NANOG 73 Hackathon

June 24th, 2018
DENVER, CO

Sponsored By

JUNIPER NETWORKS | Engineering Simplicity

WIRELESS INFORMATION

- **NANOG - This is 802.1X secured using the below user:pass (2.4Ghz and 5Ghz)**
- **NANOG - legacy - This is an open, unencrypted network (2.4Ghz and 5Ghz)**

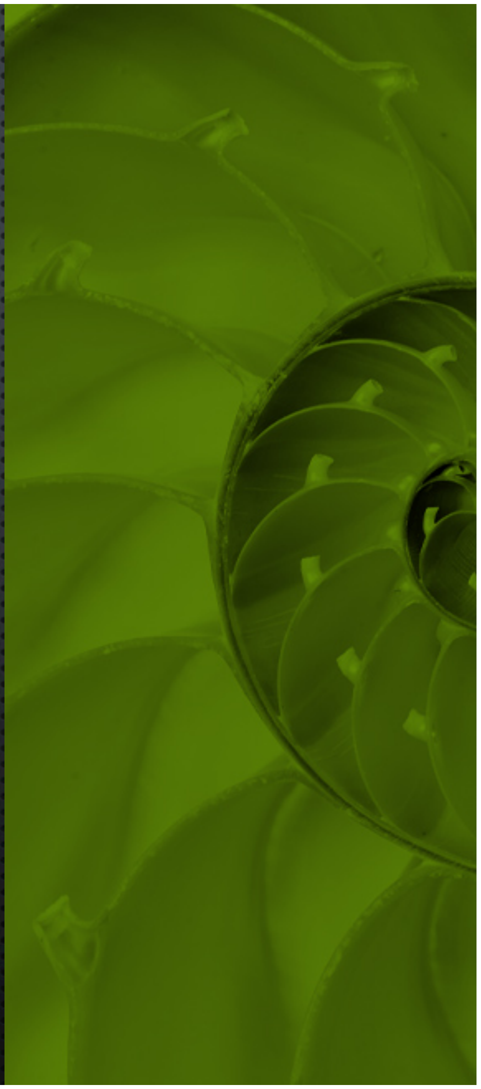
- **The login information for the 802.1x secured network is:**
- **User: nanog**
- **Pass: nanog**

GENERAL LOGISTICS

- Meals & Breaks – Mineral D-G
- Bathrooms – Near Elevators
- NANOG Registration – Centennial Foyer 4pm – 6pm
- Sunday Social Event - Cervantes Concert Hall – 6:30pm - 10:30pm
    - NANOG BADGE REQUIRED FOR ENTRY

AGENDA

- 8:30AM – 9:30AM REGISTRATION & BREAKFAST
- 9:30AM – 10:30AM OPENING/INTRODUCTION, AND TUTORIAL
- 10:30AM BREAK INTO GROUPS
- 12:30PM -1:30PM LUNCH
- 1:30PM RESUME GROUPS
- 3:00PM BREAK – REFRESHMENTS
- 6:00PM HACK DEADLINE, PROTOTYPE DEMOS, VOTING
- 6:50PM CLOSING & RAFFLE GIVEAWAY
- 7:00PM - 8:00PM HACKATHON RECEPTION



What to expect

- This workshop is designed to teach you how to integrate multi-vendor tools to create a cohesive security solution to protect your business services.
- This is **NOT** a workshop designed to teach you how to use a specific set of tools, and there will be help along the way to ensure you have a positive experience.

- Stages:
  - Phase 1: Familiarization with the tools – Single attacker
  - Phase 2: More volume and more IPs using automation
  - Phase 3: is targeted.

*Unnecessary services or permissions will be present.

And the winner is…

- Forensics
- How you stopped it or…
- How you would stop it
- You will vote on the best solution and presentation

What you should get out of this

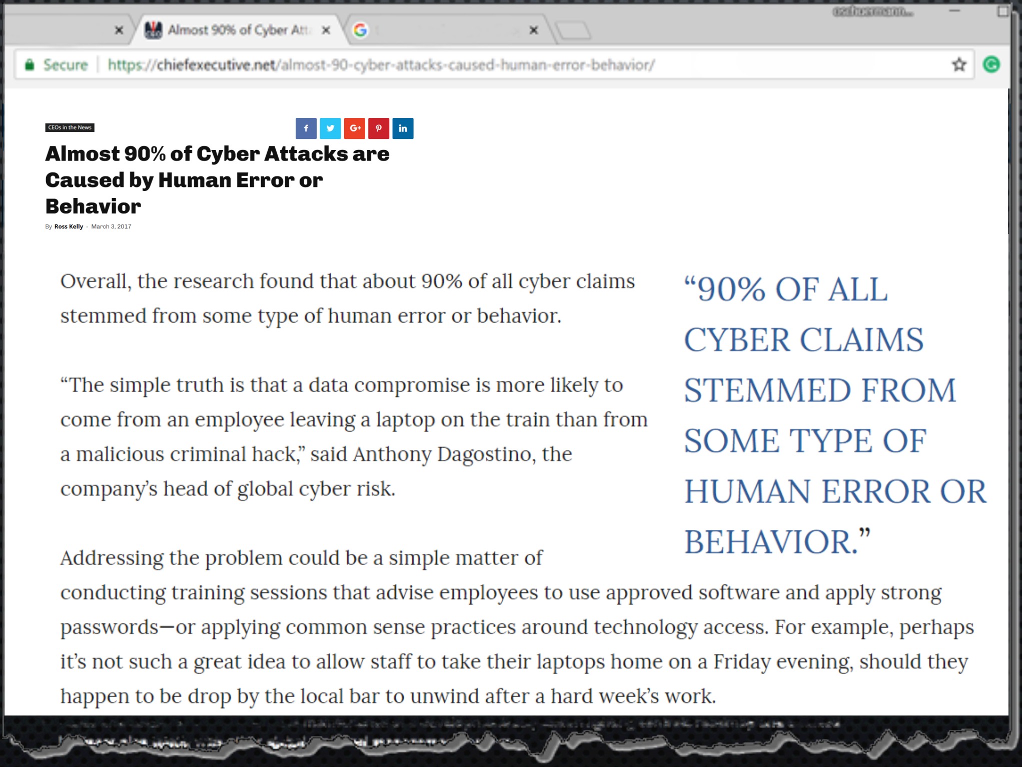
- Understanding that Automation plays a crucial part in security
- The Importance of Interoperability and Integration
- Scope of Automation for Security
- Security will require multiple parts of your organization to work together
- Understanding of DevSecOps

Devsecops

DevOpsqatestinfosec

- "In other words, when you hear "DevOps" today, you should probably be thinking *DevOpsQATestInfoSec*." - Gene Kim

Humans: Errors and Other bad behavior


Almost 90% of Cyber Att
Secure
https://chiefexecutive.net/almost-90-cyber-attacks-caused-human-error-behavior/
CEOs in the News

Almost 90% of Cyber Attacks are Caused by Human Error or Behavior

By Ross Kelly · March 3, 2017

Overall, the research found that about 90% of all cyber claims stemmed from some type of human error or behavior.

"The simple truth is that a data compromise is more likely to come from an employee leaving a laptop on the train than from a malicious criminal hack," said Anthony Dagostino, the company's head of global cyber risk.

"90% OF ALL CYBER CLAIMS STEMMED FROM SOME TYPE OF HUMAN ERROR OR BEHAVIOR."

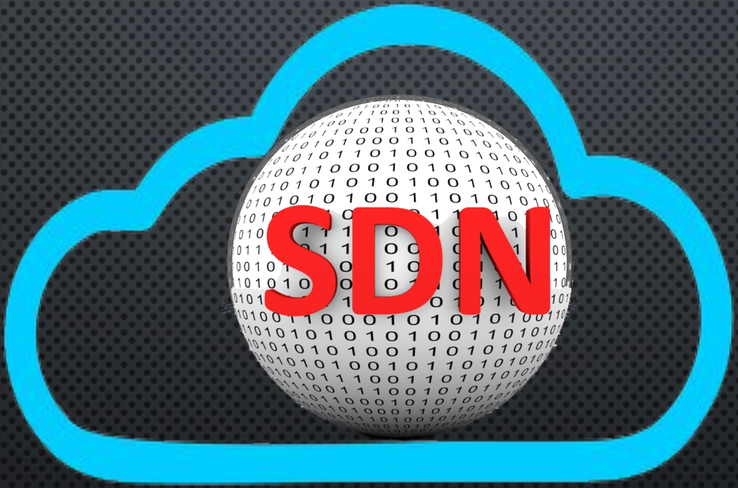
Addressing the problem could be a simple matter of conducting training sessions that advise employees to use approved software and apply strong passwords—or applying common sense practices around technology access. For example, perhaps it's not such a great idea to allow staff to take their laptops home on a Friday evening, should they happen to be drop by the local bar to unwind after a hard week's work.

