Leveraging the Capability of Extreme Networks XMC's Governance Engine

Enabling complex tests via the IGE 'TEST FUNCTION' feature

Presented by Kevin DeTerra

kdeterra@extrementworks.com

Extreme®

Connect Beyond the Network

Leveraging XMC’s Governance Engine

- Customers can create custom IGE test regime using the GUI
  - Leverage regular expression (regex) for pattern matching in configuration items
  - declaring pass /fail test criteria
  - Setting test importance / ‘weight’ for scoring
  - Assumption is that these custom tests reside only on the customer instance of XMC
    - No GUI method for exporting these modifications
- **Let’s explore how XMC’s IGE is capable of even more!**
  - Perform rigorous tests leveraging python
    - examples: Inspecting ACLs, routing peer authentication, AAA configurations, etc.
  - More descriptive results
    - ‘test details’ reporting directs operators to root cause
  - Exported customized test regimes
    - Develop on one XMC instance and shared with many
    - Disclaimer: Not recommended to develop on a customer’s XMC

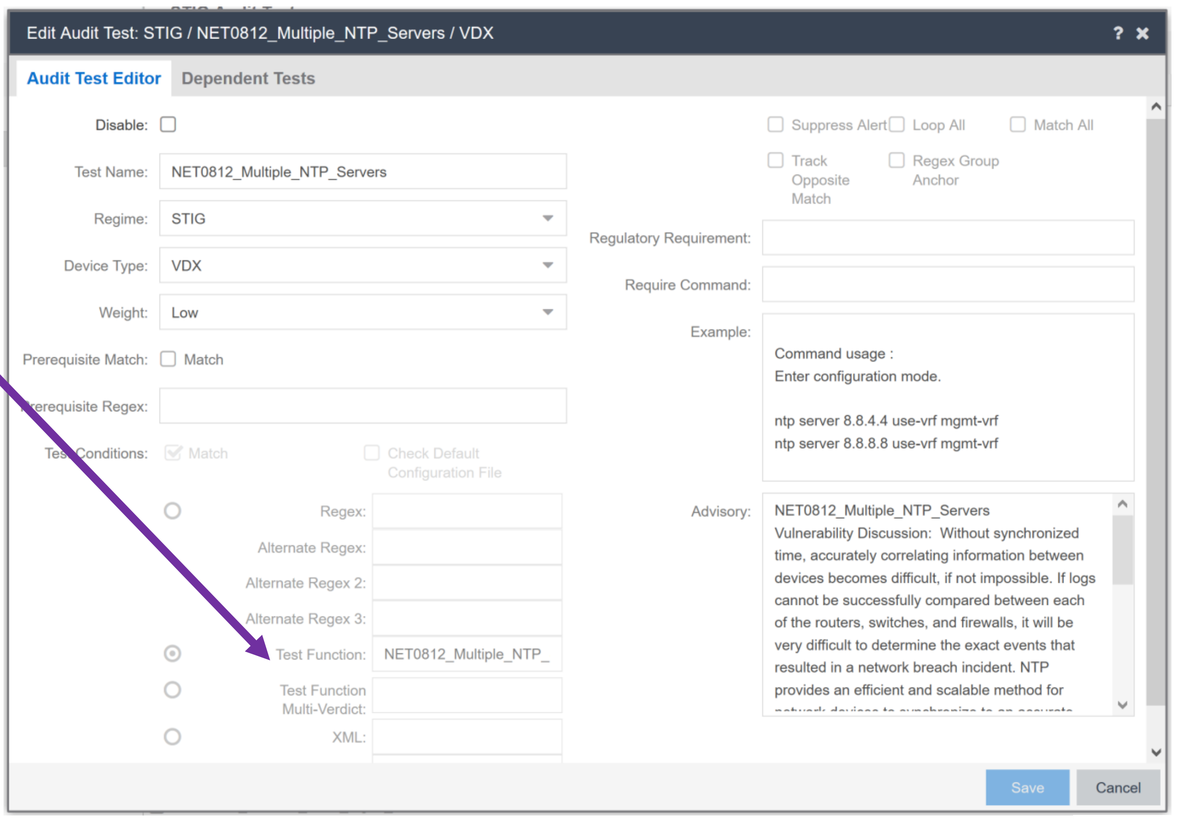
Discussion Topics

- Preparing a new test regime and python script
- The DEVICE.conf file and variable relationship to the GUI
- Enable the governance engine to use your code
  - Importing the new python code as a library
  - Loading the library’s function_map
- Python code used in your tests
  - Define python functions and a function_map
- The required files and their how they relate to the OneView
- The required meta data about the tests you add
- Exporting a test regime from XMC
- Troubleshooting Hints
- Importing a test regime to XMC
- How to activate our new regime / tests

Leveraging XMC’s Governance Engine

Purpose of this presentation is to show how to implement exportable test regime complete with python driven tests

- The GUI allows us to execute python code
  - The ‘Test Function’ enables the execution of a python function
  - But there are a few things that need to be setup to leverage this capability
- The following slides will describe this process



Edit Audit Test: STIG / NET0812_Multiple_NTP_Servers / VDX

Disable:	☐
Test Name:	NET0812_Multiple_NTP_Servers
Regime:	STIG
Device Type:	VDX
Weight:	Low
Prerequisite Match:	☐ Match
Prerequisite Regex:	
Test Conditions:	✓ Match ☐ Check Default Configuration File
○ Regex:	
Alternate Regex:	
Alternate Regex 2:	
Alternate Regex 3:	
● Test Function:	NET0812_Multiple_NTP_
○ Test Function Multi-Verdict:	
○ XML:	

☐ Suppress Alert ☐ Loop All ☐ Match All ☐ Track Opposite Match ☐ Regex Group Anchor

Regulatory Requirement:

Require Command:

Example:

```
Command usage :
Enter configuration mode.

ntp server 8.8.4.4 use-vrf mgmt-vrf
ntp server 8.8.8.8 use-vrf mgmt-vrf
```

Advisory: NET0812_Multiple_NTP_Servers
Vulnerability Discussion: Without synchronized time, accurately correlating information between devices becomes difficult, if not impossible. If logs cannot be successfully compared between each of the routers, switches, and firewalls, it will be very difficult to determine the exact events that resulted in a network breach incident. NTP provides an efficient and scalable method for

High Level Tasks

- General workflow
  - Create a new test regime folder, a new folder for each test complete with device files (DEVICE.conf)
  - Create a python file for our new test regime
  - Modify the governnce-engine.py script to import our new test regime as a library and it's python functions as a function_map
  - Modify the db-reset script to load our new test regime into XMC
  - Add each test one by one
    - Add a DEVICE.conf --to define the test to XMC
    - Add a python function --to implement the test
    - Update the function_map with the new function
  - Test that the new code compiles and produces the desired results
  - Repeat
- Save work / export
- Import

