scale this to a higher number of nodes

- **Traffic Engineering**
  Reroute traffic to different paths as per issue reported

- **Visualization**
  Use data visualization tools such as Grafana to create dashboards
Takeaway

- Tools – Scapy, NetworkX, Python, ExaBGP
- Power of Network Programmability
- Planning
- Team work
- Don’t give up!
Team Members

- Animesh Gupta (animesh.gupta@colorado.edu)
- Vibhum Chandorkar (vibhum.chandorkar@colorado.edu)
- Ameya Korgaonkar (ameya.korgaonkar@colorado.edu)
- Jose Dahlson Irenish Kumar (joir9977@colorado.edu)
- Apurva Bhiwapurkar (Apurva.bhiwapurkar@colorado.edu)
THANK YOU 😊
NANOG76 Active Monitoring - Picon

Common issues:

- Islands of console capability
- Poor documentation
- Little or no proactive testing
- Pain when needed but not present
Open Hyperterm..... It’s in the start menu..... the button in the lower corner of your screen
The Solution

Open-source self-discovery framework

- Commodity hardware
- Operate over untrusted network
- Single view of all capabilities
- Pick and implement advanced features over time
  - Concurrency, logging of console output, point-and-click access
All you need

USB Serial Adapter

Raspberry Pi 3

Less than $50 USD!
Why Pi?

- Inexpensive
- Linux based
- Puppet, Chef, Ansible
- Wide range of hardware/hack support
- Linux Authentication (kerberos, ssh keys, etc)
- 1G Ram, SD Card Storage, decent CPU

>
Initial Work

- Python3, Flask, Bootstrap
- Webserver+API
  - API for registration, UI for device list
- Self-registering agent
  - Reported available TTY, interfaces, addresses
  - SSH to central server
Next Steps

- Additional Web and UI Capabilities (configurability)
- Programmatic console access
- In-browser console access
- Proxy agent for joining other consoles (Cisco, Digi, Avocent, etc)
- Physical configurations for deployment such as cases, custom FTDI daughterboards, etc
2019-06-09 22:56:11,855 INFO: Starting PiCon Agent...
2019-06-09 22:56:11,856 INFO: Using endpoint http://172.28.10.2:5000/api/s, holdtime 300s, reporting interval 60
2019-06-09 22:56:12,010 DEBUG: http://172.28.10.2:5000 "POST /api/register HTTP/1.1" 200 88
2019-06-09 22:56:12,013 DEBUG: Sent JSON in POST body:
{
    "holdtime": 300,
    "hostname": "picon",
    "interfaces": {
        "eth0": {
            "addr": [
                "172.28.10.5"
            ],
            "state": true
        },
        "lo": {
            "addr": [],
            "state": false
        },
        "wlan0": {
            "addr": [
                "199.187.221.226",
                "2620:0:ce0:101:ba27:ebff:fe43:b34c"
            ],
            "state": true
        }
    },
    "ports": [
        "ttyUSB0"
    ],
    "sn": "000000003316e619"
}
2019-06-09 22:56:12,014 DEBUG: Received JSON in POST response:
{
    "status": "ok",
    "tunnel": {
        "port": 2220,
        "server": "172.28.10.3"
    }
}
2019-06-09 22:56:12,015 INFO: Starting SSH tunnel connection to 172.28.10.3:2220
<table>
<thead>
<tr>
<th>Hostname</th>
<th>Serial</th>
<th>Addresses</th>
<th>Ports</th>
<th>First seen</th>
<th>Last seen</th>
<th>Status</th>
</tr>
</thead>
</table>
| picon    | 0000000003316e619 | eth0
172.28.10.5
wlan0
199.187.221.226
2620::ce0:101:ba27:ebff:fe43:b34c | ttyUSB0
fw-1
### Network Interfaces

<table>
<thead>
<tr>
<th>Interface Name</th>
<th>State</th>
<th>Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>eth0</td>
<td>1</td>
<td>172.28.10.5</td>
</tr>
<tr>
<td>wlan0</td>
<td>1</td>
<td>199.187.221.226</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2620:0:ce0:101:ba27:ebff:fe43:b34c</td>
</tr>
</tbody>
</table>

### Serial Ports

<table>
<thead>
<tr>
<th>Port name</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>ttyUSB0</td>
<td>fw:1</td>
</tr>
<tr>
<td>ttyUSB1</td>
<td>available/future use</td>
</tr>
</tbody>
</table>
Console for port ttyUSB0 on host picon

Refresh  Request Exclusive

User <websockets.server.WebSocketServerProtocol object at 0x7fa7a30f5710> requested Exclusivity

3 users connected

Close  Save changes
Accomplishments

- Team gained valuable experience with:
  - Websockets package and async workflows
  - Interacting with tty via python serial module
  - More features of Bootstrap and rusty jquery skills
- Various UI tweaks and progress
- Websockets POC is embedded in codebase
Take-aways

Flask and Bootstrap simplify non-developer progress

Websockets and asynchronous IO do not

Trying to implement the UI workflow will take much more planning

Less-technical takeaways from the challenge
Future Work/Next Steps

- Integrate Websockets with console access
- Basic authentication and privilege system
- Identify hardware for POC deployment
- Extend/abstract configuration
- Proxy agent
Thanks

https://github.com/piconsole/picon

Thanks Tesuto, Oracle and NANOG for this event