
cisco
You make possible

NSO Access Control

Role-based and Resource-based Access

Fatih Ayvaz

BRKOPS-2700

cisco Live!

Barcelona | January 27-31, 2020

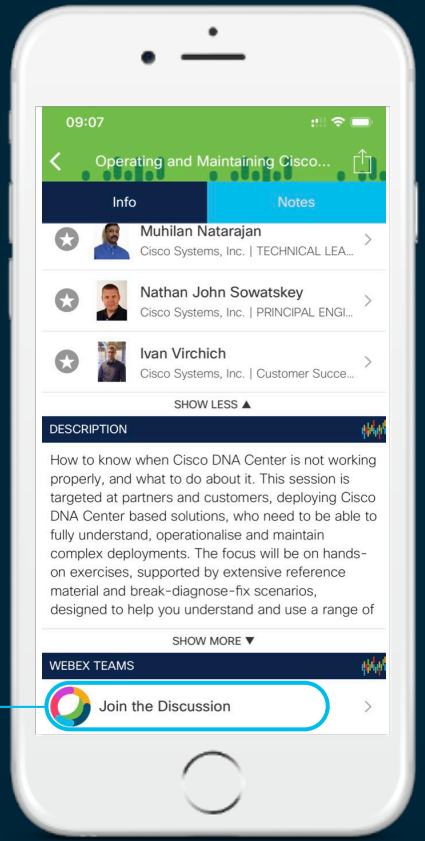
Cisco Webex Teams

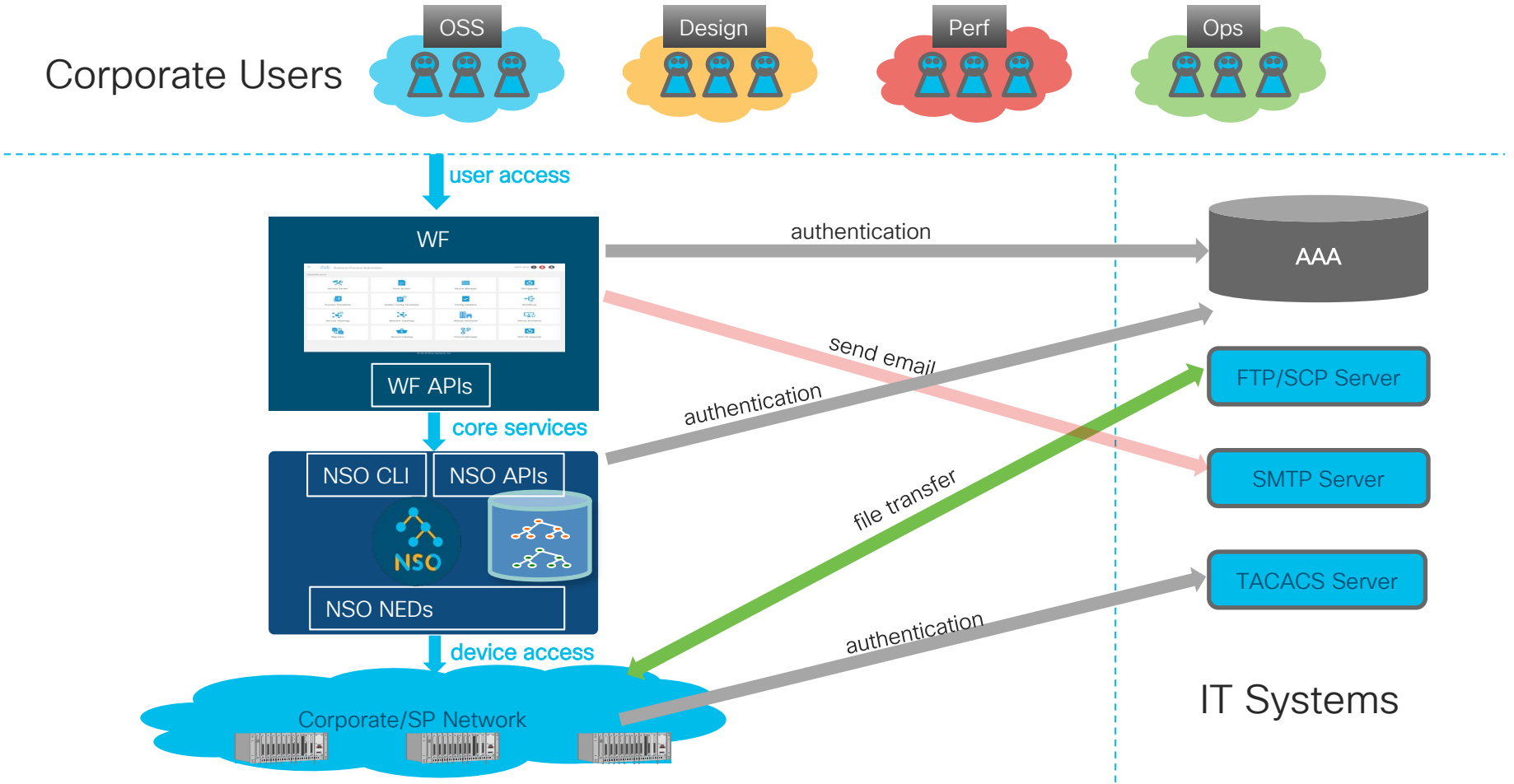
Questions?

