

Script to generate PA XML config for objects, policy rules and user-ID mappings

***** Script has been enhanced to support various type of provisions, including user-id mappings.**

Configuration parameters have been separated from the script so now we have 2 files - panapi.sh and pa.conf. Edit pa.conf to suit your need.

```
$ cat pa.conf
# -----
#
# Configuration
#
# Supported config includes the following.
#
# Device > Users
# Network > Interfaces > Ethernet
# Network > Interfaces > Loopback (with zone and vr assignment)
# Network > Interfaces > Tunnel (with zone and vr assignment)
# Network > Zones
# Network > IKE Gateways
# Network > IPSec Tunnels (with static routes through tunnels)
# Objects > Addresses
# Objects > Address Groups
# Objects > Services
# Objects > Custom URL Category with url.txt
# Policies > Security
# Policies > NAT
# Policies > PBF
# Network > VR > default > Static Routes
# Network > VR > default > BGP peer groups x peers
# User-ID mapping
#
# Features and Capacity, All PAN-OS Releases
# https://loop.paloaltonetworks.com/docs/DOC-3950
#
```

```
PA="10.10.123.7"
USER="admin"
PASS="admin"

TARGET="PA"
# TARGET="PANORAMA"

#
# The second PA to which config is mirrored. (e.g. IPSec)
#

#PA2="10.10.123.8"
#PA2=""
USER2="admin"
PASS2="admin"

# -----

PANORAMA_DEVICE_GROUP="POC-TEST"
PANORAMA_TEMPLATE="POC-TEST"
# PANORAMA_TEMPLATE_STACK="Finland"

# -----

VSYS="vsys1"

#DEFAULT_ZONE1="trust"
#DEFAULT_ZONE2="untrust"
DEFAULT_ZONE1="L3-Trust"
DEFAULT_ZONE2="L3-Untrust"
DEFAULT_ZONE3=""
```

Script to generate PA XML config for objects, policy rules and user-ID mappings

```
DEFAULT_VR="default"

N_USERS=1
N_NET_INT_ETHERNET=1000
N_NET_INT_LOOPBACK=1
N_NET_INT_TUNNEL=0
N_NET_ZONES=1
N_NET_IKE=0
N_NET_IPSEC=0

N_OBJS_ADDRESS=1
N_OBJS_ADDRESS_GROUP=1
N_OBJS_SERVICE=1

#
# Total custom URL entries = N_OBJS_URL_CATS x N_OBJS_URL_ENTRIES
#
N_OBJS_URL_CATS=5
N_OBJS_URL_ENTRIES=1000

#
# Policy
#
N_RULES_SEC=1
N_RULES_NAT=1
N_RULES_PBF=0

N_VR_STATIC=0

N_VR_BGP_PEER_GROUPS=1
N_VR_BGP_PEERS_PER_GROUP=1000

#
# Total IP-USER mappings = N_UID_NETS x N_UID_ENTRIES
#
N_UID_NETS=4
```

```
N_UID_ENTRIES=30

# -----
#
# Detailed settings
#

# N_USERS=0
USER_NAME="user%03d"
USER_PASS="paloAlto123" # create an authentication profile for testing


# N_NET_INT_ETHERNET=0
INT_ETHERNET_NAME="ethernet1/13"
INT_ETHERNET_TYPE="L3" # only L3 is supported
INT_ETHERNET_IP="192.%d.%d.1/24" # 192.i.j.1/24
INT_ETHERNET_IP_OCTET_i="168"
INT_ETHERNET_IP_OCTET_j="3"
INT_ETHERNET_VR="$DEFAULT_VR"
INT_ETHERNET_ZONE="$DEFAULT_ZONE1"


# N_NET_INT_LOOPBACK=0
#INT_LOOPBACK_IP=""
INT_LOOPBACK_IP="111.111.%d.%d" # 111.111.x.x/32
INT_LOOPBACK_VR="$DEFAULT_VR"
INT_LOOPBACK_ZONE="$DEFAULT_ZONE1"


# N_NET_INT_TUNNEL=0
INT_TUNNEL_IP=""
#INT_TUNNEL_IP="227.100.%d.%d/30" # 227.100.x.x/30
INT_TUNNEL_VR="$DEFAULT_VR"
INT_TUNNEL_ZONE="$DEFAULT_ZONE1"


# N_NET_ZONES=0
ZONE_NAME="Zone-%03d"


# N_NET_IKE=1000
```