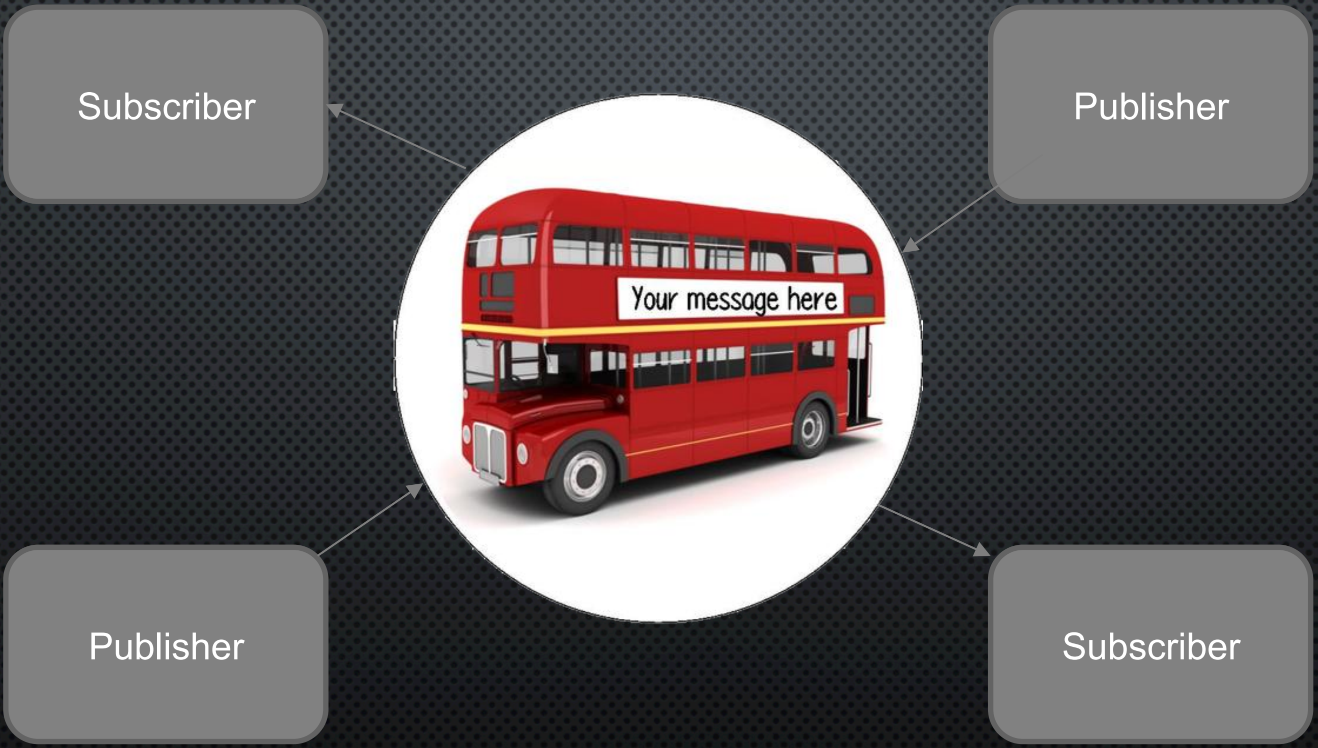
The Cloud

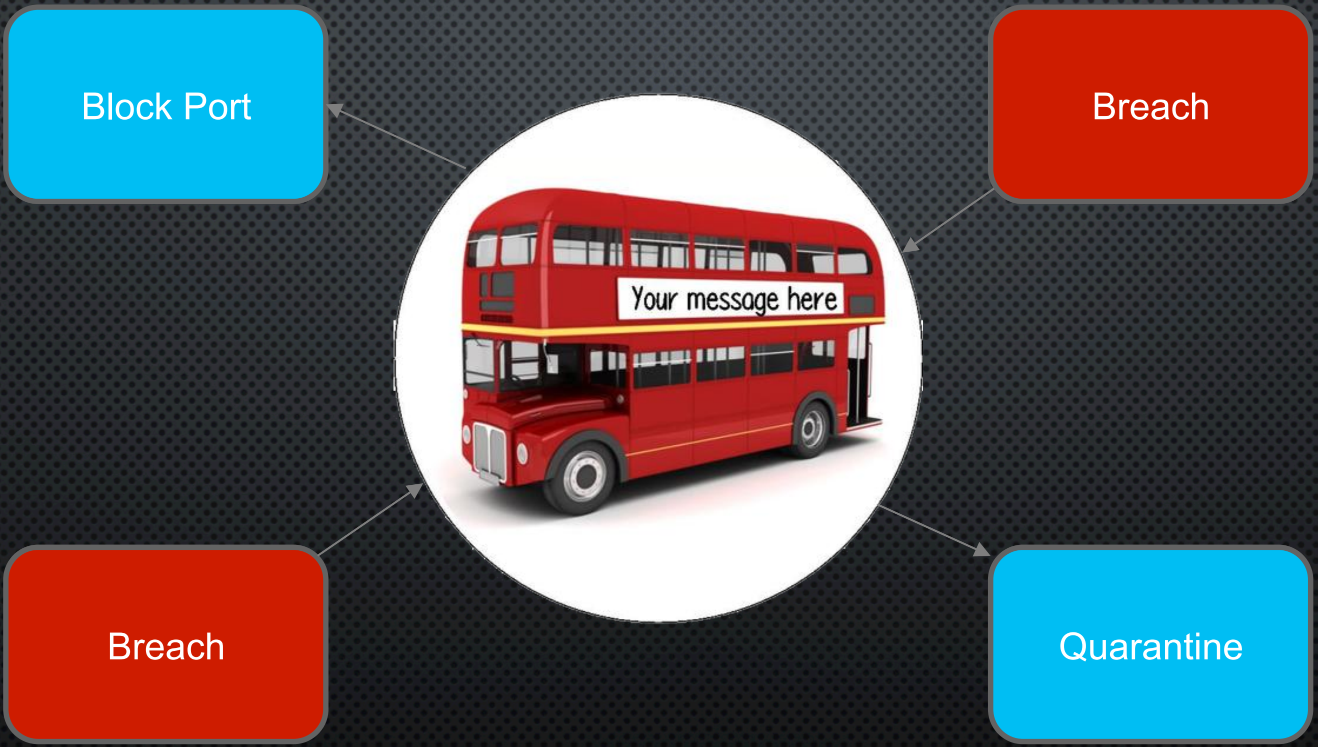
THERE IS NO CLOUD.
IT'S JUST SOMEBODY ELSE'S COMPUTER.

Getting lots of calls now?

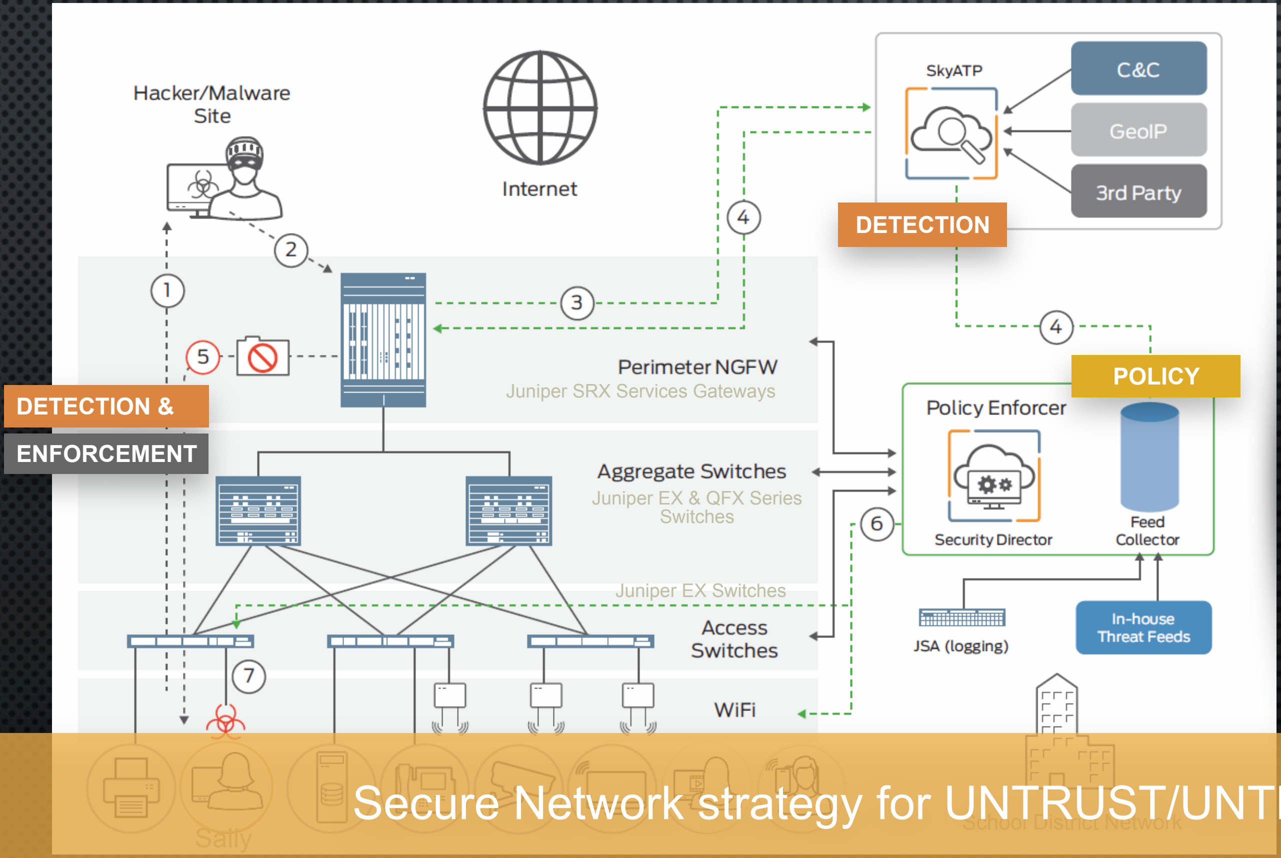
WE CARRY OUR OWN CONTRACTS
$99 DOWN DELIVERS TAX LICENSE WEEKLY
FINE USED
100% FINANCING
599


SDN


Subscriber
Publisher
Your message here
Publisher
Subscriber


Block Port
Breach
Your message here
Breach
Quarantine

YOUR NETWORK COULD BE THE FIREWALL



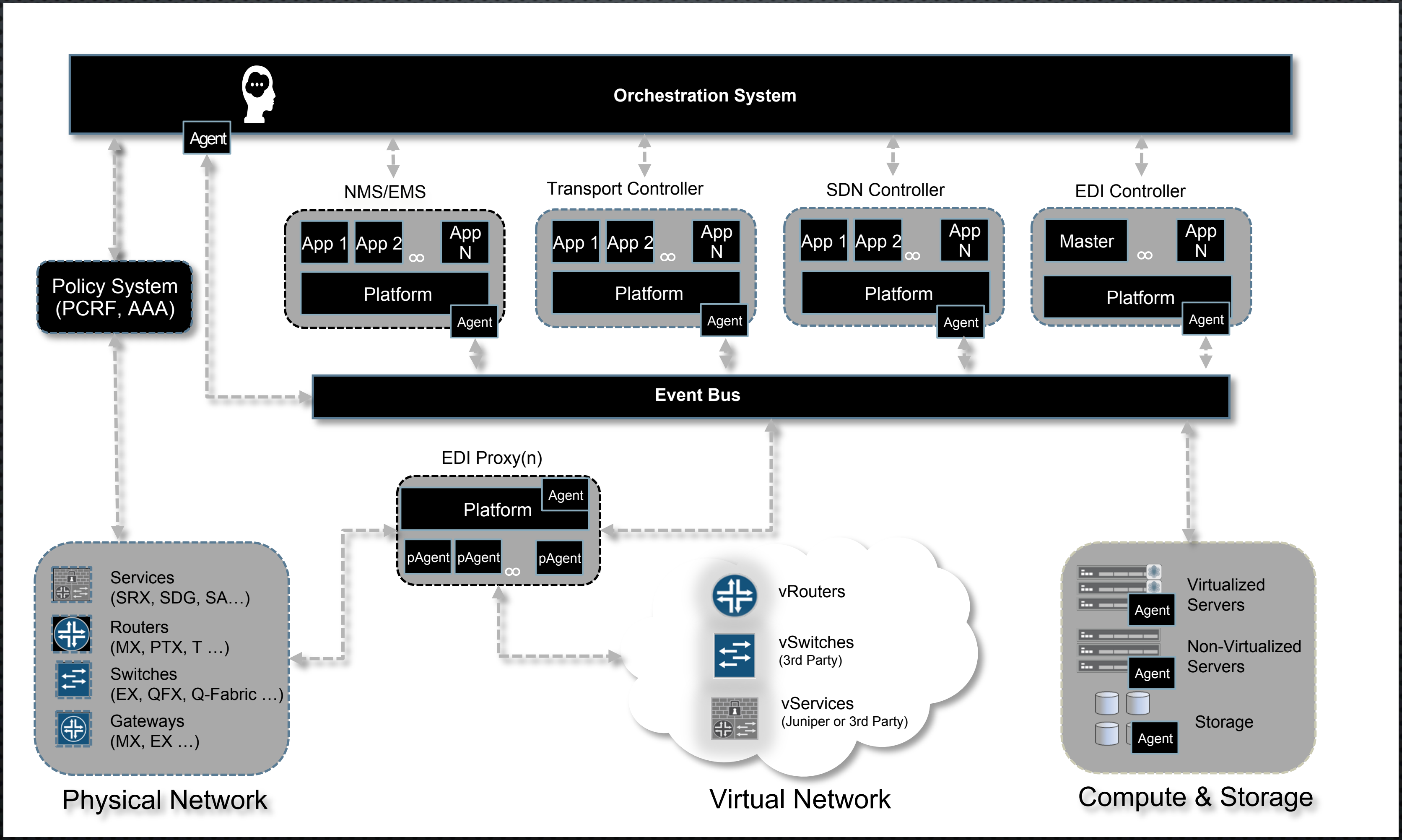
End-to-end visibility districts need to secure the entire network

Instant threat intelligence and detection

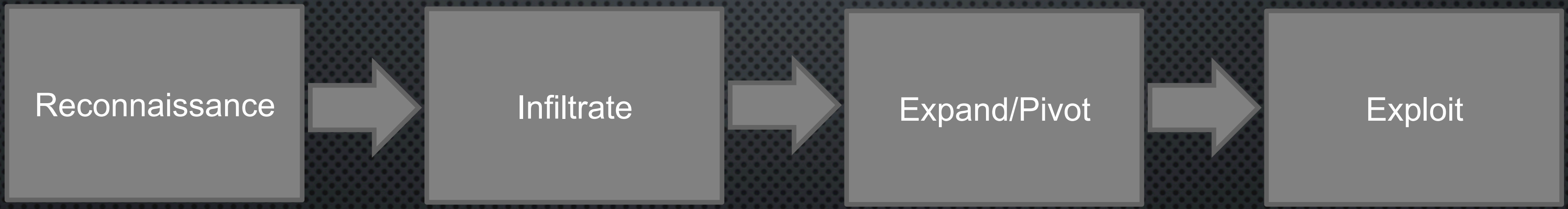
Dynamically adapting policy, deployed in real-time