Use Cisco Webex Teams to chat with the speaker after the session

How

1. Find this session in the Cisco Events Mobile App
2. Click “Join the Discussion”
3. Install Webex Teams or go directly to the team space
4. Enter messages/questions in the team space



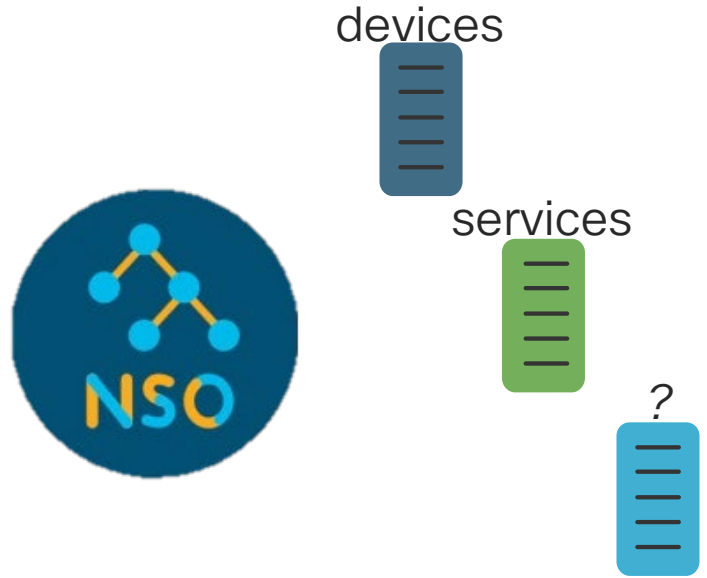


Have you already answered ..

- Who is going to access NSO?
- What are the roles? What are the permissions/restrictions of each role?
  - Is everyone allowed to access (read, write) all devices?
  - How about services?
- Which interfaces will be used to access?
- What information will consumer need to provide to gain access?
- Where are the users and groups associations stored? Which attributes?
- Will all the access interfaces be treated in same way?
- ..

Why do we need to control access to NSO?

Planning	· View devices · View services
OpCo Operations	· Un-deploy services · Add devices
Engineering	· Design service · Add a new vendor
Core Operations	· Configure ASR9K · Replace hw
Edge Operations	· Configure ASR920 · Ping test



Today's session is about ...

We will ...

- Northbound access control
- Configuration options for aaa
- Accessing /devices/device
- Accessing /services
- Troubleshooting
- Examples and hints

We will NOT ...