Script to generate PA XML config for objects, policy rules and user-ID mappings

```
IKE_NAME="IKE_Gateway-%d"
IKE_VERSION="ikev2" # ikev1, ikev2 or ikev2-preferred
#IKE_INTERFACE="loopback.%d" # loopback or ethernet subinterfaces
#IKE_IP_LOCAL="127.%d.%d.1/32"
#IKE_IP_PEER="127.%d.%d.2"
IKE_INTERFACE="ethernet1/23" # loopback or ethernet subinterfaces
IKE_IP_LOCAL="100.1.0.1"
IKE_IP_LOCAL_PREFIX="/16"
IKE_IP_PEER="100.1.0.2"
IKE_IP_PEER_PREFIX="/16" # required for config mirror
IKE_PRESHARED_KEY="test123" # -AQ==cojt0Pw/L6ToM8G41aOKFIWh7w=CVJ5/F84i6cL7ejjM15fRA==
IKE_CRYPT_PROFILE="default"

#
# N_NET_IPSEC=12000
#
IPSEC_NAME="IPSec_Tunnel-%d" # IPSec_Tunnel-$i
#IPSEC_IKE_GATEWAY="IKE_Gateway-%d" # IKE_Gateway-$i
IPSEC_IKE_GATEWAY="IKE_Gateway-1"
IPSEC_CRYPT_PROFILE="default"
IPSEC_REPLAY_PROTECTION="no"
IPSEC_PROXY_ID_LIMIT=250
IPSEC_PROXY_ID_NAME="Proxy_ID-%d.%d" # Proxy_ID-$i.$j
IPSEC_PROXY_ID_LOCAL="172.16.%d.%d" # 172.16.$i.$j
IPSEC_PROXY_ID_REMOTE="192.168.%d.%d" # 192.168.$i.$j
IPSEC_PROXY_ID_PROTOCOL="any" # not used
IPSEC_ROUTE_ADD="no" # yes or no, whether routes should also be installed with the shell script
IPSEC_VR="$DEFAULT_VR"

# N_OBJS_ADDRESS=0
ADDR_NAME="Address-%03d"
# IP Netmask
ADDR_TYPE="ip-netmask"
ADDR_ADDRESS="10.%d.%d.0/24" # 10.$i.$j.0/24
#ADDR_ADDRESS="%0s2401:b200:2000:%x::/64" # 2401:b200:2000:$j::/64, $i is ignored and hidden
# IP Range
#ADDR_TYPE="ip-range"
#ADDR_RANGE="10"
#ADDR_ADDRESS="10.10.%d.%d" # 10.10.$i.$j-10.10.$i.$j(j+10), 0 < $j < 255
#ADDR_ADDRESS="2401:b200:2000::%x%02x" # 2401:b200:2000::$i$j-2401:b200:2000::$i$j(j+10), 0 < $j < 255
# FQDN
```

Script to generate PA XML config for objects, policy rules and user-ID mappings

```
#ADDR_TYPE="fqdn"
#ADDR_ADDRESS="w%d.panlab%d.local" # w$j.panlab$i.local, $j comes first for fqdn

#
# N_OBJS_ADDRESS_GROUP=0
#
ADDR_GROUP_NAME="Address_Group-%03d"
ADDR_GROUP_MEMBER_COUNT="5"
#
# Address objects, in case they are not set previously
#
N_OBJS_ADDRESS=1 # addresses will be recycled if this number is too small
ADDR_NAME="Address-%03d" # in case it is not set previously

#
# N_OBJS_SERVICE=0
#
SERVICE_NAME="service-%s%05d"
SERVICE_PROTOCOL="both" # "tcp" or "udp" or "both"
SERVICE_PORT_DST="10000" # initial dst port number
SERVICE_PORT_SRC="" # non-empty value will make source port grow with the destination port

#
# Custom URL categories
#
# N_OBJS_URL_CATS=4
# N_OBJS_URL_ENTRIES=1000
URL_CAT_NAME_i=1 # i the initial index
URL_CAT_NAME="URL_Category-%d"
URL_ENTRY_j=1 # j the initial index
URL_ENTRY="w%d.x%d.com"
URL_TYPE="URL List"

#
# N_RULES_SEC=0
#
SEC_RULEBASE="pre-rulebase" # pre-rulebase or post-rulebase for Panorama
#SEC_NAME="Rule-%5d"
SEC_NAME="Rule-%d"
```