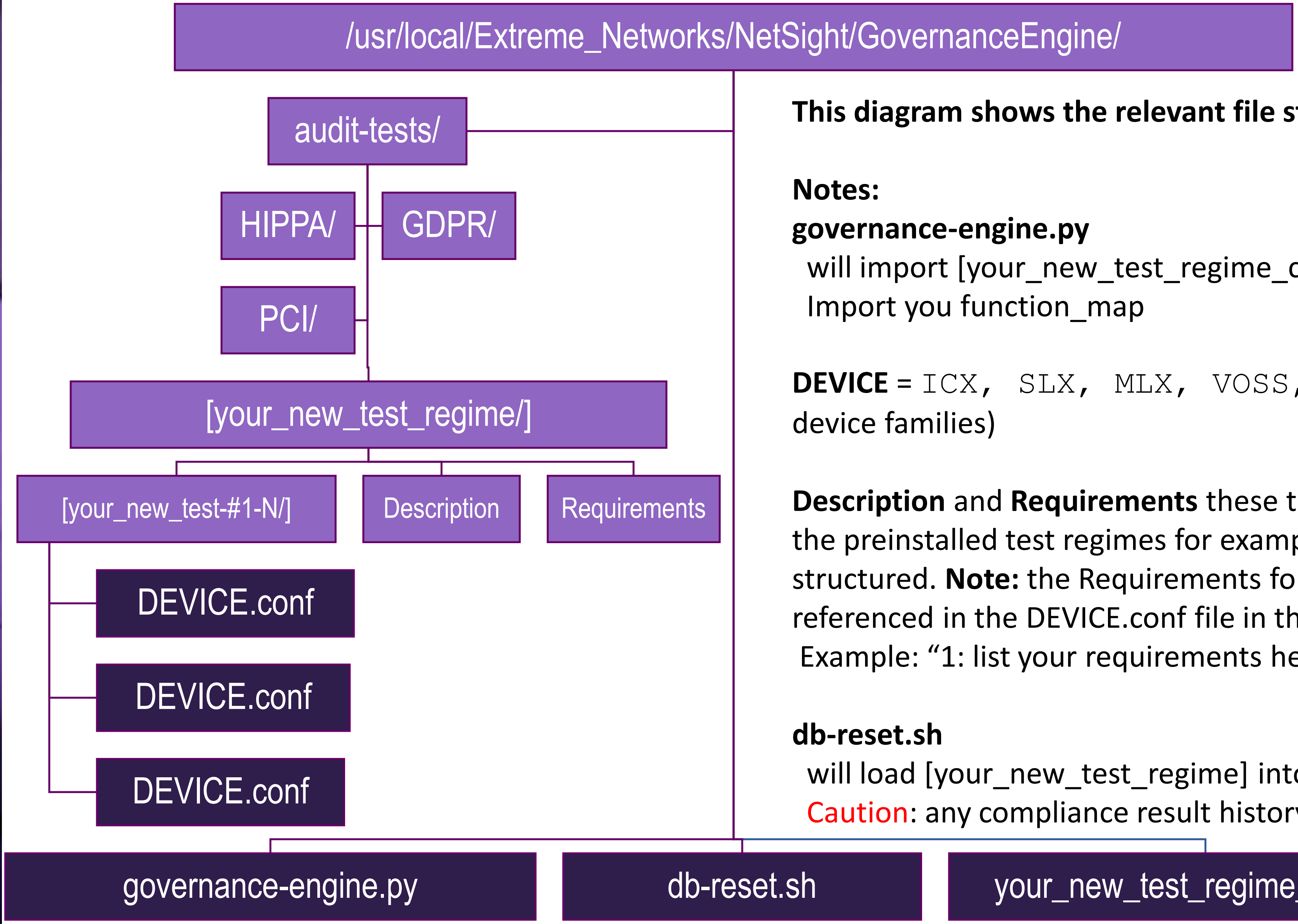
Preparing a New Test Regime and Python Script

- Login as root user and change directory to the governance engine's test directory
  - cd /usr/local/Extreme_Networks/NetSight/GovernanceEngine/audit-tests/
- Create a file for `[your_new_python_code.py]`
  - `touch your_new_python_code.py` (for example `sigaudit.py`)
- For this example we will use the following two variables
  - [your_new_test_regime] = "STIG"
  - [your_new_test] = a unique name (such as NETXXXX_*) for each test.
- Create a folder for your new test regime
  - mkdir [your_new_test_regime]
- Add the 'Description' and 'Requirements' text files
  - echo "add your description here" > [your_new_test_regime]/Description
  - Echo "add your test requirements here" > [your_new_test_regime]/Requirements

Preparing a New Test Regime and Python Script

- Within that test regime, create a folder for each new test within your regime
  - cd [your_new_test_regime]
  - mkdir [your_new_test]
- Add a DEVICE.conf file to the [your_new_test] folder
  - Each test requires at least one DEVICE.conf file in this folder
  - The name device equates to the device family
    - For a VDX switch it would be '`VDX.conf`'
    - `ICX, SLX, MLX, VOSS, EXOS` (and others) are some of the valid '`.conf`' filenames.
    - The IGE will choose the matching .conf file for the device family is it testing
- Hint: Browse the other test regimes for *.conf examples to get started with a copy and paste to minimize errors

Relevant Files and Folders



This diagram shows the relevant file structure.

Notes:

governance-engine.py
 will import [your_new_test_regime_code.py]
 Import you function_map

DEVICE = `ICX, SLX, MLX, VOSS, EXOS,` (and other device families)

Description and **Requirements** these text files are mandatory. See the preinstalled test regimes for examples on how they are structured. **Note:** the Requirements format has an index that is referenced in the DEVICE.conf file in the ‘AUDIT_CRITERIA’ filed.
 Example: “1: list your requirements here”

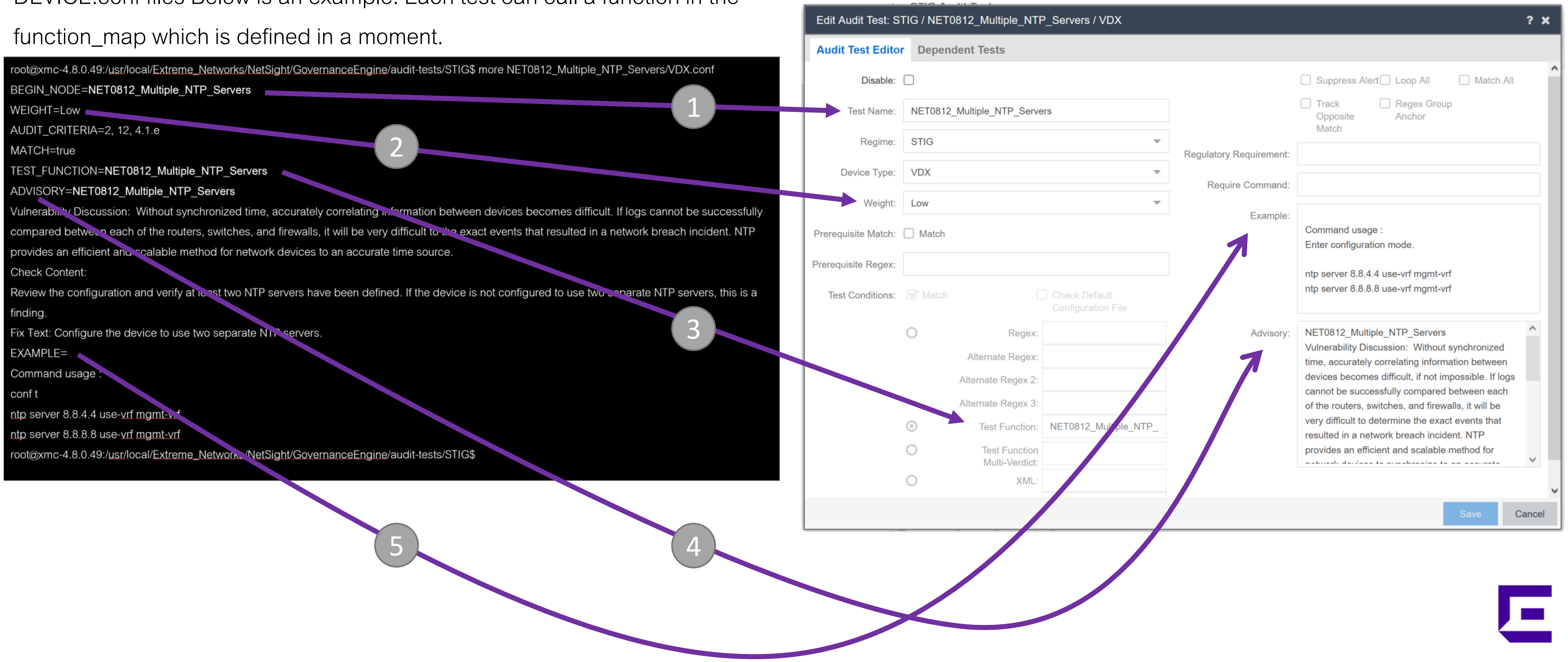
db-reset.sh
 will load [your_new_test_regime] into XMC’s DB
 Caution: any compliance result history will be erased!