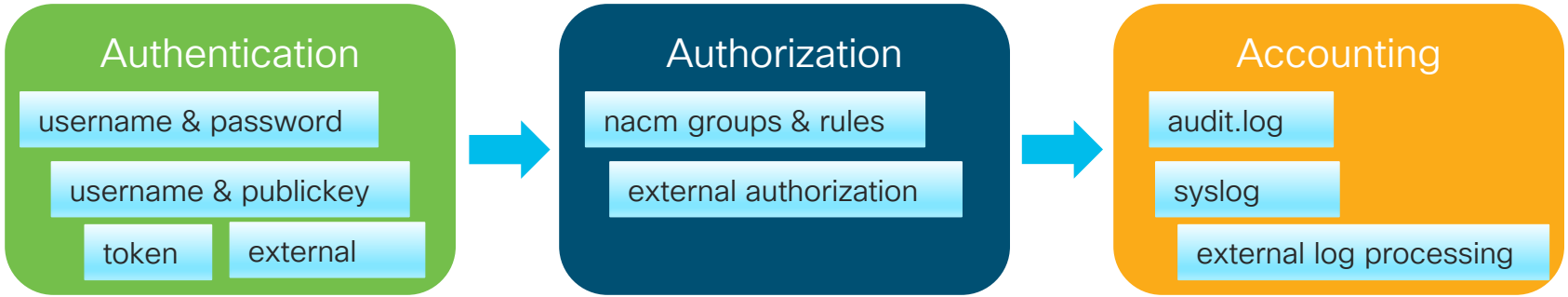
- NSO architecture
- NED capabilities
- NETCONF, RESTCONF, etc.
- NSO service design or FastMap or templates
- Device level access, authgroups
- OpenSSH

Agenda

- Overview of NSO AAA Capabilities
- Overview of NACM Architecture
- Relationship between Users and User Groups
- Assignment of Access Privileges
- Controlling Access to Devices and Services
- Examples from Cisco CX Live Deployments
- Discussions and Conclusion

NSO AAA

AAA functions in NSO



Access interfaces

- CLI >> Console, SSH
- NETCONF >> TCP, SSH (built-in), SSH (OpenSSH)
- RESTCONF >> HTTP
- SNMP
- WEBUI >> HTTP, SSL
- maapi_authenticate()
  - >> username and password
  - maapi_authenticate2() >> + src_addr, src_port, context, and prot
  - * aaa/externalAuthentication/includeExtra

Authentication methods on NSO

- Username and password
  - CLI, NETCONF, RESTCONF, SNMP, WebUI
  - External authentication, PAM, local authentication
  - Authentication order in ncs.conf: e.g. <auth-order>external-authentication local-authentication</auth-order>
- Public key
  - CLI, NETCONF
- Token validation
  - RESTCONF
  - External validation

Built-in SSH Server

Built-in SSH server

- Modeled in tailf-ncs-ssh.yang
- Supports DSA, RSA, EDDSA (ED25519) key types.
- Controls ssh host keys fetched from devices.
- Access methods using SSH:
  - NETCONF (TCP: 2022)
  - CLI (TCP: 2024)

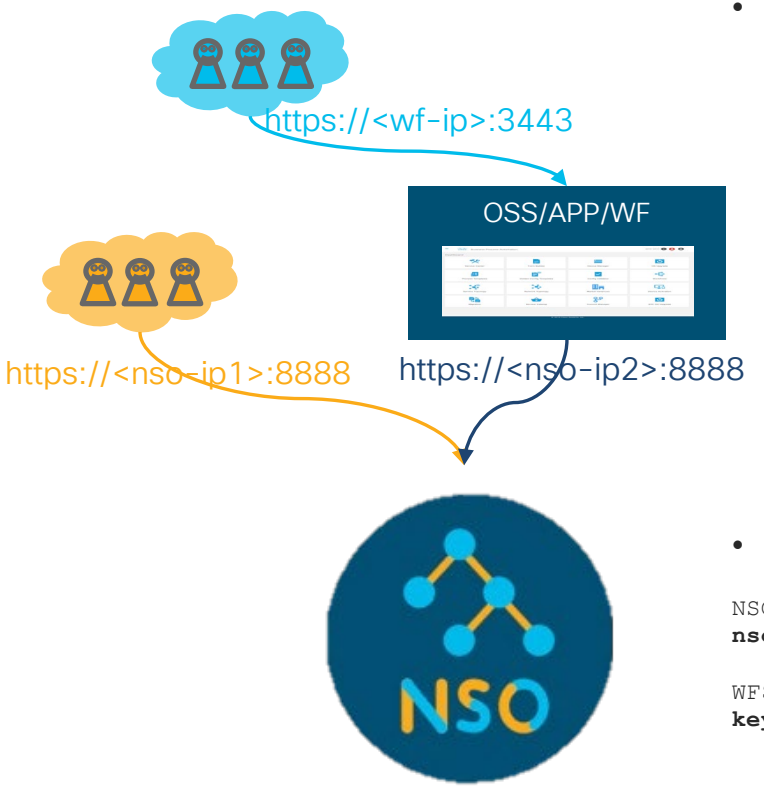
SSH configs to consider

```
FAAYVAZ-M-J0S2:ssh faayvaz$ ls
/Users/faayvaz/NSO-5.3-
20200108/etc/ncs/ssh
ssh_host_ed25519_key
        ssh_host_ed25519_key.pub
```