Enforce security everywhere

Secure Network strategy for UNTRUST/UNTRUST Model


Orchestration System
Agent
Policy System (PCRF, AAA)
NMS/EMS
App 1
App 2
App N
Platform
Agent
Transport Controller
App 1
App 2
App N
Platform
Agent
SDN Controller
App 1
App 2
App N
Platform
Agent
EDI Controller
Master
App N
Platform
Agent
Event Bus
EDI Proxy(n)
Platform
Agent
pAgent
pAgent
pAgent
Services (SRX, SDG, SA…)
Routers (MX, PTX, T …)
Switches (EX, QFX, Q-Fabric …)
Gateways (MX, EX …)
Physical Network
vRouters
vSwitches (3rd Party)
vServices (Juniper or 3rd Party)
Virtual Network
Virtualized Servers
Agent
Non-Virtualized Servers
Agent
Storage
Agent
Compute & Storage

Anatomy of an attack

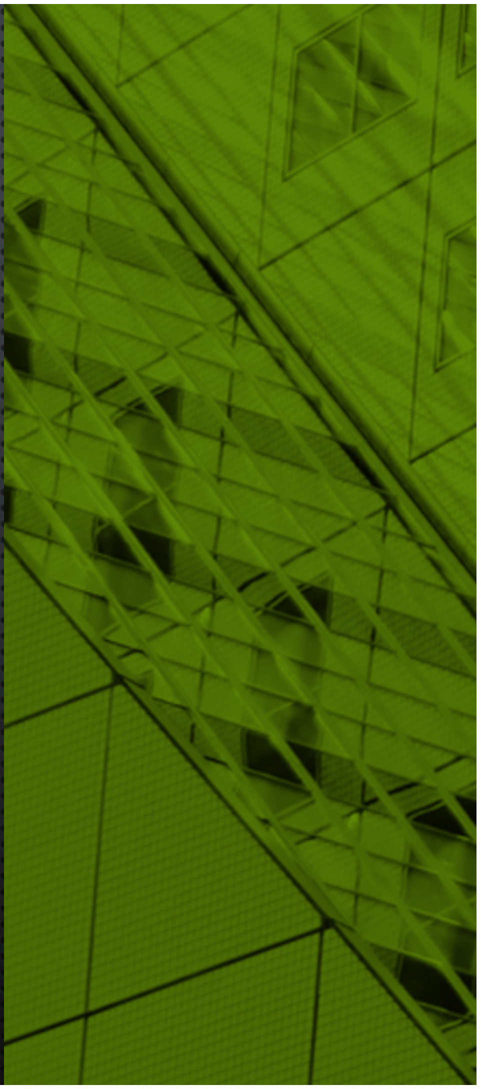


Today’s Event

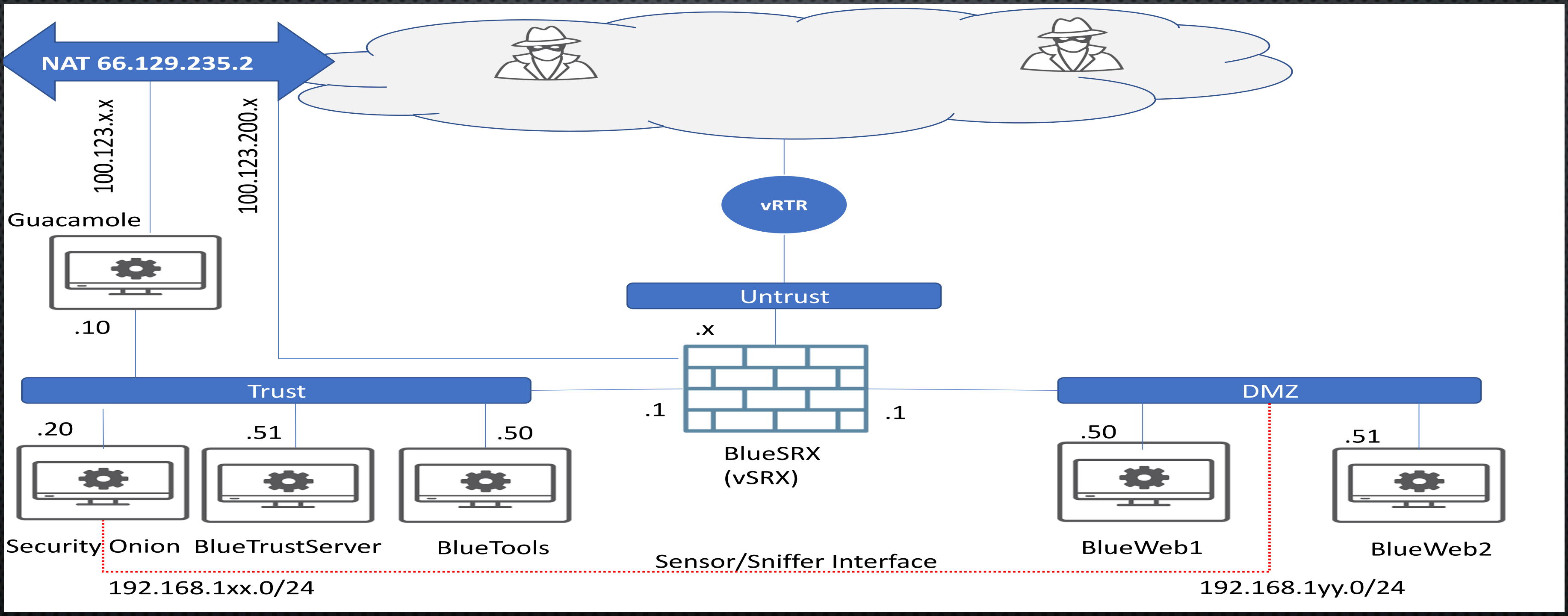
- Focus is on Defending Your Assets
- 7-Hour Event progressing in Difficulty
- Blue Teams (You) must prevent Red Teams (Us) from manipulating your security posture and stopping your business services
- Blockers will be in place for increased difficulty and to teach you additional security measures
- You will select tools from a list
- More than one correct method
- We will assist you if you get stuck
- Red team will be partially automated as well as staffed by Juniper Networks™/NANOG

NANOG Hackathon

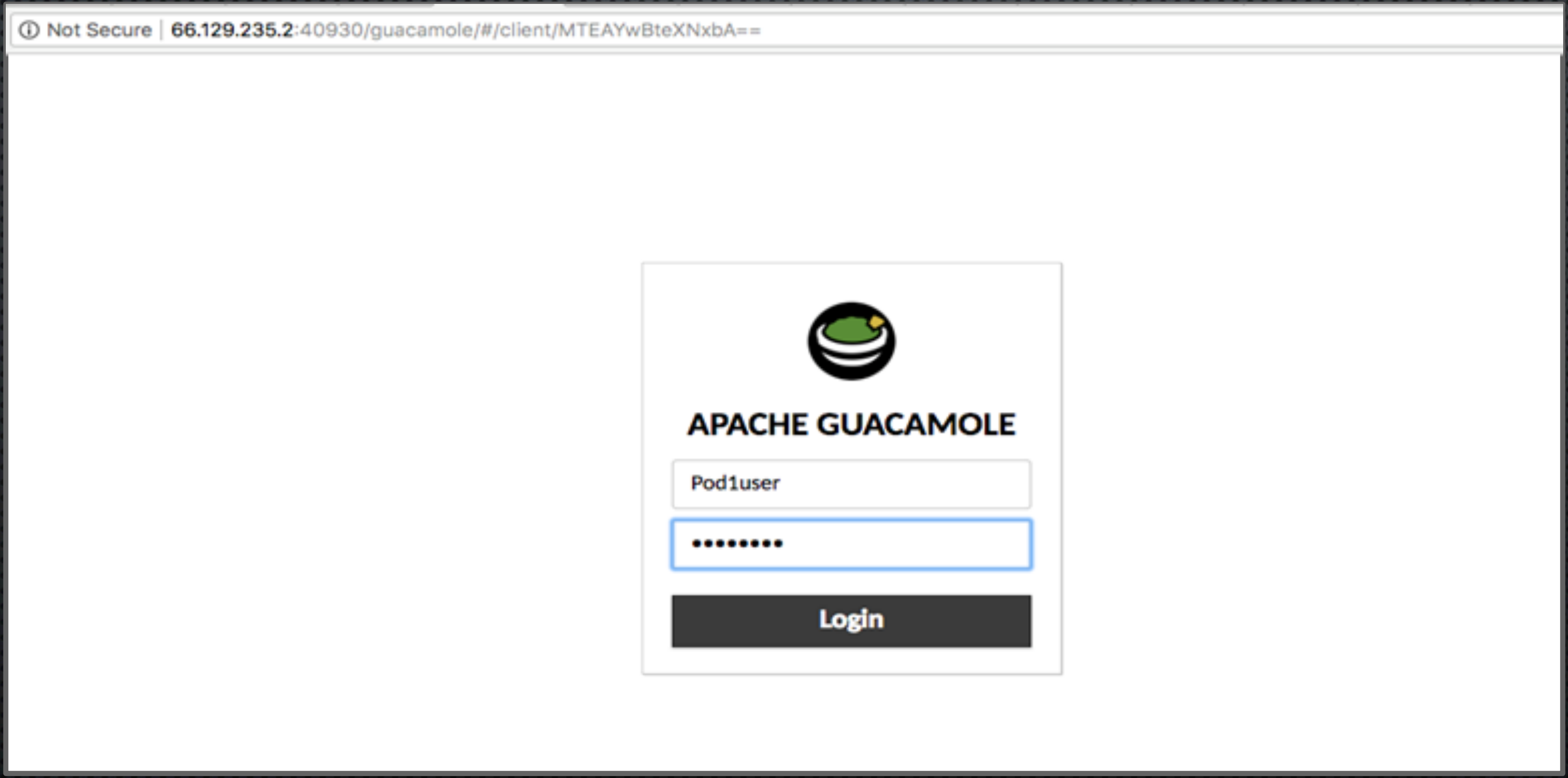
Rules of Engagement , Topology and Tools document