Script to generate PA XML config for objects, policy rules and user-ID mappings

```
SEC_SRC_ZONE="$DEFAULT_ZONE1"
SEC_DST_ZONE="$DEFAULT_ZONE2"
#SEC_SOURCE="1.1.%d.0/24%0s" # 1.1.$i.0/24, $j is hidden
#SEC_DESTINATION="%0s2.2.%d.0/24" # 2.2.$j.0/24, $i is hidden
SEC_SOURCE="11.%d.%d.0/24" # 10.$i.$j.0/24
SEC_DESTINATION="22.%d.%d.0/24" # 20.$i.$j.0/24
#SEC_SERVICE="applicaton-default"
SEC_SERVICE="any"
SEC_ACTION="deny"

# N_RULES_NAT=0
NAT_RULEBASE="pre-rulebase" # pre-rulebase or post-rulebase for Panorama
#NAT_NAME="NAT_Rule-%5d"
NAT_NAME="NAT_Rule-%d"
NAT_SRC_ZONE="$DEFAULT_ZONE1"
NAT_DST_ZONE="$DEFAULT_ZONE2"
NAT_SOURCE="11.11.%d.0/24%0s" # 1.1.$i.0/24, $j is hidden
NAT_DESTINATION="%0s22.22.%d.0/24" # 2.2.$j.0/24, $i is hidden

# N_RULES_PBF=0
PBF_RULEBASE="pre-rulebase" # pre-rulebase or post-rulesbase for Panorama
PBF_NAME="PBF_Rule-%d"
PBF_SRC_ZONE="$DEFAULT_ZONE1"
PBF_SOURCE="10.%d.%d.0/24" # 10.$i.$j.0/24
PBF_INTERFACE="ethernet1/21"
PBF_NEXTHOP="100.1.0.100"

#
# Routes
#

# N_VR_STATIC=0
VR_STATIC_VR="$DEFAULT_VR"
VR_STATIC_NAME="Route-%d"
#VR_STATIC_DESTINATION="192.168.%d.%d" # 192.168.$i.$j
VR_STATIC_DESTINATION="%0s192.168.%d.0/24" # 192.168.$j.0/24, $i is hidden
#VR_STATIC_INTERFACE="ethernet1/23"
VR_STATIC_INTERFACE=""
VR_STATIC_NEXTHOP="100.1.0.2"
```

```
# N_VR_BGP_PEER_GROUPS=1
# N_VR_BGP_PEERS_PER_GROUP=1000
#
VR_BGP_VR="$DEFAULT_VR"
VR_BGP_PEER_GROUP_NAME="Group-%d" # Group-$i
VR_BGP_PEER_GROUP_TYPE="ebgp" # ebgp is supported at the moment
VR_BGP_PEER_NAME="Peer%d-%d" # Peer$i-$j
VR_BGP_PEER_AS="200"
#
VR_BGP_PEER_LOCAL_INT="ethernet1/13.%d" # ethernet1/13.$i
VR_BGP_PEER_LOCAL_IP="192.%d.%d.1/24" # 192.$j.$k.1/24
VR_BGP_PEER_LOCAL_IP_OCTET_j="168"
VR_BGP_PEER_LOCAL_IP_OCTET_k="3"
#
VR_BGP_PEER_PEER_IP="192.%d.%d.2" # 192.$j.$k.2

#
# IP-user mappings
#

# N_UID_NETS=4
# N_UID_ENTRIES=0
UID_DOMAIN_i=1 # i the initial index
UID_USER_j=1 # j the initial index
UID_USER="domain%d\\user%d" # domain$i\user$j
#
UID_IP_j=1 # first IP *.*.*j
#
# array of UID ranges of IP's
#
UID_IP[0]="1.1.0.%d"
UID_IP[1]="2.2.0.%d"
UID_IP[2]="3.3.0.%d"
UID_IP[3]="4.4.0.%d"
UID_IP[4]="5.5.0.%d"
```


Script to generate PA XML config for objects, policy rules and user-ID mappings

```
UID_IP[5]="6.6.0.%d"
UID_IP[6]="7.7.0.%d"
UID_IP[7]="8.8.0.%d"
UID_IP[8]="9.9.0.%d"
UID_IP[9]="10.10.0.%d"
#
UID_TIMEOUT="600" # 60 means 60 minutes

# -----
$
```

Description

The attached script generates dynamic files to inject XML config to PA or Panorama.