Example of a ‘DEVICE.conf’ file

```
root@xmc-4.8.0.49:/usr/local/Extreme_Networks/NetSight/GovernanceEngine/audit-tests/STIG$ more NET0812_Multiple_NTP_Servers/VDX.conf
BEGIN_NODE=NET0812_Multiple_NTP_Servers
WEIGHT=Low
AUDIT_CRITERIA=2, 12, 4.1.e
MATCH=true
TEST_FUNCTION=NET0812_Multiple_NTP_Servers
ADVISORY=NET0812_Multiple_NTP_Servers
Vulnerability Discussion:  Without synchronized time, accurately correlating information between devices becomes difficult. If logs cannot be successfully compared between each of the routers, switches, and firewalls, it will be very difficult to the exact events that resulted in a network breach incident. NTP provides an efficient and scalable method for network devices to an accurate time source.
Check Content:
Review the configuration and verify at least two NTP servers have been defined. If the device is not configured to use two separate NTP servers, this is a finding.
Fix Text: Configure the device to use two separate NTP servers.
EXAMPLE=
Command usage :
conf t
ntp server 8.8.4.4 use-vrf mgmt-vrf
ntp server 8.8.8.8 use-vrf mgmt-vrf
root@xmc-4.8.0.49:/usr/local/Extreme_Networks/NetSight/GovernanceEngine/audit-tests/STIG$
```

The ‘DEVICE.conf’ Relationship to the Audit Test

Note: Each audit-test has one or more tests. Each test contains one or more DEVICE.conf files Below is an example. Each test can call a function in the function_map which is defined in a moment.



```
root@xmc-4.8.0.49:/usr/local/Extreme_Networks/NetSight/GovernanceEngine/audit-tests/STIG$ more NET0812_Multiple_NTP_Servers/VDX.conf
BEGIN_NODE=NET0812_Multiple_NTP_Servers
WEIGHT=Low
AUDIT_CRITERIA=2, 12, 4.1.e
MATCH=true
TEST_FUNCTION=NET0812_Multiple_NTP_Servers
ADVISORY=NET0812_Multiple_NTP_Servers
Vulnerability Discussion:  Without synchronized time, accurately correlating information between devices becomes difficult. If logs cannot be successfully
compared between each of the routers, switches, and firewalls, it will be very difficult to the exact events that resulted in a network breach incident. NTP
provides an efficient and scalable method for network devices to an accurate time source.
Check Content:
Review the configuration and verify at least two NTP servers have been defined. If the device is not configured to use two separate NTP servers, this is a
finding.
Fix Text: Configure the device to use two separate NTP servers.
EXAMPLE=
Command usage :
conf t
ntp server 8.8.4.4 use-vrf mgmt-vrf
ntp server 8.8.8.8 use-vrf mgmt-vrf
root@xmc-4.8.0.49:/usr/local/Extreme_Networks/NetSight/GovernanceEngine/audit-tests/STIG$
```

Edit Audit Test: STIG / NET0812_Multiple_NTP_Servers / VDX

Audit Test Editor | Dependent Tests

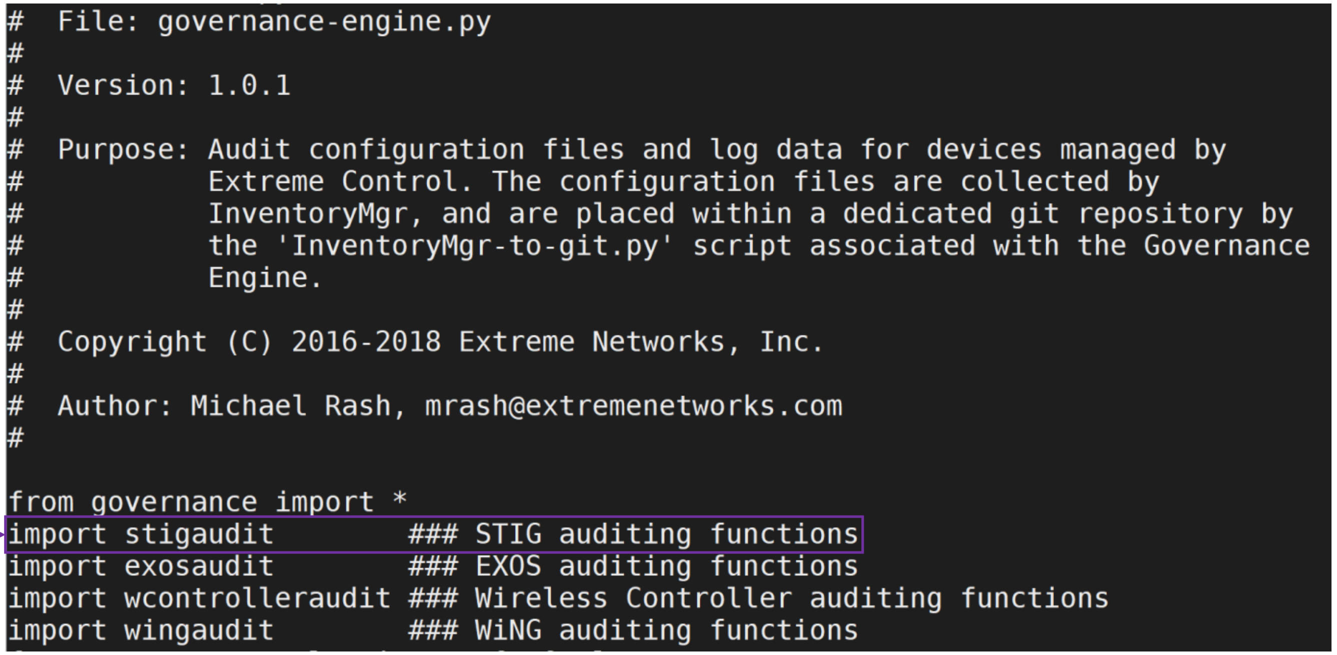
Disable:	☐
Test Name:	NET0812_Multiple_NTP_Servers
Regime:	STIG
Device Type:	VDX
Weight:	Low
Prerequisite Match:	☐ Match
Prerequisite Regex:	
Test Conditions:	✓ Match; ☐ Check Default Configuration File
○ Regex:	
Alternate Regex:	
Alternate Regex 2:	
Alternate Regex 3:	
● Test Function:	NET0812_Multiple_NTP_
○ Test Function Multi-Verdict:	
○ XML:	

☐ Suppress Alert ☐ Loop All ☐ Match All
☐ Track Opposite Match ☐ Regex Group Anchor

Regulatory Requirement:	
Require Command:	
Example:	Command usage : Enter configuration mode. ntp server 8.8.4.4 use-vrf mgmt-vrf ntp server 8.8.8.8 use-vrf mgmt-vrf
Advisory:	NET0812_Multiple_NTP_Servers Vulnerability Discussion: Without synchronized time, accurately correlating information between devices becomes difficult, if not impossible. If logs cannot be successfully compared between each of the routers, switches, and firewalls, it will be very difficult to determine the exact events that resulted in a network breach incident. NTP provides an efficient and scalable method for