Blue Team Topology



Accessing Blue POD:
https://66.129.235.2:40930/guacamole

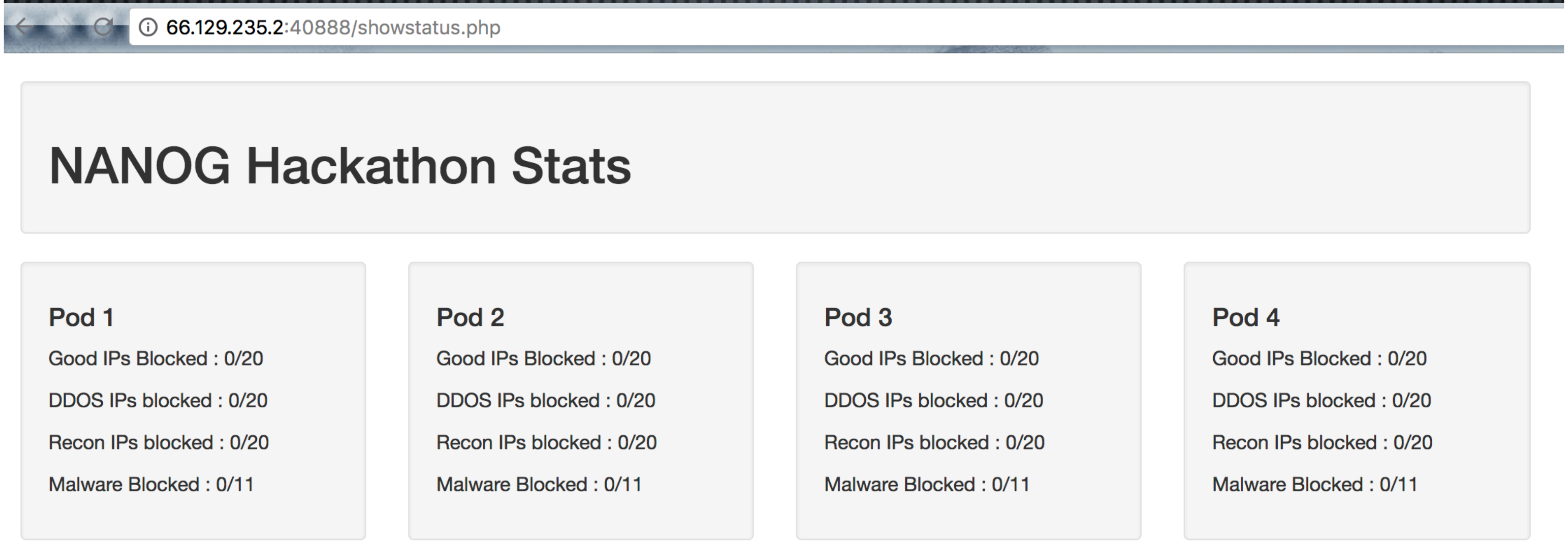

ⓘ Not Secure | 66.129.235.2:40930/guacamole/#/client/MTEAYwBteXNxbA==

APACHE GUACAMOLE

Pod1user

•••••••••

Accessing Dashboard: HTTP://66.129.235.2:40888/SHOWSTATUS.PHP



NANOG Hackathon Stats

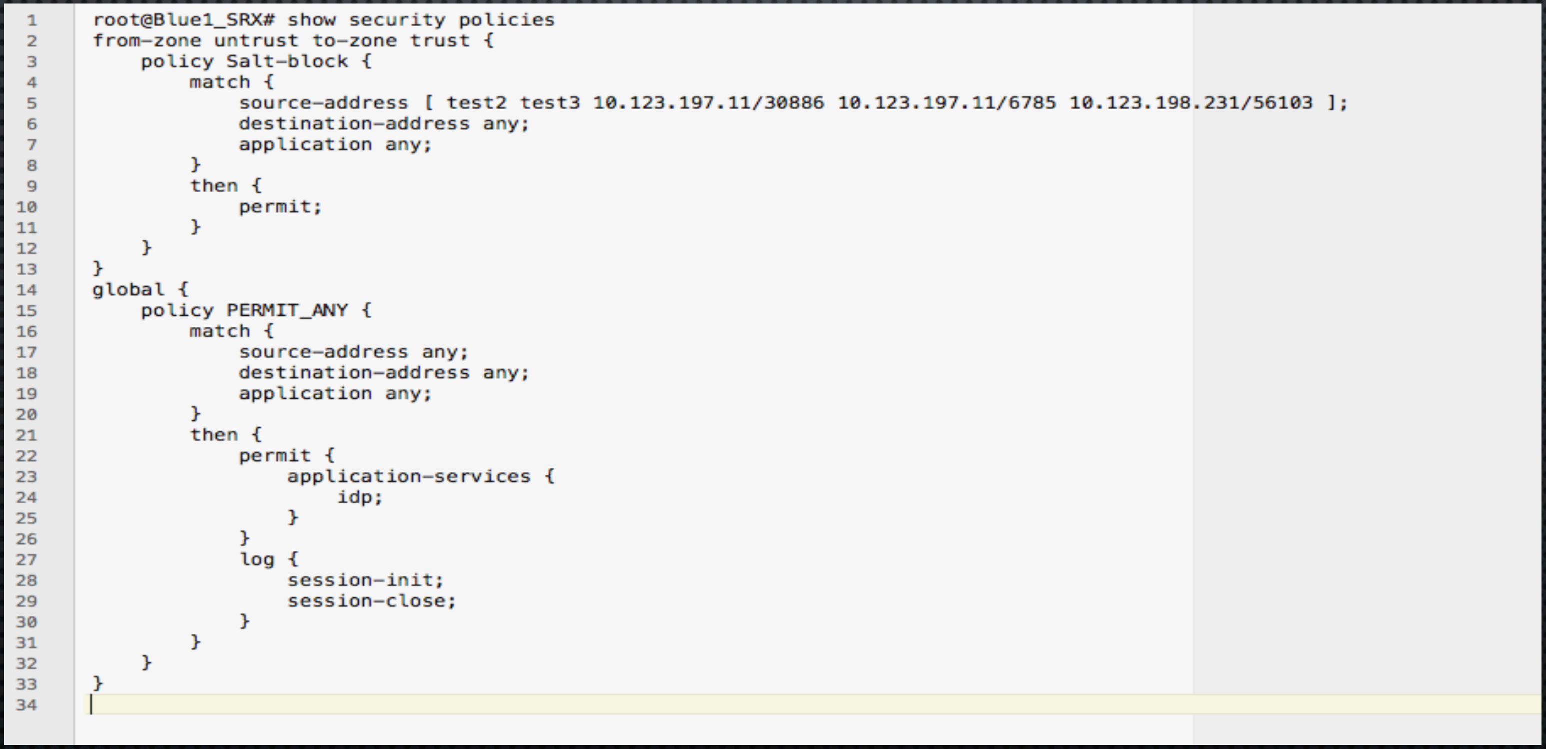
Pod 1	Pod 2	Pod 3	Pod 4
Good IPs Blocked : 0/20	Good IPs Blocked : 0/20	Good IPs Blocked : 0/20	Good IPs Blocked : 0/20
DDOS IPs blocked : 0/20	DDOS IPs blocked : 0/20	DDOS IPs blocked : 0/20	DDOS IPs blocked : 0/20
Recon IPs blocked : 0/20	Recon IPs blocked : 0/20	Recon IPs blocked : 0/20	Recon IPs blocked : 0/20
Malware Blocked : 0/11	Malware Blocked : 0/11	Malware Blocked : 0/11	Malware Blocked : 0/11

Credentials

Accessing Blue POD: https://66.129.235.2:40930/guacamole

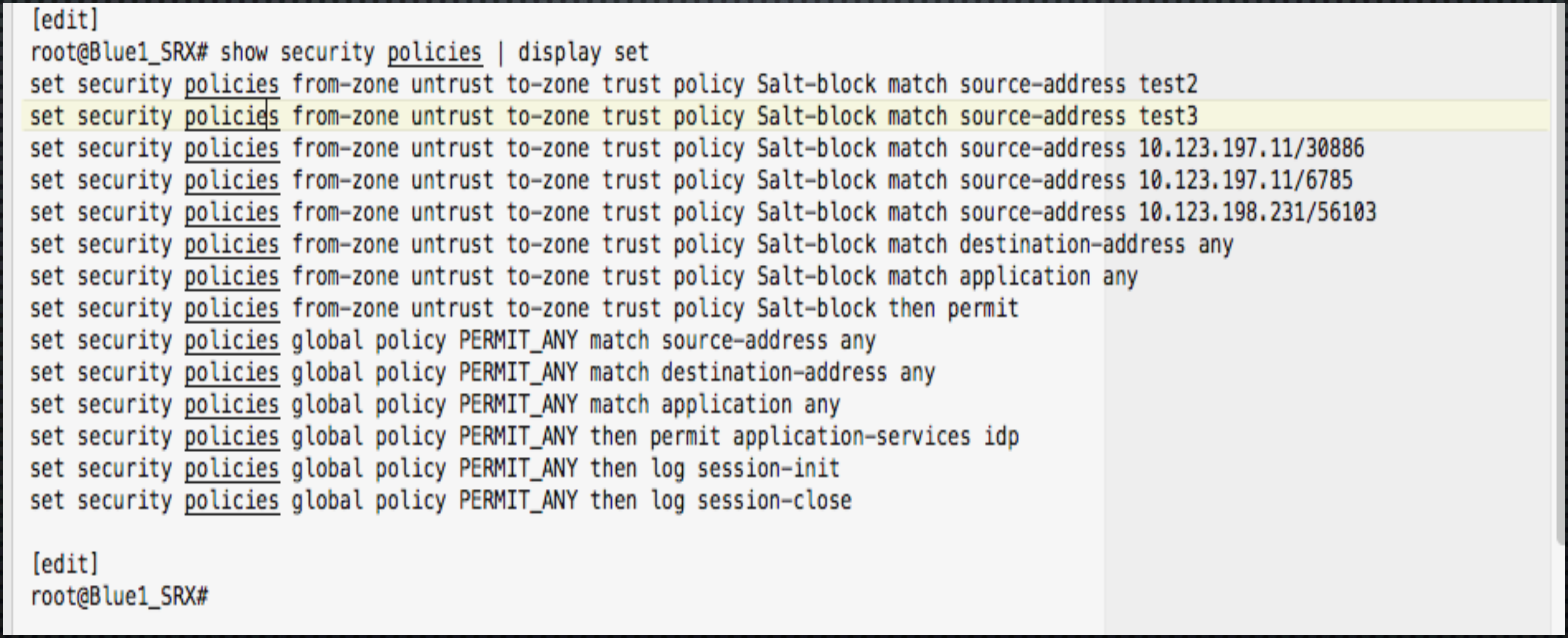
Device/App	Username	Password
Guacamole	PodXuser	Nanog18
Blue Servers (DMZ & Trust)	auser	root123
SRX	root	root123
SecurityOnion	auser	root123

Basic SRX Policy - Stanza



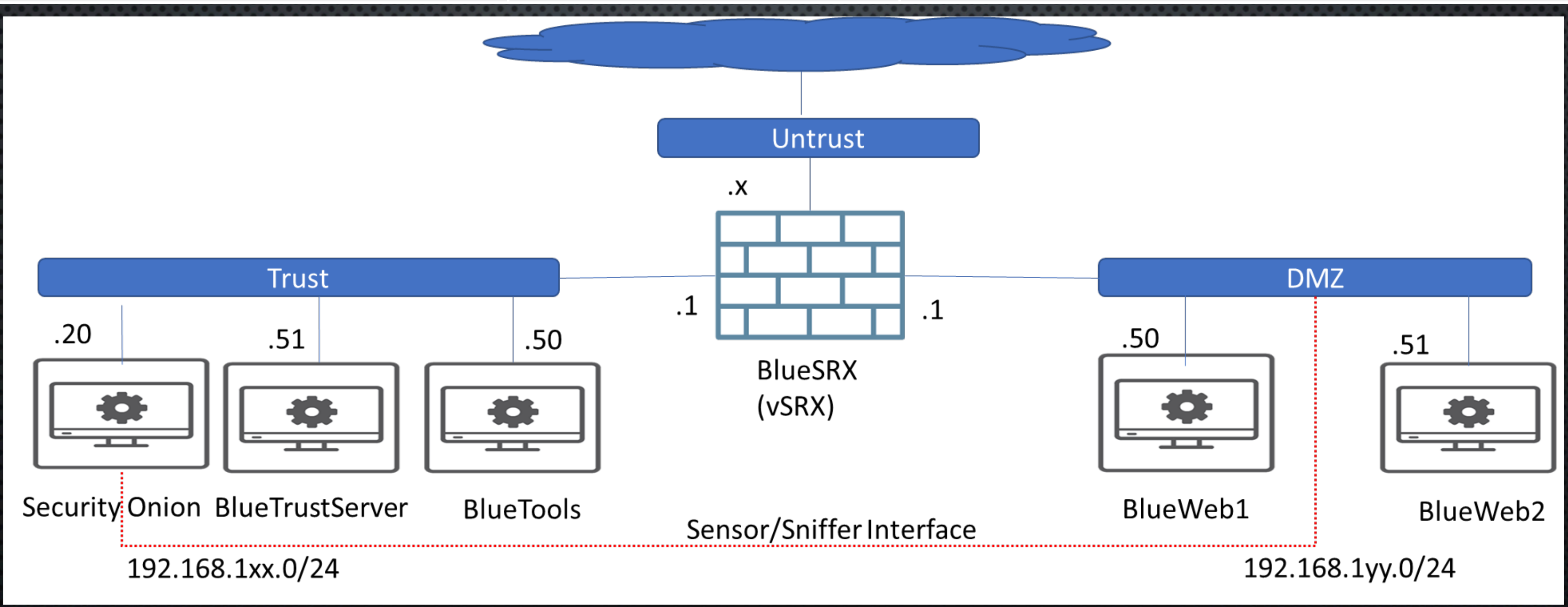
```
root@Blue1_SRX# show security policies
from-zone untrust to-zone trust {
    policy Salt-block {
        match {
            source-address [ test2 test3 10.123.197.11/30886 10.123.197.11/6785 10.123.198.231/56103 ];
            destination-address any;
            application any;
        }
        then {
            permit;
        }
    }
}
global {
    policy PERMIT_ANY {
        match {
            source-address any;
            destination-address any;
            application any;
        }
        then {
            permit {
                application-services {
                    idp;
                }
            }
            log {
                session-init;
                session-close;
            }
        }
    }
}
```