There will be multiple sets of files generated on each run. Each set consists of the following.

- A shell script *.sh (e.g. sec.sh for security rules) to load XML config with wget.
- An XML config file *.xml (e.g. sec.xml) to be posted by the script.
- A log file *.log after the script is executed.
- PA response (in XML) *.log.xml after the API call is done.
- Additional scripts and XML files *-clean.sh and *-clean.xml (e.g. addr-clean.sh), if any, for cleanup purpose.

The attached script requires 2 UNIX tools wget and xml_grep. wget is used to interact with PA, and xml_grep is for extracting data from PA XML response.

Please be sure these tools are available in your environment (Linux preferred).

Optionally, xml_pp is used to format the all-in-one XML file config.xml, which is imported by the script config.sh. We may use CLI to load the imported config with the merge option.

Depending on the size of the file config-*.xml, this step may take a while so you may want to skip it by commenting this line at the bottom of the script.

```
# xml_pp $CONFIG_FILE > $xml_file
```

Steps

1. Modify pa.conf to suit your need. For example, N_RULES_SEC is for the number of security rules. The hard limit for each type of rules or objects is 65,536.
2. Run the main script. The script will determine the API key and use it accordingly in the dynamic scripts.

```
$ panapi.sh -f pa.conf

param_file = pa.conf
PA = 10.10.123.7

Network > Interfaces > Ethernet (10) with zone and VR assigned
Network > Interfaces > Loopback (10) with zone and VR assigned
Network > Interfaces > Tunnel (10) with zone and VR assigned
Network > Zones (10)
Network > IKE Gateways (10)
Network > IPSec Tunnels (10) with static routes through tunnels
Objects > Addresses (1000) .....
Objects > Addresses Groups (10)
Objects > Services (1)
Objects > Custom URL Category with url.txt (5 x 1000) .....
Policies > Security (10)
Policies > NAT (10)
Policies > PBF (10)
Network > VR > default > Static Routes (10)
Network > VR > default > BGP peer groups x peers (1 x 1000) .....

User-ID IP-user mappings (4 x 30) .

Generating config.sh and xml/config.xml.. This takes a while.

Config scripts saved under the job directory 3038

$

$ cd 3038
$ ls -la
total 104
```