Import Your New Python Test Code

- Modify the governance-engine.py script to import the new test regime
- The example on the right shows our new python code ‘`sitgaudit`’ test that will now be imported. You will name this file something else
  - Note the file we intent to import is named ‘stigaudit.py’
    - `[your_new_python_code.py]`
  - The import statement does not include the ‘`.py`’
- Our python code ‘`stigaudit.py`’ will be in the same directory as the `governance-engine.py` script
  - `/usr/local/Extreme_Networks/NetSight/GovernanceEngine/`



```
#  File: governance-engine.py
#
#  Version: 1.0.1
#
#  Purpose: Audit configuration files and log data for devices managed by
#           Extreme Control. The configuration files are collected by
#           InventoryMgr, and are placed within a dedicated git repository by
#           the 'InventoryMgr-to-git.py' script associated with the Governance
#           Engine.
#
#  Copyright (C) 2016-2018 Extreme Networks, Inc.
#
#  Author: Michael Rash, mrash@extremenetworks.com
#

from governance import *
import stigaudit        ### STIG auditing functions
import exosaudit        ### EXOS auditing functions
import wcontrolleraudit ### Wireless Controller auditing functions
import wingaudit        ### WiNG auditing functions
```

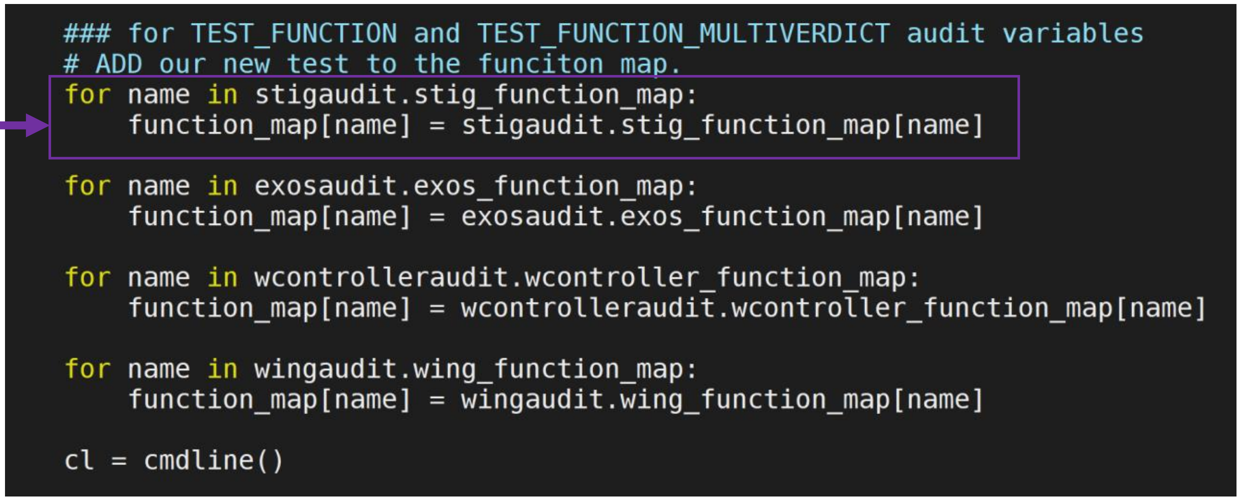
(part of) governance-engine.py

Modify the ‘governance-engine.py’ Script to Load Your New Python Code

- The function_map needs to be updated with the names of the python functions that we will write for our custom tests
- The Example on the right shows a new modification to the governance-engine.py file to include the sitgaudit.stig_fuction_map

Safety Net: Before any edits, it is always good practice to make a backup copy of any files you modify!

(part of) governance-engine.py

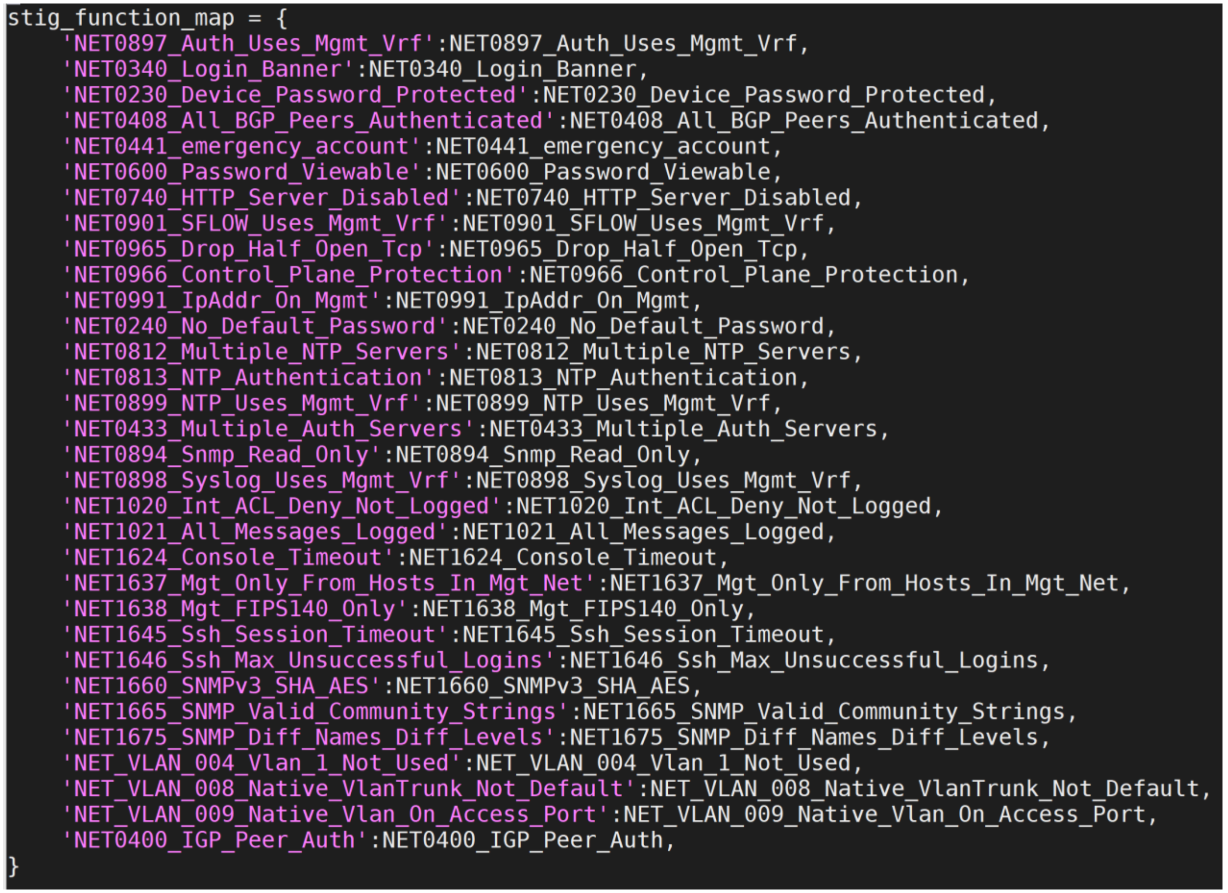


```
### for TEST_FUNCTION and TEST_FUNCTION_MULTIVERDICT audit variables
# ADD our new test to the funciton map.
for name in stigaudit.stig_function_map:
    function_map[name] = stigaudit.stig_function_map[name]

for name in exosaudit.exos_function_map:
    function_map[name] = exosaudit.exos_function_map[name]

for name in wcontrolleraudit.wcontroller_function_map:
    function_map[name] = wcontrolleraudit.wcontroller_function_map[name]

for name in wingaudit.wing_function_map:
    function_map[name] = wingaudit.wing_function_map[name]

cl = cmdline()
```