Basic SRX Policy – Display Set



```
[edit]
root@Blue1_SRX# show security policies | display set
set security policies from-zone untrust to-zone trust policy Salt-block match source-address test2
set security policies from-zone untrust to-zone trust policy Salt-block match source-address test3
set security policies from-zone untrust to-zone trust policy Salt-block match source-address 10.123.197.11/30886
set security policies from-zone untrust to-zone trust policy Salt-block match source-address 10.123.197.11/6785
set security policies from-zone untrust to-zone trust policy Salt-block match source-address 10.123.198.231/56103
set security policies from-zone untrust to-zone trust policy Salt-block match destination-address any
set security policies from-zone untrust to-zone trust policy Salt-block match application any
set security policies from-zone untrust to-zone trust policy Salt-block then permit
set security policies global policy PERMIT_ANY match source-address any
set security policies global policy PERMIT_ANY match destination-address any
set security policies global policy PERMIT_ANY match application any
set security policies global policy PERMIT_ANY then permit application-services idp
set security policies global policy PERMIT_ANY then log session-init
set security policies global policy PERMIT_ANY then log session-close

[edit]
root@Blue1_SRX#
```

From Zone	To Zone	Service
global	global	ping, traceroute
untrust	dmz	http, https, alt_http (8000 – 9000), ftp, scp
dmz	trust	http, https, alt_http (8000 – 9000), ftp, scp
Trust	Untrust	internet access – play videos, facebook, etc



Blue Team Tools:

1. Security Onion
2. Salt-Stack with J-EDI

(These two tools are pre-installed and setup on the Blue Team POD, participants are welcome to install any other tools of their choice)

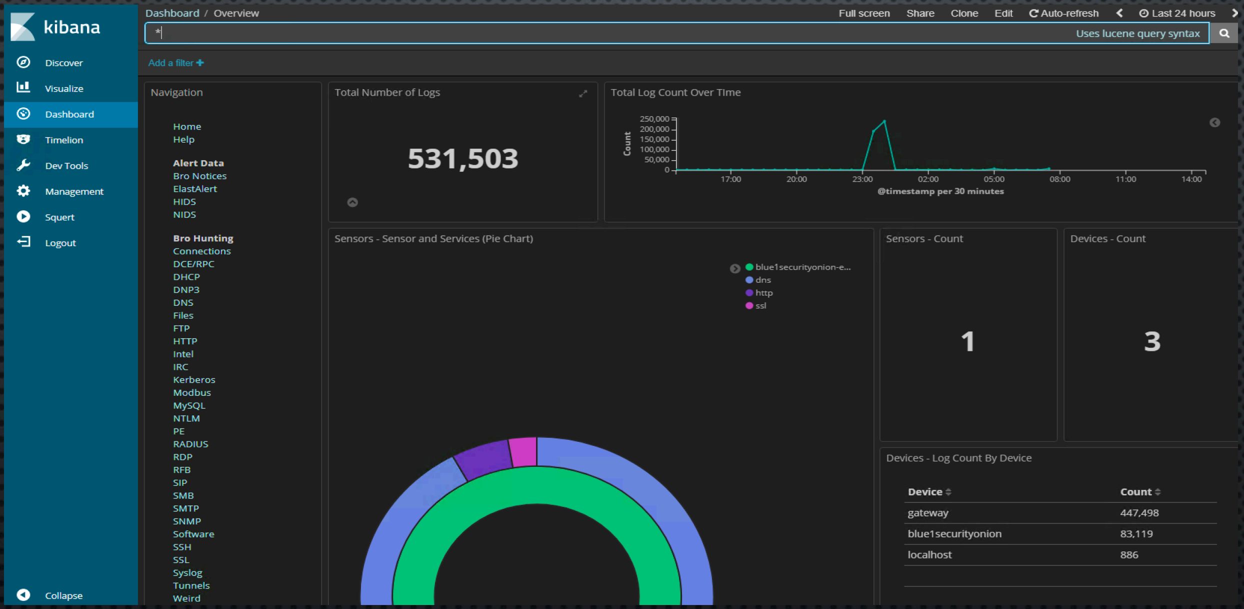
Security Onion

Introduction to the tool

- Security Onion is a free and open source Linux distribution for intrusion detection, enterprise security monitoring, and log management. It includes Elasticsearch, Logstash, Kibana, Snort, Suricata, Bro, OSSEC, Sguil, Squert, NetworkMiner, and many other security tools.
- **Accessing the Tool :**
- Security Onion's Kibana user interface can be accessed from the POD by browsing to the URL https://192.168.XXX.20/app/kibana

- **Credentials :**
- Username : auser
- Password : root123

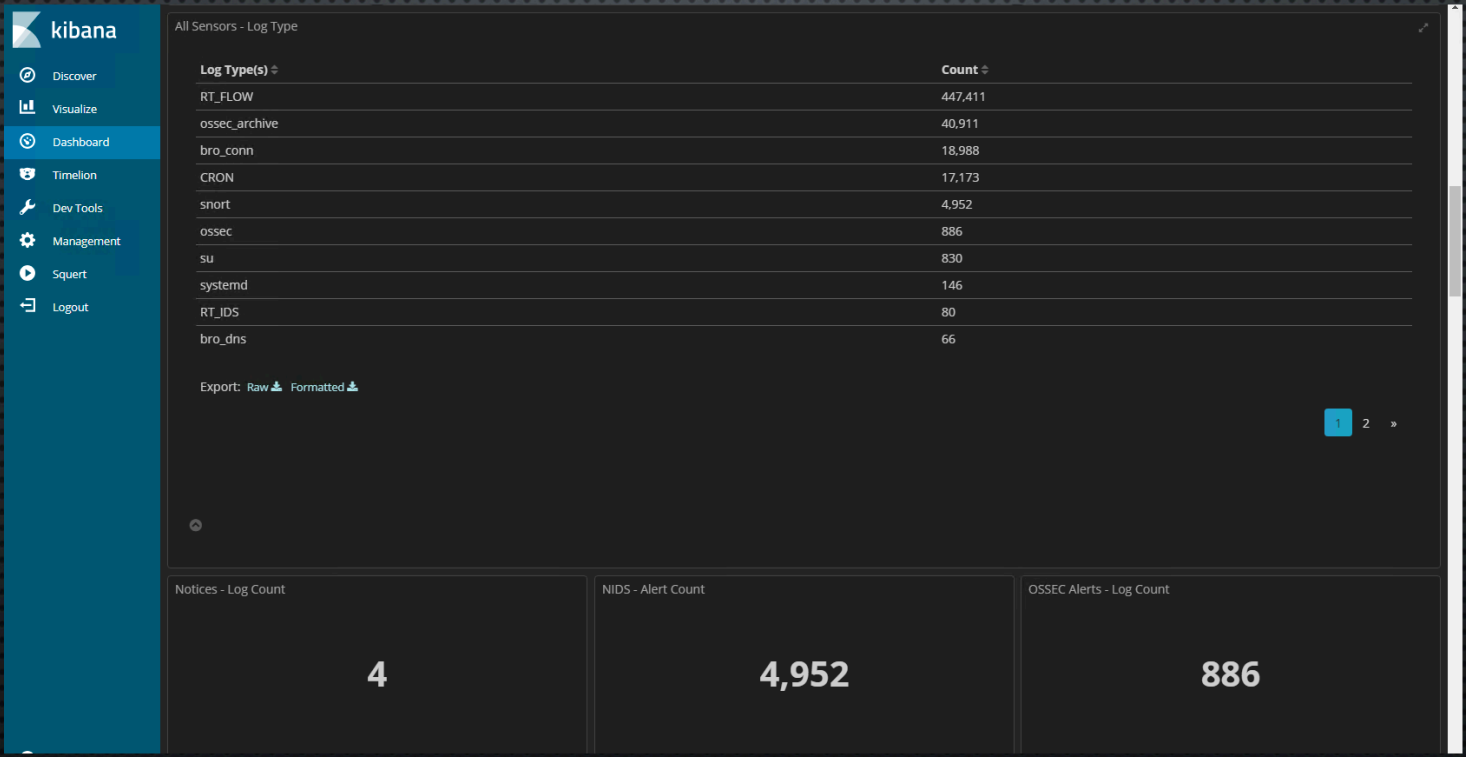
Security onion dashboard



Device	Count
gateway	447,498
blue1securityonion	83,119
localhost	886

Security Onions UI provides easy access to the alert , host , session , transaction data .
Participant can navigate using this dashboard **dashboard** and get all sort of alert metrices .

Security onion dashboard



All Sensors - Log Type

Log Type(s)	Count
RT_FLOW	447,411
ossec_archive	40,911
bro_conn	18,988
CRON	17,173
snort	4,952
ossec	886
su	830
systemd	146
RT_IDS	80
bro_dns	66

Export: Raw Formatted

1 2 »