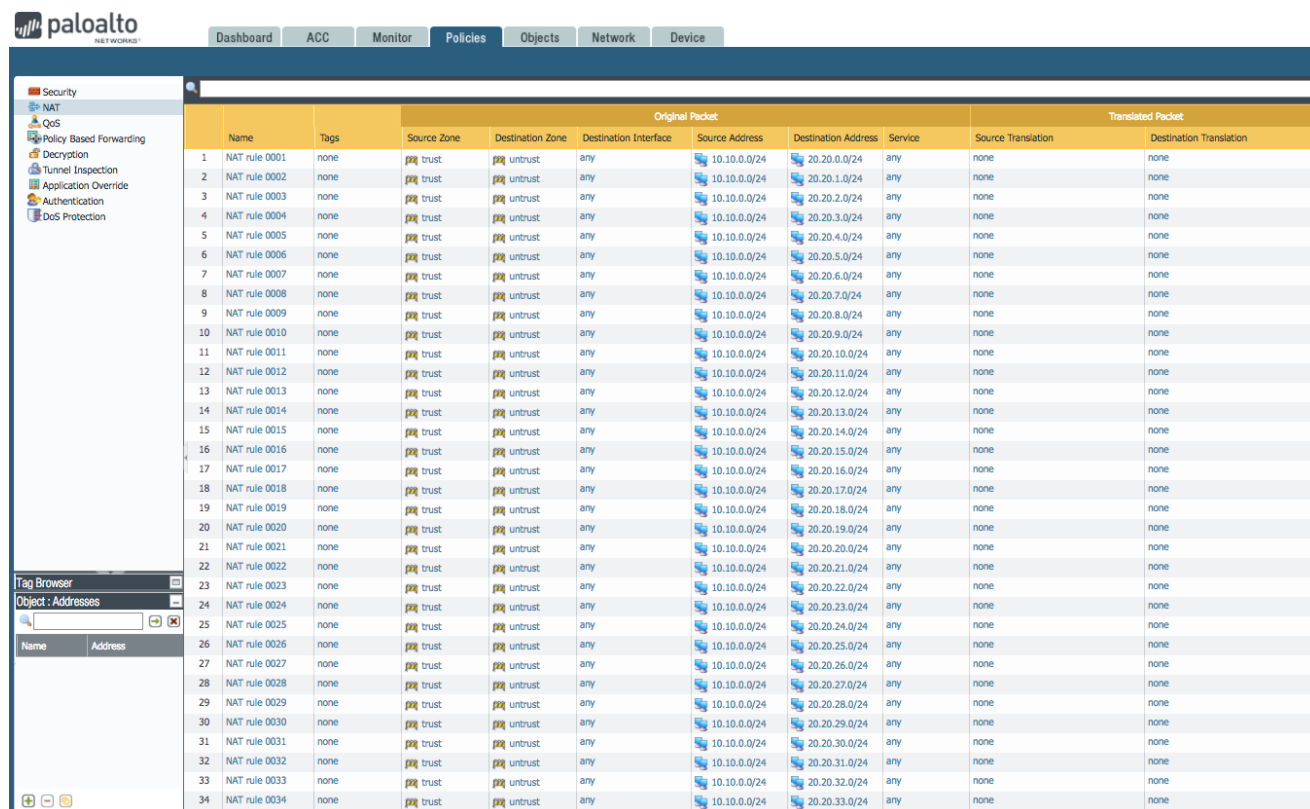
Script to generate PA XML config for objects, policy rules and user-ID mappings

```
drwxrwxr-x. 4 terence terence 4096 Jan  8 00:25 .
drwxrwxr-x. 12 terence terence 4096 Jan  8 00:25 ..
-rw-rw-r--. 1 terence terence 174 Jan  8 00:25 addr-clean.sh
-rw-rw-r--. 1 terence terence 192 Jan  8 00:25 addr_group-clean.sh
-rw-rw-r--. 1 terence terence 174 Jan  8 00:25 addr_group.sh
-rw-rw-r--. 1 terence terence 156 Jan  8 00:25 addr.sh
-rw-rw-r--. 1 terence terence 316 Jan  8 00:25 config.sh
-rw-rw-r--. 1 terence terence 582 Jan  8 00:25 ethernet-clean.sh
-rw-rw-r--. 1 terence terence 528 Jan  8 00:25 ethernet.sh
-rw-rw-r--. 1 terence terence 600 Jan  8 00:25 int-tunnel-clean.sh
-rw-rw-r--. 1 terence terence 546 Jan  8 00:25 int-tunnel.sh
drwxrwxr-x. 2 terence terence 40 Jan  8 00:25 logs
-rw-rw-r--. 1 terence terence 582 Jan  8 00:25 loopback-clean.sh
-rw-rw-r--. 1 terence terence 528 Jan  8 00:25 loopback.sh
-rw-rw-r--. 1 terence terence 171 Jan  8 00:25 nat-clean.sh
-rw-rw-r--. 1 terence terence 153 Jan  8 00:25 nat.sh
-rw-rw-r--. 1 terence terence 171 Jan  8 00:25 pbf-clean.sh
-rw-rw-r--. 1 terence terence 153 Jan  8 00:25 pbf.sh
-rw-rw-r--. 1 terence terence 159 Jan  8 00:25 route.sh
-rw-rw-r--. 1 terence terence 192 Jan  8 00:25 peer_group-clean.sh
-rw-rw-r--. 1 terence terence 333 Jan  8 00:25 peer_group.sh
-rw-rw-r--. 1 terence terence 171 Jan  8 00:25 sec-clean.sh
-rw-rw-r--. 1 terence terence 153 Jan  8 00:25 sec.sh
-rw-rw-r--. 1 terence terence 183 Jan  8 07:06 service-clean.sh
-rw-rw-r--. 1 terence terence 165 Jan  8 07:06 service.sh
-rw-rw-r--. 1 terence terence 153 Jan  8 00:25 uid.sh
-rw-rw-r--. 1 terence terence 171 Jan  8 00:25 url-clean.sh
-rw-rw-r--. 1 terence terence 153 Jan  8 00:25 url.sh
-rw-rw-r--. 1 terence terence 94465 Jan  8 00:25 url.txt
-rw-rw-r--. 1 terence terence 174 Jan  8 07:06 user-clean.sh
-rw-rw-r--. 1 terence terence 156 Jan  8 07:06 user.sh
-rw-rw-r--. 1 terence terence 236 Jan  8 00:25 vpn-ike.sh
-rw-rw-r--. 1 terence terence 189 Jan  8 00:25 vpn-ipsec-route.sh
-rw-rw-r--. 1 terence terence 262 Jan  8 00:25 vpn-ipsec.sh
drwxrwxr-x. 2 terence terence 4096 Jan  8 00:25 xml
-rw-rw-r--. 1 terence terence 156 Jan  8 00:25 zone.sh
$
```