Test Regime Function Map

- The governance-engine.py will expect our python code to include a function map
  - We called this 'stig_function_map'
  - Simple key value pairs
- The function_map links the python code function to the name of the test as defined in the 'TEST_FUNCITON' field of the 'DEVICE.conf' files
  - For simplicity and traceability, use the same name for the test and the function (if you have a one to one mapping)

(part of) stigaudit.py



```
stig_function_map = {
    'NET0897_Auth_Uses_Mgmt_Vrf':NET0897_Auth_Uses_Mgmt_Vrf,
    'NET0340_Login_Banner':NET0340_Login_Banner,
    'NET0230_Device_Password_Protected':NET0230_Device_Password_Protected,
    'NET0408_All_BGP_Peers_Authenticated':NET0408_All_BGP_Peers_Authenticated,
    'NET0441_emergency_account':NET0441_emergency_account,
    'NET0600_Password_Viewable':NET0600_Password_Viewable,
    'NET0740_HTTP_Server_Disabled':NET0740_HTTP_Server_Disabled,
    'NET0901_SFLOW_Uses_Mgmt_Vrf':NET0901_SFLOW_Uses_Mgmt_Vrf,
    'NET0965_Drop_Half_Open_Tcp':NET0965_Drop_Half_Open_Tcp,
    'NET0966_Control_Plane_Protection':NET0966_Control_Plane_Protection,
    'NET0991_IpAddr_On_Mgmt':NET0991_IpAddr_On_Mgmt,
    'NET0240_No_Default_Password':NET0240_No_Default_Password,
    'NET0812_Multiple_NTP_Servers':NET0812_Multiple_NTP_Servers,
    'NET0813_NTP_Authentication':NET0813_NTP_Authentication,
    'NET0899_NTP_Uses_Mgmt_Vrf':NET0899_NTP_Uses_Mgmt_Vrf,
    'NET0433_Multiple_Auth_Servers':NET0433_Multiple_Auth_Servers,
    'NET0894_Snmp_Read_Only':NET0894_Snmp_Read_Only,
    'NET0898_Syslog_Uses_Mgmt_Vrf':NET0898_Syslog_Uses_Mgmt_Vrf,
    'NET1020_Int_ACL_Deny_Not_Logged':NET1020_Int_ACL_Deny_Not_Logged,
    'NET1021_All_Messages_Logged':NET1021_All_Messages_Logged,
    'NET1624_Console_Timeout':NET1624_Console_Timeout,
    'NET1637_Mgt_Only_From_Hosts_In_Mgt_Net':NET1637_Mgt_Only_From_Hosts_In_Mgt_Net,
    'NET1638_Mgt_FIPS140_Only':NET1638_Mgt_FIPS140_Only,
    'NET1645_Ssh_Session_Timeout':NET1645_Ssh_Session_Timeout,
    'NET1646_Ssh_Max_Unsuccessful_Logins':NET1646_Ssh_Max_Unsuccessful_Logins,
    'NET1660_SNMPv3_SHA_AES':NET1660_SNMPv3_SHA_AES,
    'NET1665_SNMP_Valid_Community_Strings':NET1665_SNMP_Valid_Community_Strings,
    'NET1675_SNMP_Diff_Names_Diff_Levels':NET1675_SNMP_Diff_Names_Diff_Levels,
    'NET_VLAN_004_Vlan_1_Not_Used':NET_VLAN_004_Vlan_1_Not_Used,
    'NET_VLAN_008_Native_VlanTrunk_Not_Default':NET_VLAN_008_Native_VlanTrunk_Not_Default,
    'NET_VLAN_009_Native_Vlan_On_Access_Port':NET_VLAN_009_Native_Vlan_On_Access_Port,
    'NET0400_IGP_Peer_Auth':NET0400_IGP_Peer_Auth,
}
```

An Example of ‘Your New Python Code’

- Using your favorite editor, create a new file for your python code
- Use import to import any required libraries (like the regex library)
  - Import re
- Define any required global variables, in this case I’ve predefined the following Booleans:
  - PASSEDTEST=False
  - FAILEDTEST=True
- Getting Started with python functions
  - The device configuration is passed to our code using the variable (a list) called ‘`lines`’
  - Our code will loop over each ‘`line`’ in the list of ‘`lines`’
  - We can use a regular expression ‘re’ to find the line we want to look for.
    - If `re.search` returns a value to ‘`m`’ then that line matched.
  - In the return statement, after the pass/fail indicator, we can give the operator some data that will show in the XMC GUI and reports.

Hints:

Use an on-line regex evaluator https://regex101.com/

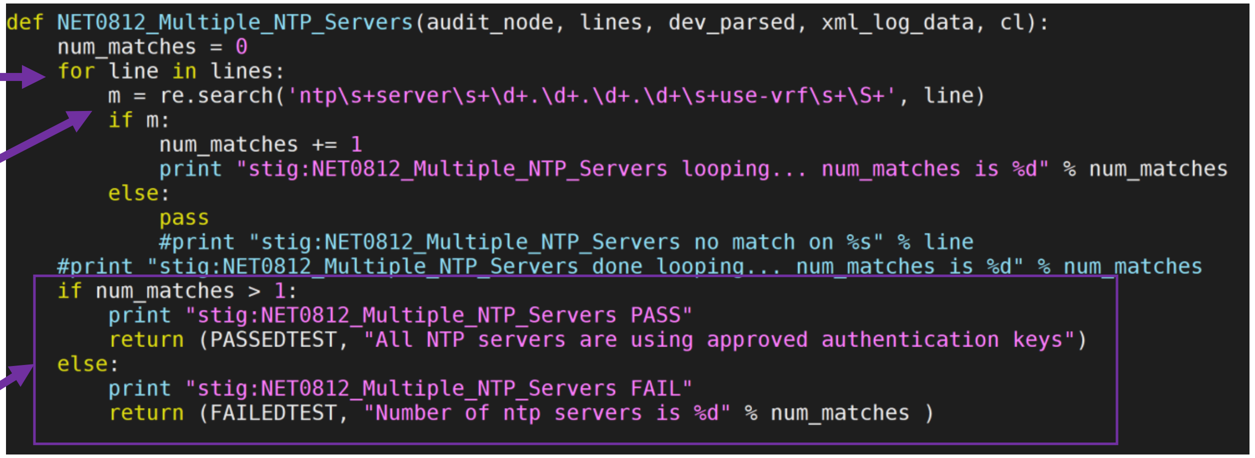
Learn about regex match-groups. Ex: (\S+)

Try to design regex that are easy to read and forgiving

- use \s+ instead of whitespaces

Print statements are directed to:

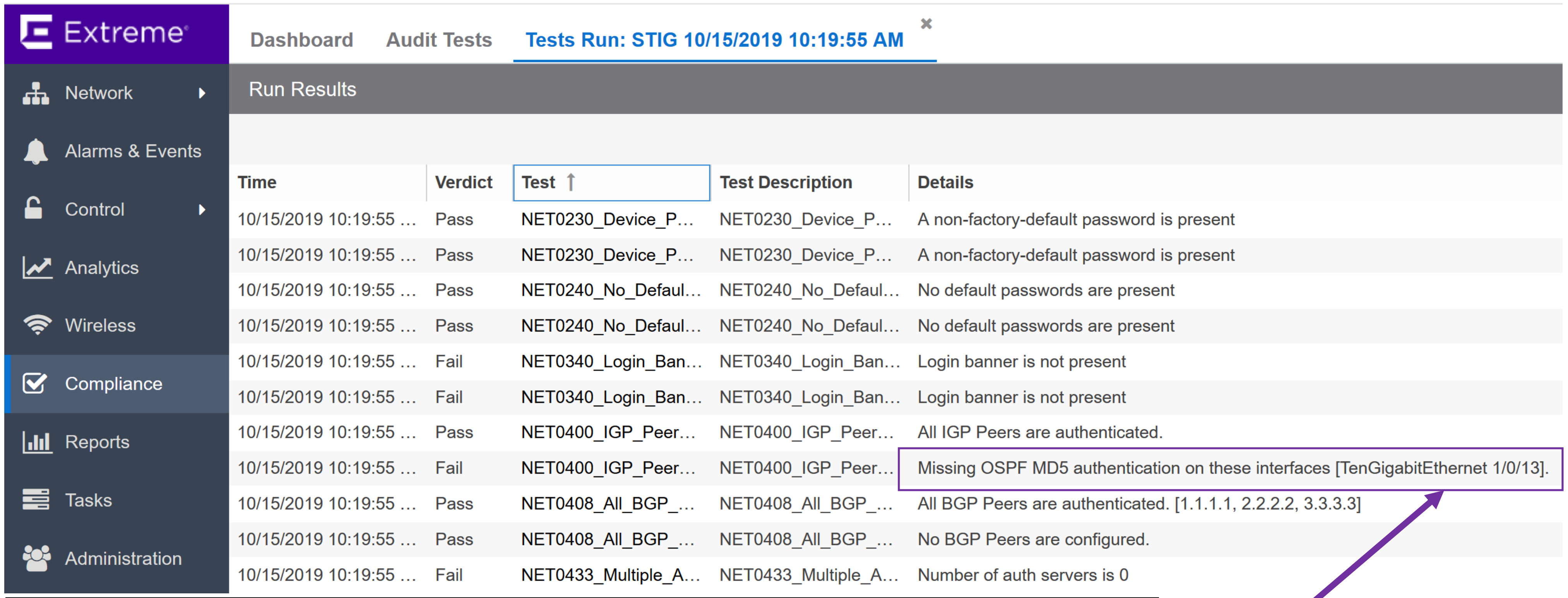
`/usr/local/Extreme_Networks/NetSight/appdata/logs/gov_eng.log`



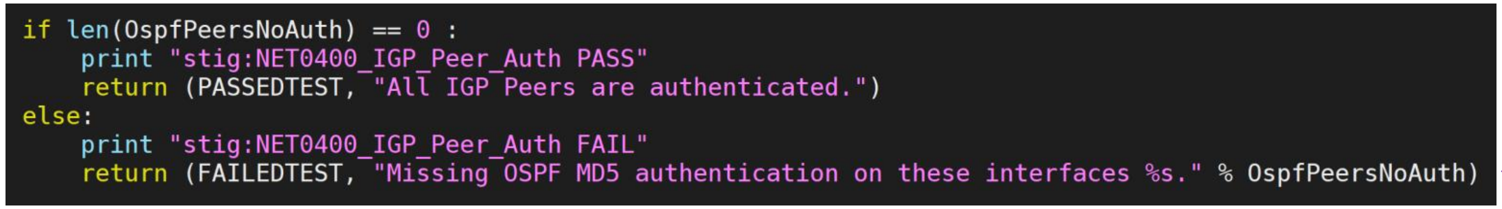
```
def NET0812_Multiple_NTP_Servers(audit_node, lines, dev_parsed, xml_log_data, cl):
    num_matches = 0
    for line in lines:
        m = re.search('ntp\s+server\s+\d+.\d+.\d+.\d+\s+use-vrf\s+\S+', line)
        if m:
            num_matches += 1
            print "stig:NET0812_Multiple_NTP_Servers looping... num_matches is %d" % num_matches
        else:
            pass
            #print "stig:NET0812_Multiple_NTP_Servers no match on %s" % line
    #print "stig:NET0812_Multiple_NTP_Servers done looping... num_matches is %d" % num_matches
    if num_matches > 1:
        print "stig:NET0812_Multiple_NTP_Servers PASS"
        return (PASSEDTEST, "All NTP servers are using approved authentication keys")
    else:
        print "stig:NET0812_Multiple_NTP_Servers FAIL"
        return (FAILEDTEST, "Number of ntp servers is %d" % num_matches )
```

(part of) ‘stigaudit.py’

Python Returning Test “Details” Identifies Root Cause



Time	Verdict	Test	Test Description	Details
10/15/2019 10:19:55 …	Pass	NET0230_Device_P…	NET0230_Device_P…	A non-factory-default password is present
10/15/2019 10:19:55 …	Pass	NET0230_Device_P…	NET0230_Device_P…	A non-factory-default password is present
10/15/2019 10:19:55 …	Pass	NET0240_No_Defaul…	NET0240_No_Defaul…	No default passwords are present
10/15/2019 10:19:55 …	Pass	NET0240_No_Defaul…	NET0240_No_Defaul…	No default passwords are present
10/15/2019 10:19:55 …	Fail	NET0340_Login_Ban…	NET0340_Login_Ban…	Login banner is not present
10/15/2019 10:19:55 …	Fail	NET0340_Login_Ban…	NET0340_Login_Ban…	Login banner is not present
10/15/2019 10:19:55 …	Pass	NET0400_IGP_Peer…	NET0400_IGP_Peer…	All IGP Peers are authenticated.
10/15/2019 10:19:55 …	Fail	NET0400_IGP_Peer…	NET0400_IGP_Peer…	Missing OSPF MD5 authentication on these interfaces [TenGigabitEthernet 1/0/13].
10/15/2019 10:19:55 …	Pass	NET0408_All_BGP_…	NET0408_All_BGP_…	All BGP Peers are authenticated. [1.1.1.1, 2.2.2.2, 3.3.3.3]
10/15/2019 10:19:55 …	Pass	NET0408_All_BGP_…	NET0408_All_BGP_…	No BGP Peers are configured.
10/15/2019 10:19:55 …	Fail	NET0433_Multiple_A…	NET0433_Multiple_A…	Number of auth servers is 0



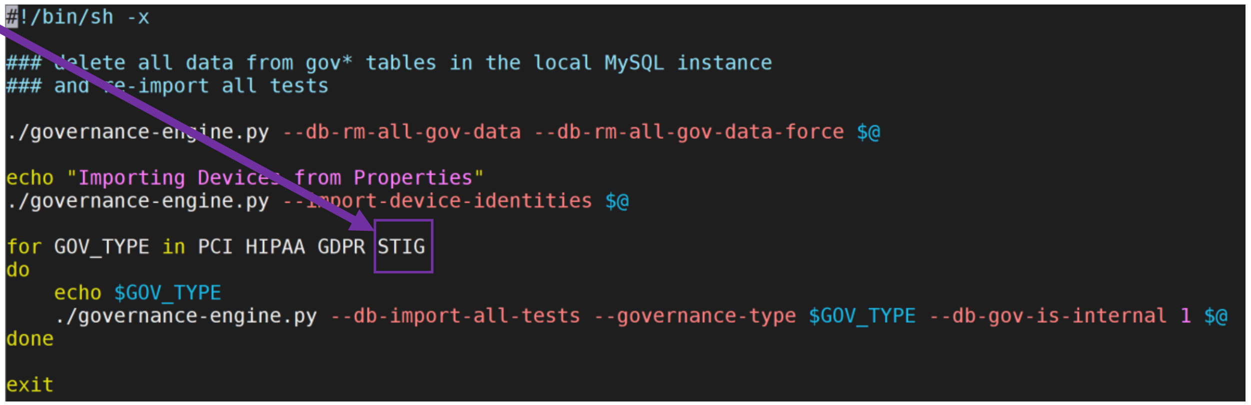
```
if len(OspfPeersNoAuth) == 0 :
    print "stig:NET0400_IGP_Peer_Auth PASS"
    return (PASSEDTEST, "All IGP Peers are authenticated.")
else:
    print "stig:NET0400_IGP_Peer_Auth FAIL"
    return (FAILEDTEST, "Missing OSPF MD5 authentication on these interfaces %s." % OspfPeersNoAuth)
```