Notices - Log Count

4

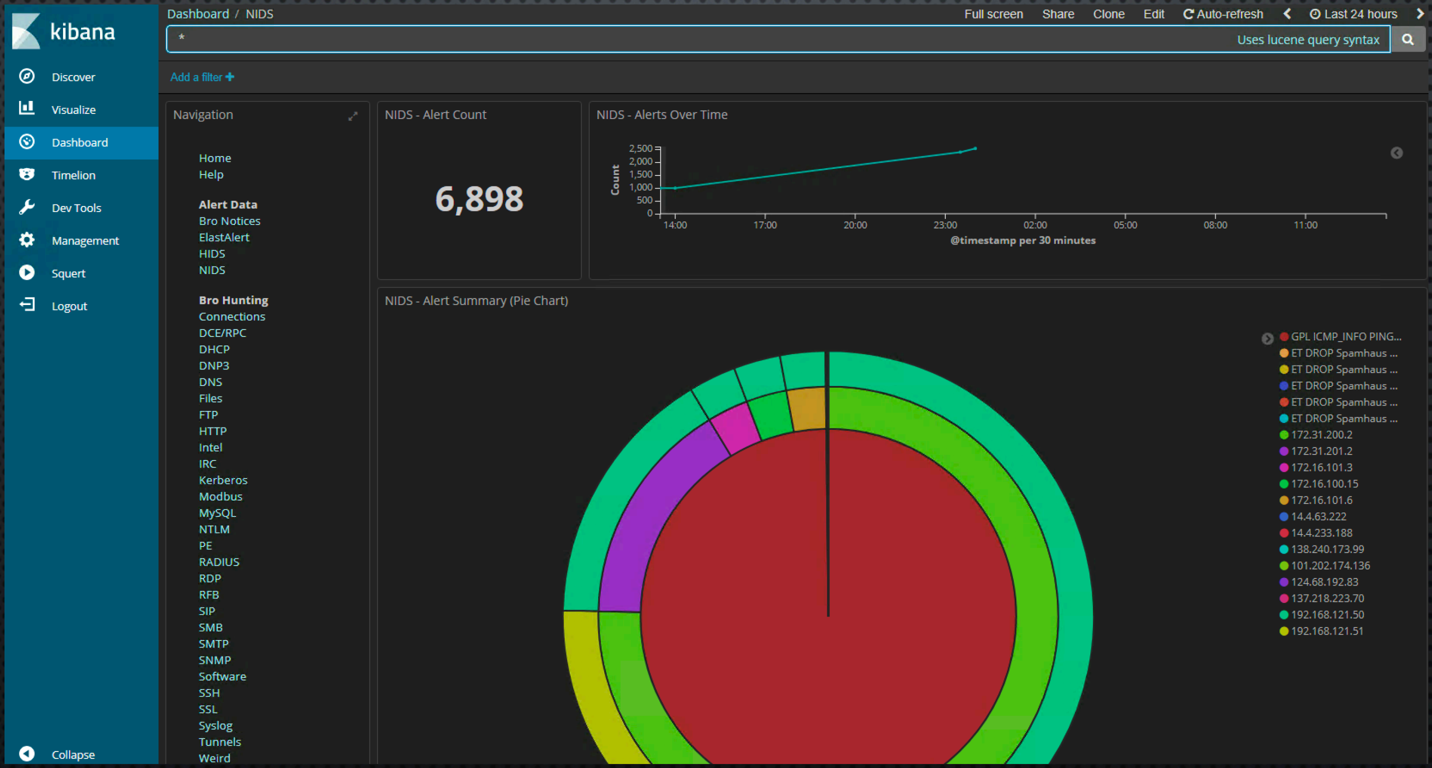
NIDS - Alert Count

4,952

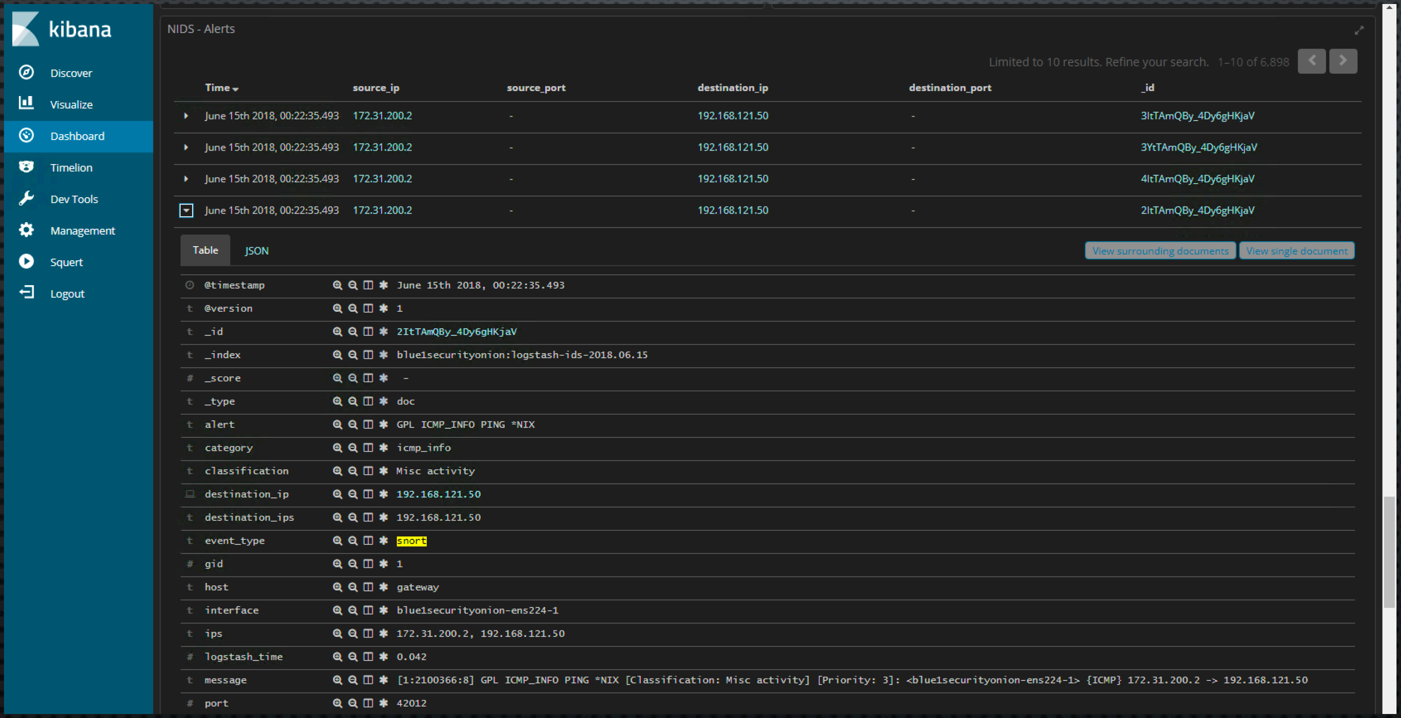
OSSEC Alerts - Log Count

886

Security onion dashboard



Security onion dashboard



NIDS - Alerts

Limited to 10 results. Refine your search. 1–10 of 6,898

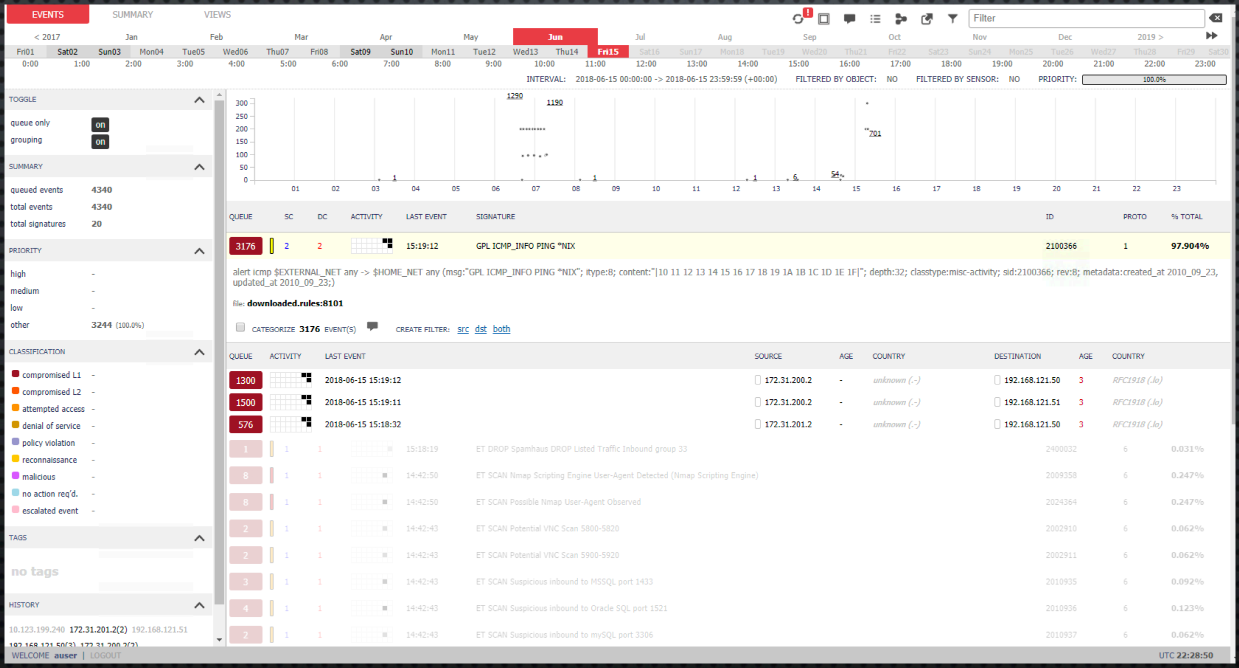
Time	source_ip	source_port	destination_ip	destination_port	_id
June 15th 2018, 00:22:35.493	172.31.200.2	-	192.168.121.50	-	3ItTAmQBy_4Dy6gHKjaV
June 15th 2018, 00:22:35.493	172.31.200.2	-	192.168.121.50	-	3YtTAmQBy_4Dy6gHKjaV
June 15th 2018, 00:22:35.493	172.31.200.2	-	192.168.121.50	-	4ItTAmQBy_4Dy6gHKjaV
June 15th 2018, 00:22:35.493	172.31.200.2	-	192.168.121.50	-	2ItTAmQBy_4Dy6gHKjaV

Table JSON View surrounding documents View single document

@timestamp	June 15th 2018, 00:22:35.493
@version	1
_id	2ItTAmQBy_4Dy6gHKjaV
_index	blue1securityonion:logstash-ids-2018.06.15
_score	-
_type	doc
alert	GPL ICMP_INFO PING *NIX
category	icmp_info
classification	Misc activity
destination_ip	192.168.121.50
destination_ips	192.168.121.50
event_type	snort
gid	1
host	gateway
interface	blue1securityonion-ens224-1
ips	172.31.200.2, 192.168.121.50
logstash_time	0.042
message	[1:2100366:8] GPL ICMP_INFO PING *NIX [Classification: Misc activity] [Priority: 3]: <blue1securityonion-ens224-1> {ICMP} 172.31.200.2 -> 192.168.121.50
port	42012

NIDS alert (continued) : More stats can be found at the bottom of the page to dig deeper into one log

SQUERT



INTERVAL: 2018-06-15 00:00:00 -> 2018-06-15 23:59:59 (+00:00) FILTERED BY OBJECT: NO FILTERED BY SENSOR: NO PRIORITY: 100.0%

TOGGLE

queue only	on
grouping	on

SUMMARY

queued events	4340
total events	4340
total signatures	20

PRIORITY

high	-
medium	-
low	-
other	3244 (100.0%)

CLASSIFICATION

🟥 compromised L1	-
🟧 compromised L2	-
🟧 attempted access	-
🟫 denial of service	-
🟪 policy violation	-
🟨 reconnaissance	-
🟪 malicious	-
🟦 no action req'd.	-
🟥 escalated event	-

TAGS

no tags

HISTORY

10.123.199.240 172.31.201.2(2) 192.168.121.51
192.168.121.50(3) 172.31.200.2(2)

QUEUE	SC	DC	ACTIVITY	LAST EVENT	SIGNATURE	ID	PROTO	% TOTAL
3176	2	2		15:19:12	GPL ICMP_INFO PING *NIX	2100366	1	97.904%

alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"GPL ICMP_INFO PING *NIX"; itype:8; content:"|10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F|"; depth:32; classtype:misc-activity; sid:2100366; rev:8; metadata:created_at 2010_09_23, updated_at 2010_09_23;)

file: **downloaded.rules:8101**

CATEGORIZE **3176** EVENT(S) CREATE FILTER: src dst both

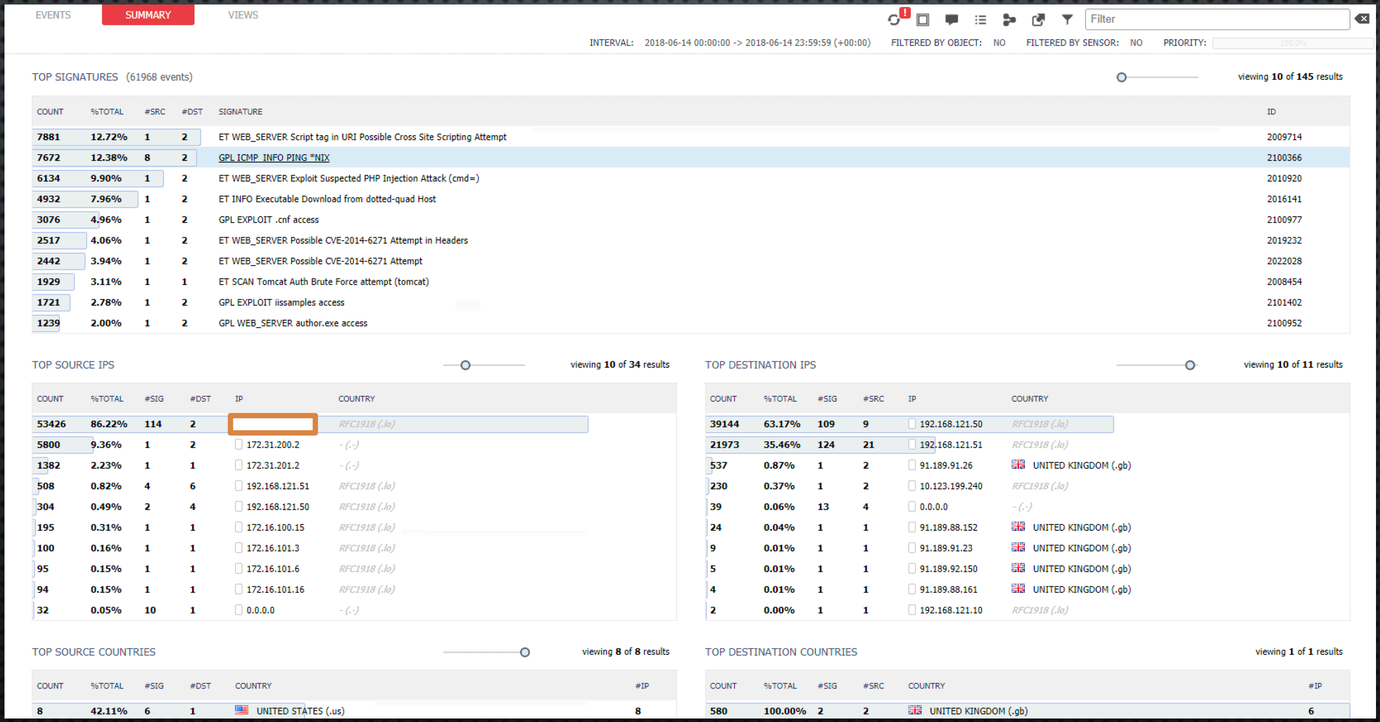
QUEUE	ACTIVITY	LAST EVENT	SOURCE	AGE	COUNTRY	DESTINATION	AGE	COUNTRY
1300		2018-06-15 15:19:12	172.31.200.2	-	unknown (.-)	192.168.121.50	3	RFC1918 (.lo)
1500		2018-06-15 15:19:11	172.31.200.2	-	unknown (.-)	192.168.121.51	3	RFC1918 (.lo)
576		2018-06-15 15:18:32	172.31.201.2	-	unknown (.-)	192.168.121.50	3	RFC1918 (.lo)