3. You have the scripts generated for security and NAT policies. Run them individually and observe the added rules on Web UI.

```
$ sh nat.sh
$
```

Script to generate PA XML config for objects, policy rules and user-ID mappings



	Name	Tags	Source Zone	Destination Zone	Destination Interface	Original Packet		Service	Translated Packet	
						Source Address	Destination Address		Source Translation	Destination Translation
1	NAT rule 0001	none	trust	untrust	any	10.10.0.0/24	20.20.0.0/24	any	none	none
2	NAT rule 0002	none	trust	untrust	any	10.10.0.0/24	20.20.1.0/24	any	none	none
3	NAT rule 0003	none	trust	untrust	any	10.10.0.0/24	20.20.2.0/24	any	none	none
4	NAT rule 0004	none	trust	untrust	any	10.10.0.0/24	20.20.3.0/24	any	none	none
5	NAT rule 0005	none	trust	untrust	any	10.10.0.0/24	20.20.4.0/24	any	none	none
6	NAT rule 0006	none	trust	untrust	any	10.10.0.0/24	20.20.5.0/24	any	none	none
7	NAT rule 0007	none	trust	untrust	any	10.10.0.0/24	20.20.6.0/24	any	none	none
8	NAT rule 0008	none	trust	untrust	any	10.10.0.0/24	20.20.7.0/24	any	none	none
9	NAT rule 0009	none	trust	untrust	any	10.10.0.0/24	20.20.8.0/24	any	none	none
10	NAT rule 0010	none	trust	untrust	any	10.10.0.0/24	20.20.9.0/24	any	none	none
11	NAT rule 0011	none	trust	untrust	any	10.10.0.0/24	20.20.10.0/24	any	none	none
12	NAT rule 0012	none	trust	untrust	any	10.10.0.0/24	20.20.11.0/24	any	none	none
13	NAT rule 0013	none	trust	untrust	any	10.10.0.0/24	20.20.12.0/24	any	none	none
14	NAT rule 0014	none	trust	untrust	any	10.10.0.0/24	20.20.13.0/24	any	none	none
15	NAT rule 0015	none	trust	untrust	any	10.10.0.0/24	20.20.14.0/24	any	none	none
16	NAT rule 0016	none	trust	untrust	any	10.10.0.0/24	20.20.15.0/24	any	none	none
17	NAT rule 0017	none	trust	untrust	any	10.10.0.0/24	20.20.16.0/24	any	none	none
18	NAT rule 0018	none	trust	untrust	any	10.10.0.0/24	20.20.17.0/24	any	none	none
19	NAT rule 0019	none	trust	untrust	any	10.10.0.0/24	20.20.18.0/24	any	none	none
20	NAT rule 0020	none	trust	untrust	any	10.10.0.0/24	20.20.19.0/24	any	none	none
21	NAT rule 0021	none	trust	untrust	any	10.10.0.0/24	20.20.20.0/24	any	none	none
22	NAT rule 0022	none	trust	untrust	any	10.10.0.0/24	20.20.21.0/24	any	none	none
23	NAT rule 0023	none	trust	untrust	any	10.10.0.0/24	20.20.22.0/24	any	none	none
24	NAT rule 0024	none	trust	untrust	any	10.10.0.0/24	20.20.23.0/24	any	none	none
25	NAT rule 0025	none	trust	untrust	any	10.10.0.0/24	20.20.24.0/24	any	none	none
26	NAT rule 0026	none	trust	untrust	any	10.10.0.0/24	20.20.25.0/24	any	none	none
27	NAT rule 0027	none	trust	untrust	any	10.10.0.0/24	20.20.26.0/24	any	none	none
28	NAT rule 0028	none	trust	untrust	any	10.10.0.0/24	20.20.27.0/24	any	none	none
29	NAT rule 0029	none	trust	untrust	any	10.10.0.0/24	20.20.28.0/24	any	none	none
30	NAT rule 0030	none	trust	untrust	any	10.10.0.0/24	20.20.29.0/24	any	none	none
31	NAT rule 0031	none	trust	untrust	any	10.10.0.0/24	20.20.30.0/24	any	none	none
32	NAT rule 0032	none	trust	untrust	any	10.10.0.0/24	20.20.31.0/24	any	none	none
33	NAT rule 0033	none	trust	untrust	any	10.10.0.0/24	20.20.32.0/24	any	none	none
34	NAT rule 0034	none	trust	untrust	any	10.10.0.0/24	20.20.33.0/24	any	none	none