Note: Opportunity to include pass/fail details to the operator from the python code to indicate the root cause

Load your new test regime in to XMC

- Modify the db-reset.sh script to load the new test regime folder (STIG) to XMC's database
- Run the script
  - sh db-reset.sh
- **Note: running the `db-reset.sh` script will erase any existing regime test result history from the XMC database**
- After the script completes without error, log into the XMC OneView web UI and verify that the new regime (in this example "STIG") is listed

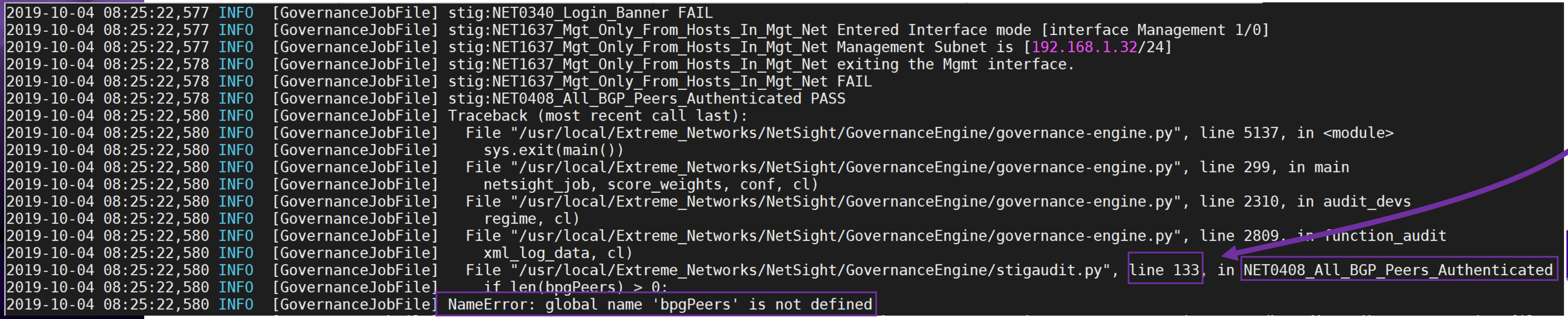
db-reset.sh



```
#!/bin/sh -x

### delete all data from gov* tables in the local MySQL instance
### and re-import all tests

./governance-engine.py --db-rm-all-gov-data --db-rm-all-gov-data-force $@

echo "Importing Devices from Properties"
./governance-engine.py --import-device-identities $@

for GOV_TYPE in PCI HIPAA GDPR STIG
do
    echo $GOV_TYPE
    ./governance-engine.py --db-import-all-tests --governance-type $GOV_TYPE --db-gov-is-internal 1 $@
done

exit
```

Troubleshooting Hints

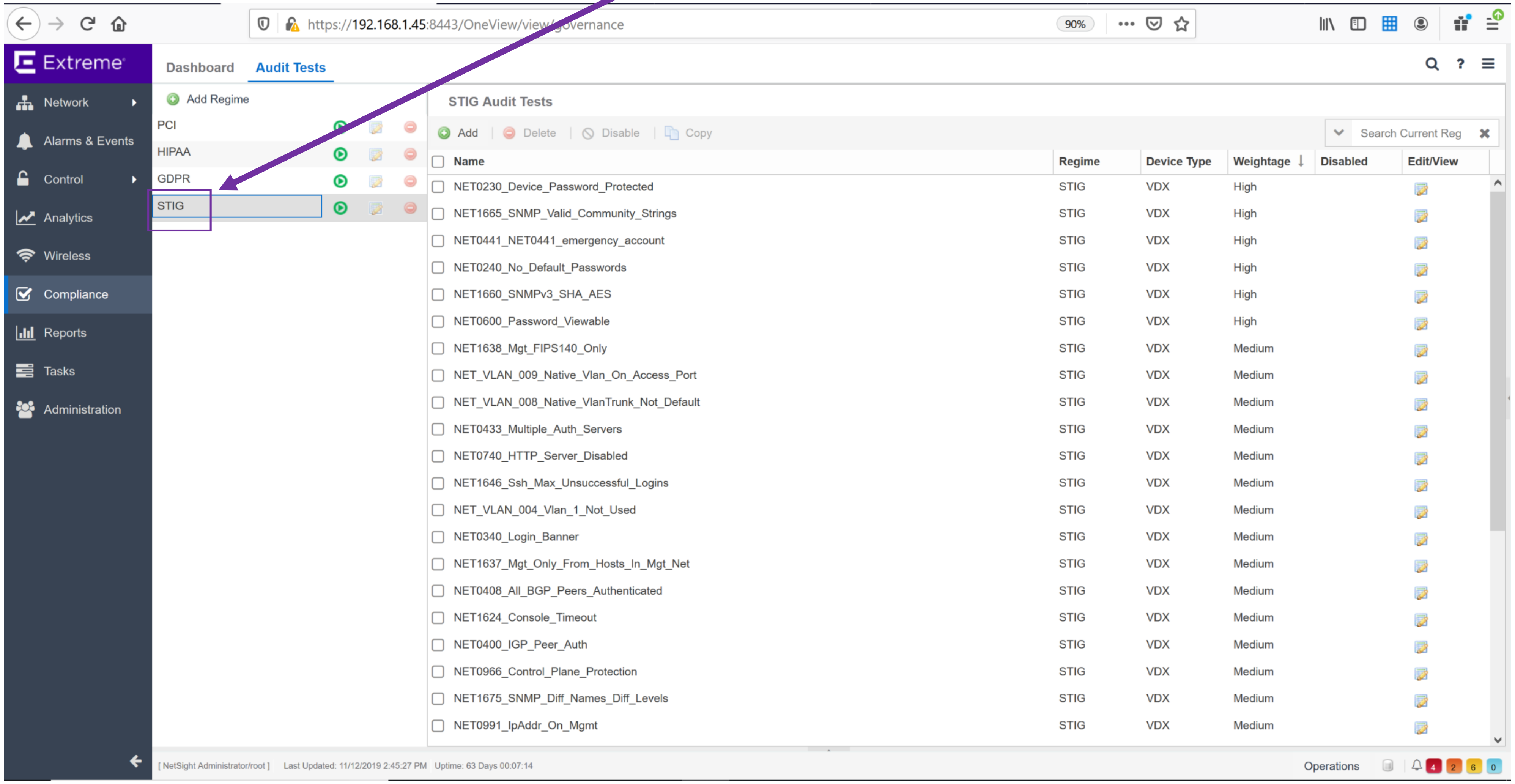
- Python code runs within the XMC IGE environment
- It will be compiled at runtime
- Functions defined in DEVICE.conf are validated when db-reset.sh is run
- Any compile errors will be directed to the gov_eng.log file
  - `/usr/local/Extreme_Networks/NetSight/appdata/logs/gov_eng.log`
  - The python file and offending function and line number will be shown along with a reason for failure
  - Print statements in your code also go to the log
- An indication of compile errors, but not the errors themselves will also be shown in the XMC OneView UI
- Use tools like more, less, tail or a text editor to inspect `gov_eng.log`



```
2019-10-04 08:25:22,577 INFO  [GovernanceJobFile] stig:NET0340_Login_Banner FAIL
2019-10-04 08:25:22,577 INFO  [GovernanceJobFile] stig:NET1637_Mgt_Only_From_Hosts_In_Mgt_Net Entered Interface mode [interface Management 1/0]
2019-10-04 08:25:22,577 INFO  [GovernanceJobFile] stig:NET1637_Mgt_Only_From_Hosts_In_Mgt_Net Management Subnet is [192.168.1.32/24]
2019-10-04 08:25:22,578 INFO  [GovernanceJobFile] stig:NET1637_Mgt_Only_From_Hosts_In_Mgt_Net exiting the Mgmt interface.
2019-10-04 08:25:22,578 INFO  [GovernanceJobFile] stig:NET1637_Mgt_Only_From_Hosts_In_Mgt_Net FAIL
2019-10-04 08:25:22,578 INFO  [GovernanceJobFile] stig:NET0408_All_BGP_Peers_Authenticated PASS
2019-10-04 08:25:22,580 INFO  [GovernanceJobFile] Traceback (most recent call last):
2019-10-04 08:25:22,580 INFO  [GovernanceJobFile]   File "/usr/local/Extreme_Networks/NetSight/GovernanceEngine/governance-engine.py", line 5137, in <module>
2019-10-04 08:25:22,580 INFO  [GovernanceJobFile]     sys.exit(main())
2019-10-04 08:25:22,580 INFO  [GovernanceJobFile]   File "/usr/local/Extreme_Networks/NetSight/GovernanceEngine/governance-engine.py", line 299, in main
2019-10-04 08:25:22,580 INFO  [GovernanceJobFile]     netsight_job, score_weights, conf, cl)
2019-10-04 08:25:22,580 INFO  [GovernanceJobFile]   File "/usr/local/Extreme_Networks/NetSight/GovernanceEngine/governance-engine.py", line 2310, in audit_devs
2019-10-04 08:25:22,580 INFO  [GovernanceJobFile]     regime, cl)
2019-10-04 08:25:22,580 INFO  [GovernanceJobFile]   File "/usr/local/Extreme_Networks/NetSight/GovernanceEngine/governance-engine.py", line 2809, in function_audit
2019-10-04 08:25:22,580 INFO  [GovernanceJobFile]     xml_log_data, cl)
2019-10-04 08:25:22,580 INFO  [GovernanceJobFile]   File "/usr/local/Extreme_Networks/NetSight/GovernanceEngine/stigaudit.py", line 133, in NET0408_All_BGP_Peers_Authenticated
2019-10-04 08:25:22,580 INFO  [GovernanceJobFile]     if len(bpgPeers) > 0:
2019-10-04 08:25:22,580 INFO  [GovernanceJobFile] NameError: global name 'bpgPeers' is not defined
```