QUEUE	SC	DC	ACTIVITY	LAST EVENT	SIGNATURE	ID	PROTO	% TOTAL
1	1	1		15:18:19	ET DROP Spamhaus DROP Listed Traffic Inbound group 33	2400032	6	0.031%
8	1	1		14:42:50	ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine)	2009358	6	0.247%
8	1	1		14:42:50	ET SCAN Possible Nmap User-Agent Observed	2024364	6	0.247%
2	1	1		14:42:43	ET SCAN Potential VNC Scan 5800-5820	2002910	6	0.062%
2	1	1		14:42:43	ET SCAN Potential VNC Scan 5900-5920	2002911	6	0.062%
3	1	1		14:42:43	ET SCAN Suspicious inbound to MSSQL port 1433	2010935	6	0.092%
4	1	1		14:42:43	ET SCAN Suspicious inbound to Oracle SQL port 1521	2010936	6	0.123%
2	1	1		14:42:43	ET SCAN Suspicious inbound to mySQL port 3306	2010937	6	0.062%

WELCOME **auser** | LOGOUT UTC **22:28:50**

<u>SQUERT Dash-Board</u> (can be accessed from the left navigation pane of Kibana Dashboard)

SQUERT



INTERVAL: 2018-06-14 00:00:00 -> 2018-06-14 23:59:59 (+00:00) FILTERED BY OBJECT: NO FILTERED BY SENSOR: NO PRIORITY:

TOP SIGNATURES (61968 events) viewing **10** of **145** results

COUNT	%TOTAL	#SRC	#DST	SIGNATURE	ID
7881	12.72%	1	2	ET WEB_SERVER Script tag in URI Possible Cross Site Scripting Attempt	2009714
7672	12.38%	8	2	GPL ICMP_INFO PING *NIX	2100366
6134	9.90%	1	2	ET WEB_SERVER Exploit Suspected PHP Injection Attack (cmd=)	2010920
4932	7.96%	1	2	ET INFO Executable Download from dotted-quad Host	2016141
3076	4.96%	1	2	GPL EXPLOIT .cnf access	2100977
2517	4.06%	1	2	ET WEB_SERVER Possible CVE-2014-6271 Attempt in Headers	2019232
2442	3.94%	1	2	ET WEB_SERVER Possible CVE-2014-6271 Attempt	2022028
1929	3.11%	1	1	ET SCAN Tomcat Auth Brute Force attempt (tomcat)	2008454
1721	2.78%	1	2	GPL EXPLOIT iissamples access	2101402
1239	2.00%	1	2	GPL WEB_SERVER author.exe access	2100952

TOP SOURCE IPS viewing **10** of **34** results

COUNT	%TOTAL	#SIG	#DST	IP	COUNTRY
53426	86.22%	114	2	[redacted]	RFC1918 (.lo)
5800	9.36%	1	2	172.31.200.2	- (.-)
1382	2.23%	1	1	172.31.201.2	- (.-)
508	0.82%	4	6	192.168.121.51	RFC1918 (.lo)
304	0.49%	2	4	192.168.121.50	RFC1918 (.lo)
195	0.31%	1	1	172.16.100.15	RFC1918 (.lo)
100	0.16%	1	1	172.16.101.3	RFC1918 (.lo)
95	0.15%	1	1	172.16.101.6	RFC1918 (.lo)
94	0.15%	1	1	172.16.101.16	RFC1918 (.lo)
32	0.05%	10	1	0.0.0.0	- (.-)

TOP DESTINATION IPS viewing **10** of **11** results

COUNT	%TOTAL	#SIG	#SRC	IP	COUNTRY
39144	63.17%	109	9	192.168.121.50	RFC1918 (.lo)
21973	35.46%	124	21	192.168.121.51	RFC1918 (.lo)
537	0.87%	1	2	91.189.91.26	UNITED KINGDOM (.gb)
230	0.37%	1	2	10.123.199.240	RFC1918 (.lo)
39	0.06%	13	4	0.0.0.0	- (.-)
24	0.04%	1	1	91.189.88.152	UNITED KINGDOM (.gb)
9	0.01%	1	1	91.189.91.23	UNITED KINGDOM (.gb)
5	0.01%	1	1	91.189.92.150	UNITED KINGDOM (.gb)
4	0.01%	1	1	91.189.88.161	UNITED KINGDOM (.gb)
2	0.00%	1	1	192.168.121.10	RFC1918 (.lo)

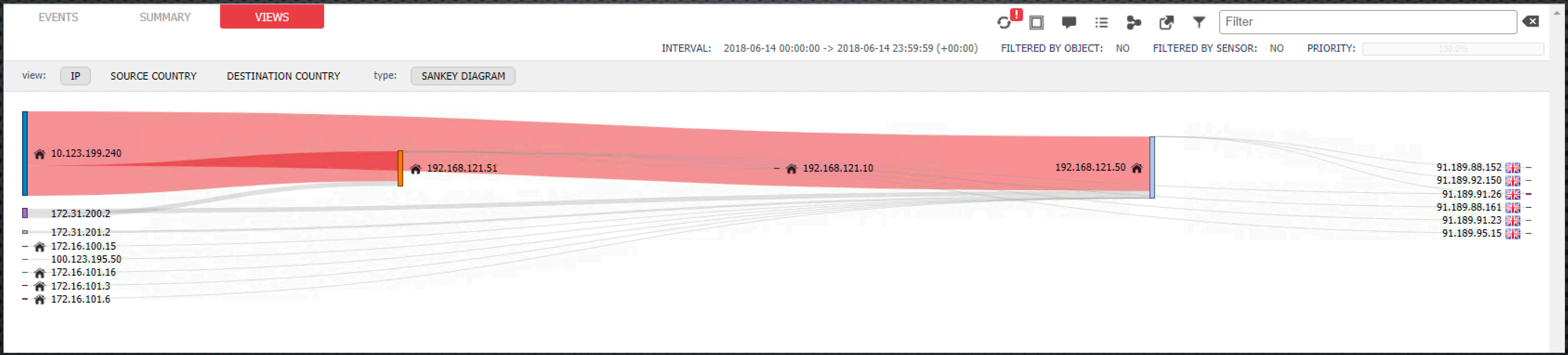
TOP SOURCE COUNTRIES viewing **8** of **8** results

COUNT	%TOTAL	#SIG	#DST	COUNTRY	#IP
8	42.11%	6	1	UNITED STATES (.us)	8

TOP DESTINATION COUNTRIES viewing **1** of **1** results

COUNT	%TOTAL	#SIG	#SRC	COUNTRY	#IP
580	100.00%	2	2	UNITED KINGDOM (.gb)	6

SQUERT



Links to browse to learn more about Security onion and Squert:

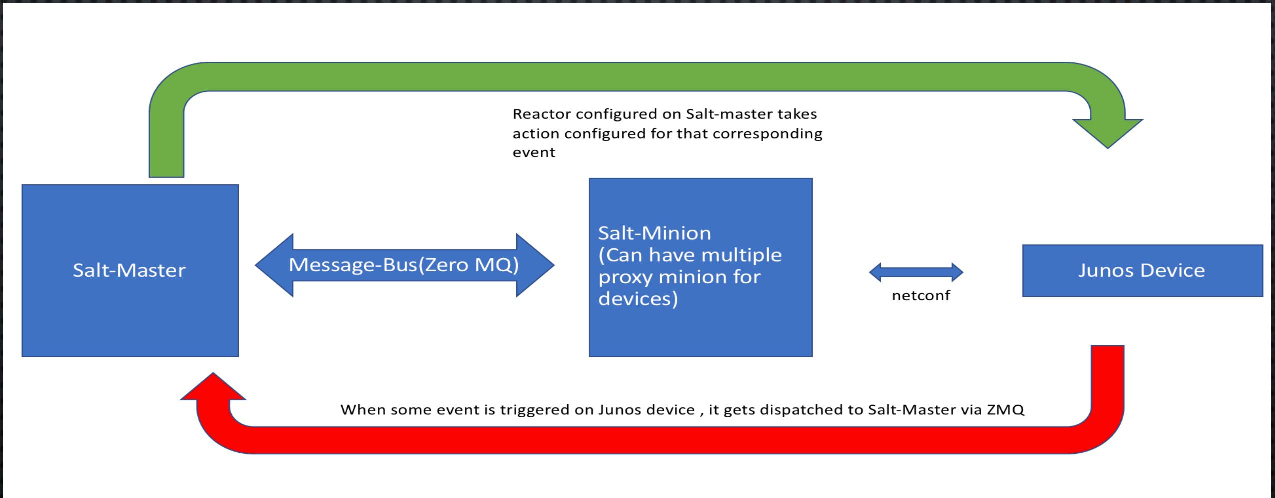
https://github.com/Security-Onion-Solutions/security-onion/

https://github.com/Security-Onion-Solutions/security-onion/wiki/Squert

Salt-Stack with J-EDI

Implementation in the POD

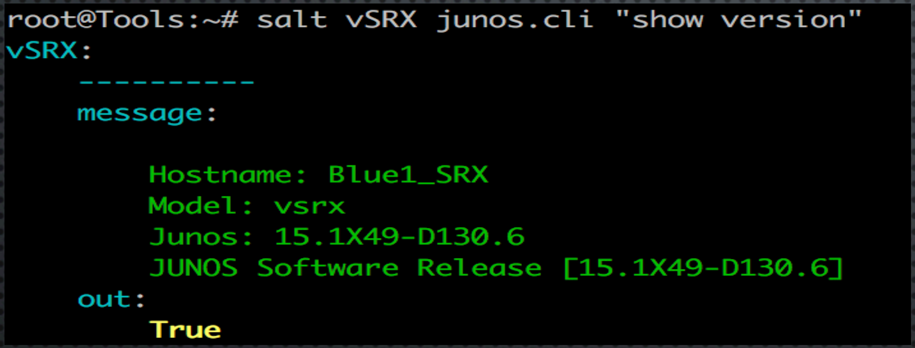
- Each Pod has a Salt-Master and Salt-minion monitoring the vSRX
- Salt-Master is setup on Tools server
- Salt-Minion is setup on Trust server


Reactor configured on Salt-master takes action configured for that corresponding event
Salt-Master
Message-Bus(Zero MQ)
Salt-Minion (Can have multiple proxy minion for devices)
netconf
Junos Device
When some event is triggered on Junos device , it gets dispatched to Salt-Master via ZMQ

Basic Salt Reactor System

Basic Junos commands and template

- SALT VSRX JUNOS.CLI "SHOW VERSION"