wget will generate logs and capture PA responses with 2 more files, so we can check for errors.

```
$ ls -l logs/nat*  
-rw-rw-r--. 1 terence terence 753 Jan  8 00:37 logs/nat.log  
-rw-rw-r--. 1 terence terence 76 Jan  8 00:37 logs/nat.log.xml  
$
```

4. Commit, or Revert to running configuration if the result is not expected.

5. The *-clean.sh scripts are prepared to remove added config. Use these scripts if the added config cannot be erased with a few clicks.

User-ID mappings

You also have a script generated for user-ID mapping.

```
$ sh uid.sh  
$
```

Run it and observe the entries with CLI.

```
> show user ip-user-mapping all
```

IP	Vsys	From	User	IdleTimeout(s)	MaxTimeout(s)
2.2.0.7	vsys1	XMLAPI	domain2\user7	34607	34607
4.4.0.26	vsys1	XMLAPI	domain4\user26	34607	34607
3.3.0.18	vsys1	XMLAPI	domain3\user18	34607	34607
2.2.0.27	vsys1	XMLAPI	domain2\user27	34607	34607
3.3.0.28	vsys1	XMLAPI	domain3\user28	34607	34607
4.4.0.20	vsys1	XMLAPI	domain4\user20	34607	34607
3.3.0.9	vsys1	XMLAPI	domain3\user9	34607	34607
3.3.0.19	vsys1	XMLAPI	domain3\user19	34607	34607
3.3.0.7	vsys1	XMLAPI	domain3\user7	34607	34607
4.4.0.10	vsys1	XMLAPI	domain4\user10	34607	34607
3.3.0.24	vsys1	XMLAPI	domain3\user24	34607	34607
1.1.0.18	vsys1	XMLAPI	domain1\user18	34607	34607
4.4.0.14	vsys1	XMLAPI	domain4\user14	34607	34607
2.2.0.6	vsys1	XMLAPI	domain2\user6	34607	34607
4.4.0.7	vsys1	XMLAPI	domain4\user7	34607	34607
3.3.0.22	vsys1	XMLAPI	domain3\user22	34607	34607
1.1.0.14	vsys1	XMLAPI	domain1\user14	34607	34607
2.2.0.23	vsys1	XMLAPI	domain2\user23	34607	34607
2.2.0.20	vsys1	XMLAPI	domain2\user20	34607	34607

References

- [How to scale out a configuration for a POC](#)
- [Script to execute CLI commands at short regular intervals](#)
- [Script to perform CLI `commit` against a list of saved config](#)
- [Scripts to collect browser content or Panachrome repeatedly over time](#)
- [Terence's list of my tech docs](#)