Our New Test Regime

Our test regime for this example is called STIG



STIG Audit Tests

Name	Regime	Device Type	Weightage	Disabled	Edit/View
NET0230_Device_Password_Protected	STIG	VDX	High		✎
NET1665_SNMP_Valid_Community_Strings	STIG	VDX	High		✎
NET0441_NET0441_emergency_account	STIG	VDX	High		✎
NET0240_No_Default_Passwords	STIG	VDX	High		✎
NET1660_SNMPv3_SHA_AES	STIG	VDX	High		✎
NET0600_Password_Viewable	STIG	VDX	High		✎
NET1638_Mgt_FIPS140_Only	STIG	VDX	Medium		✎
NET_VLAN_009_Native_Vlan_On_Access_Port	STIG	VDX	Medium		✎
NET_VLAN_008_Native_VlanTrunk_Not_Default	STIG	VDX	Medium		✎
NET0433_Multiple_Auth_Servers	STIG	VDX	Medium		✎
NET0740_HTTP_Server_Disabled	STIG	VDX	Medium		✎
NET1646_Ssh_Max_Unsuccessful_Logins	STIG	VDX	Medium		✎
NET_VLAN_004_Vlan_1_Not_Used	STIG	VDX	Medium		✎
NET0340_Login_Banner	STIG	VDX	Medium		✎
NET1637_Mgt_Only_From_Hosts_In_Mgt_Net	STIG	VDX	Medium		✎
NET0408_All_BGP_Peers_Authenticated	STIG	VDX	Medium		✎
NET1624_Console_Timeout	STIG	VDX	Medium		✎
NET0400_IGP_Peer_Auth	STIG	VDX	Medium		✎
NET0966_Control_Plane_Protection	STIG	VDX	Medium		✎
NET1675_SNMP_Diff_Names_Diff_Levels	STIG	VDX	Medium		✎
NET0991_IpAddr_On_Mgmt	STIG	VDX	Medium		✎

[NetSight Administrator/root] Last Updated: 11/12/2019 2:45:27 PM Uptime: 63 Days 00:07:14 Operations 4 2 6 0

Exporting a Test Regime From XMC

- To export the new test regime
  - Because all of our work was done on the XMC filesystem, we need only tar or zip the files that were added or modified
  - Ensure that current path is as shown below
    - `cd /usr/local/Extreme_Networks/NetSight/GovernanceEngine/`
  - Create a tar archive of the relevant files
    - `Example: tar -jcvf STIG_9-25-2019.tar.bz2 audit-tests/STIG/ stigaudit.py governance-engine.py db-reset.sh`
  - Next copy the tar archive onto removable media, or copy to a workstation using FTP/SFTP/SCP/or other protocol

Importing a Test Regime to XMC

- To import on another XMC console
  - Change directory to `/usr/local/Extreme_Networks/NetSight/GovernanceEngine/`
  - Copy the tar archive
  - Backup the `governance-engine.py` and `db-reset.sh` files just in case you need to roll back the modifications…
  - Untar the archive (Note: this will replace the `governance-engine.py`, and `db-reset.sh` files)
    - `Example: tar –jxvf STIG_9-25-2019.tar.bz2`
  - Import the changes (Note: this will erase ALL existing customer IGE test results)
    - `sh db-reset.sh`
  - Mouse away from the OneView compliance tab then back to the compliance tab or refresh the page
  - Your new regime should now be present and usable

A bit about STIGs?

- Security Technical Implementation Guides (STIG) are published as a tool to improve the security of Department of Defense (DoD) information systems
- Defines what configuration items and features are allowable
- Conformance is required to select/purchase equipment and ultimately to operate equipment
  - Product not on the Approved Product List if it can pass a STIG
- STIGs are continually updated to enhance security posture
- Continuous compliance monitoring is becoming more common
  - The IGE/Compliance feature is well suited for this purpose

Part 2 - Install a New Regime in Real-Time

- This next section will show all the steps in a single video
- We will review the modifications to these files:
  - governance-engine.py
  - db-reset.sh
- The creation of the directory structure
  - audit-test/NEWREGIME/
  - audit-test/NEWREGIME/<TEST>
- The creation of these files:
  - audit-test/NEWREGIME/Requirements
    - Contents look like “1: add your requirements here”
    - DEVICE.conf file points back to “Requirements” file ex: “AUDIT_CRITERIA=1”
  - audit-test/NEWREGIME/Description
  - audit-test/NEWREGIME/<TEST>/DEVICE.conf
  - NEWREGIMEaudit.py

Summary

- We’ve described how to add a custom IGE test regime to XMC
  - Required modifications to the governace-engine.py file
  - Required modifications to the db-rest.sh file
  - The required directory structure for [your_new_regime]
  - The required files for [your_new_test]
- How to troubleshoot [your_new_python_code]
- The Effects of running the db-reset.sh script
  - **Erases** any existing compliance test results for all regimes
  - **It is not recommended to do this on a customer’s XMC**
- Exporting [your_new_test_regime]
- Importing [your_new_test_regime]
- Example files for this exercise are located on github
  - https://github.com/deterrak/xmc-add-pthton-test-regime.git

Extreme®

Connect Beyond the Network

WWW.EXTREMENETWORKS.COM