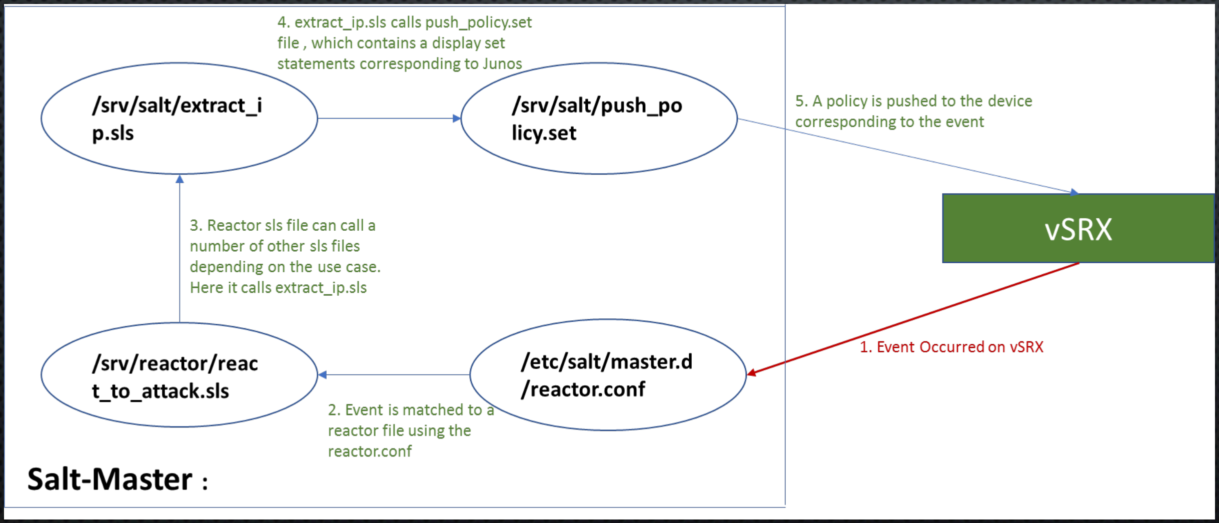
```
root@Tools:~# salt vSRX junos.cli "show version"
vSRX:
    ----------
    message:

        Hostname: Blue1_SRX
        Model: vsrx
        Junos: 15.1X49-D130.6
        JUNOS Software Release [15.1X49-D130.6]
    out:
        True
```

You can replace “show version” in the above command with Junos commands like “show interfaces”, “show configuration” and many other configuration & operational mode commands and get the results on Tools server .

- You can also configure the device from tools server using the “Junos set commands”. All you need to do is create a *commands*.set file and use it like this : tools~# **salt '*device_name*' junos.install_config ’path to .set file’**
- A sample of the “.set” is present in /**srv/salt/push_policy.set** location on the tools server. This file is being used in a reactor (*described later in this doc*).

Salt-reactor

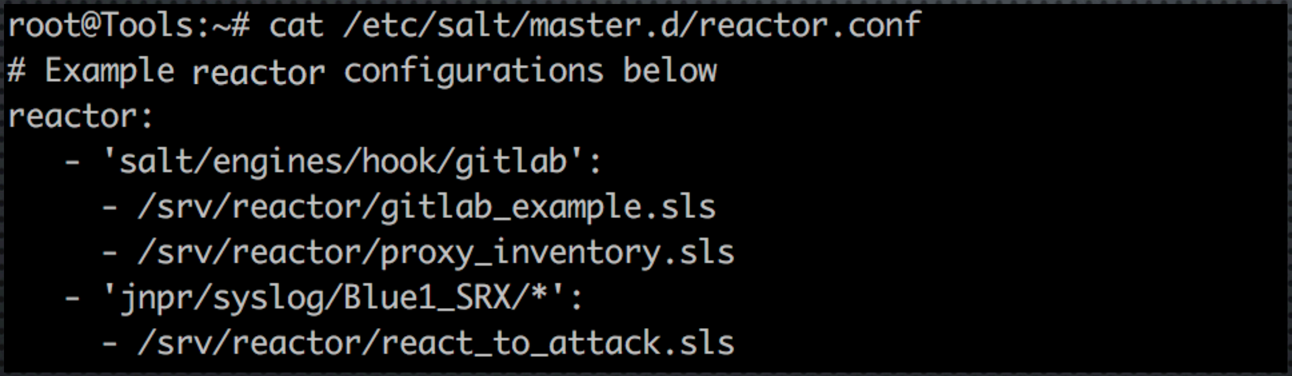
- The main purpose of the salt reactor is to listen to events taking place on the vSRX and react based on the actions already configured via Ansible , YAML , Python scripts already configured on the salt-master.
- A work flow of **which and how files on salt-master interact** corresponding to the event is described below :


4. extract_ip.sls calls push_policy.set file , which contains a display set statements corresponding to Junos
/srv/salt/extract_ip.sls
/srv/salt/push_policy.set
5. A policy is pushed to the device corresponding to the event
vSRX
3. Reactor sls file can call a number of other sls files depending on the use case. Here it calls extract_ip.sls
1. Event Occurred on vSRX
/srv/reactor/react_to_attack.sls
/etc/salt/master.d/reactor.conf
2. Event is matched to a reactor file using the reactor.conf
Salt-Master :

This diagram only show a single workflow of how salt reactor works. It is implemented in the POD assigned to the each team and the purpose of it to get participant familiar with Salt. Participants can create any number of workflows they want .

Salt: Files, their purpose and details

1. /etc/salt/master.d/reactor.conf

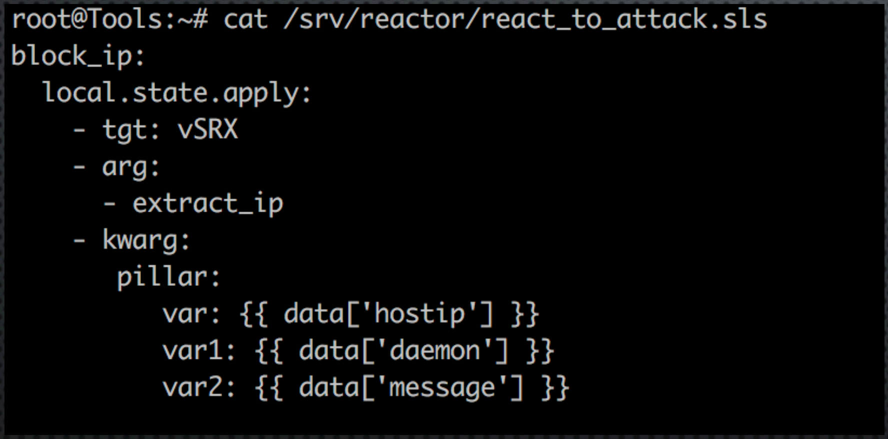


```
root@Tools:~# cat /etc/salt/master.d/reactor.conf
# Example reactor configurations below
reactor:
   - 'salt/engines/hook/gitlab':
     - /srv/reactor/gitlab_example.sls
     - /srv/reactor/proxy_inventory.sls
   - 'jnpr/syslog/Blue1_SRX/*':
     - /srv/reactor/react_to_attack.sls
```

The expression is matching the events published by vSRX on message bus to a .sls file. It is a regular expression used for matching and whenever an event is published starting with “jnpr/syslog/Blue_1SRX/” corresponding .sls file is triggered.

Salt: Files, their purpose and details

2. /srv/reactor/react_to_attack.sls



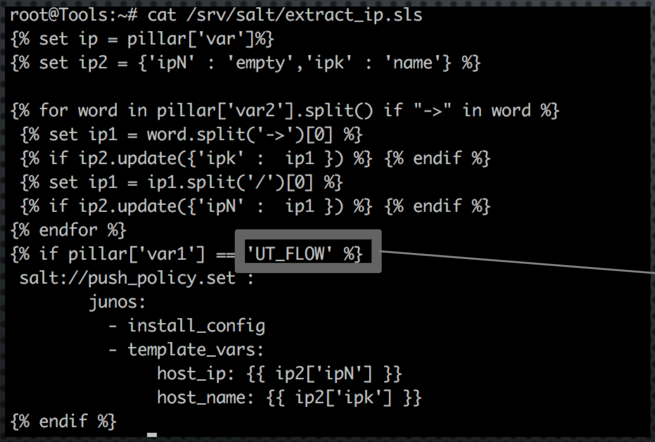
```
root@Tools:~# cat /srv/reactor/react_to_attack.sls
block_ip:
  local.state.apply:
    - tgt: vSRX
    - arg:
      - extract_ip
    - kwarg:
       pillar:
          var: {{ data['hostip'] }}
          var1: {{ data['daemon'] }}
          var2: {{ data['message'] }}
```

hostip, daemon, and message are the attributes that belong to the message log corresponding to the event published on the message bus. Here we are assigning them to separate variables so that this data can be utilized in next scripts to react to event and push policies.

The event log is received in JSON format on the salt master bus. You can grab any data from it in this file using the “key” and later use that in your scripts.

Salt: Files, their purpose and details

3. /srv/salt/extract_ip.sls :

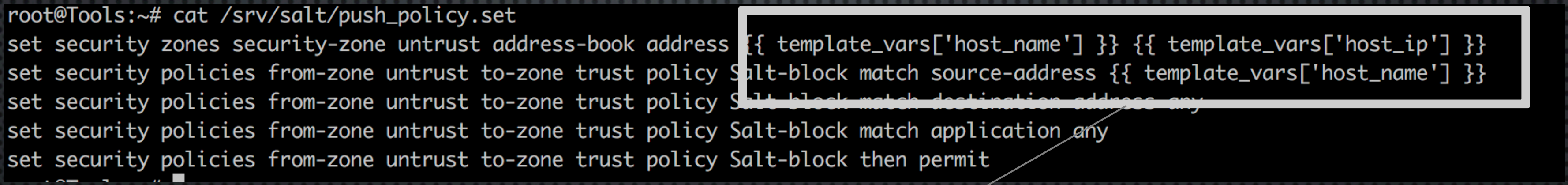


```
root@Tools:~# cat /srv/salt/extract_ip.sls
{% set ip = pillar['var']%}
{% set ip2 = {'ipN' : 'empty','ipk' : 'name'} %}

{% for word in pillar['var2'].split() if "->" in word %}
 {% set ip1 = word.split('->')[0] %}
 {% if ip2.update({'ipk' :  ip1 }) %} {% endif %}
 {% set ip1 = ip1.split('/')[0] %}
 {% if ip2.update({'ipN' :  ip1 }) %} {% endif %}
{% endfor %}
{% if pillar['var1'] == 'UT_FLOW' %}
 salt://push_policy.set :
        junos:
          - install_config
          - template_vars:
              host_ip: {{ ip2['ipN'] }}
              host_name: {{ ip2['ipk'] }}
{% endif %}
```

As we don't want to push policy corresponding to all the events that we receive, this condition 'UT_FLOW' makes this reactor to push policy corresponding to UTM events only. Participants can put several other match criteria here matching to their event log, to push policy in response to specific events.

Salt: Files, their purpose and details

4. /srv/salt/push_policy.set



```
root@Tools:~# cat /srv/salt/push_policy.set
set security zones security-zone untrust address-book address {{ template_vars['host_name'] }} {{ template_vars['host_ip'] }}
set security policies from-zone untrust to-zone trust policy Salt-block match source-address {{ template_vars['host_name'] }}
set security policies from-zone untrust to-zone trust policy Salt-block match destination-address any
set security policies from-zone untrust to-zone trust policy Salt-block match application any
set security policies from-zone untrust to-zone trust policy Salt-block then permit
```

These are not the part of Junos syntax . These are template variables that we getting from the extract_ip.sls and using to push a policy corresponding to the IP address filtered from the log.

Other Useful resources to learn about Salt

JUNOS SALT MODULE:

HTTPS://DOCS.SALTSTACK.COM/EN/LATEST/REF/MODULES/ALL/SALT.MODULES.JUNOS.HTML

GITHUB EXAMPLE :

HTTPS://GITHUB.COM/KSATOR/JUNOS-AUTOMATION-WITH-SALTSTACK/WIKI/17.-JUNOS_SYSLOG-ENGINE-AND-SALT'S-REACTOR-SYSTEM-END-TO-END-DEMO

Rules of engagement

YOU MUST NOT BLOCK LEGITIMATE CUSTOMER TRAFFIC.

YOU MUST ENSURE THAT THE FOLLOWING SERVICES ARE REACHABLE FROM THE INTERNET.
- PING, HTTP (80 & 8080), HTTPS, FTP

YOU CAN USE IPS BUT DUE TO CUSTOMER SLA PENALTIES AND PREVIOUS ISSUES, YOU CANNOT USE IPS TO BLOCK INLINE.

Thank you!

Engineering
Simplicity