- SSH server keys
  - /ncs-config/aaa/ssh-server-key-dir: ${NCS_DIR}/etc/ncs/ssh
- Duration to close ssh session
  - /ncs-config/aaa/ssh-login-grace-time [PT10M]
- Max number of attempts to close ssh session
  - /ncs-config/aaa/ssh-max-auth-tries [unbounded]
- Public key authentication method
  - /ncs-config/aaa/ssh-pubkey-authentication: (none, local, *system)
- User ssh keys:
  - local: /aaa/authentication/users/user{$USER}/ssh_keydir
  - system: $HOME/.ssh

WEBUI

WEBUI configs to consider



- SSL settings in ncs.conf
  - /ncs-config/webui/transport/ssl/enabled
  - /ncs-config/webui/transport/ssl/key-file (string)
    - `<key-file>/etc/ncs/ssl/cert/`**`nso_acme_com.key`**`</key-file>`
  - /ncs-config/webui/transport/ssl/cert-file (string)
    - `<cert-file>/etc/ncs/ssl/cert/`**`nso_acme_com.cer`**`</cert-file>`
  - /ncs-config/webui/transport/ssl/ca-cert-file (string)
    - `<ca-cert-file>/etc/ncs/ssl/cert/`**`CACert.cer`**`</ca-cert-file>`
  - /ncs-config/webui/transport/ssl/protocols (string)
    - `<protocols>`**`tlsv1.2`**`</protocols>`
- Verify server-side and client-side certs with openssl!

```
NSO$ openssl s_client -connect nso.acme.com:8888 -cert nso_acme_com.cer -key
nso_acme_com.key

WF$ openssl s_client -connect wf.acme.com:3443 -cert /opt/wf/wf_acme_com.cer -
key /opt/wf/wf_acme_com.key
```

- INSTALL client-side certificates!
- Edit /etc/hosts entries for hostnames!

IPC Access

IPC port access

- Client libraries connect. E.g.: ncs_cli, ncs_load, netconf-subsys, etc.
- Users with shell access are trusted (by default)
  - User must be in a linux group which is allowed.
- Configuration options
  - /ncs-config/ncs-ipc-address/ip (ipv4-address | ipv6-address) [127.0.0.1]
  - /ncs-config/ncs-ipc-address/port (port-number) [4569]
- Restricting IP access:
  - /ncs-config/ncs-ipc-access-check/enabled (boolean) [false]
  - /ncs-config/ncs-ipc-access-check/filename (string)
  - The file should be protected via OS file permissions.
  - Client should set environment variable NCS_IPC_ACCESS_FILE.
  - IMPORTANT! if this is set, and ipc-access-check is disabled, client connection will fail!

Local Authentication

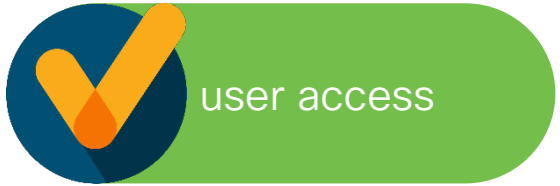
AAA settings in ncs.conf


```
<aaa>
  <ssh-server-key-dir>${NCS_DIR}/etc/ncs/ssh</ssh-server-key-dir>
  <!-- Depending on OS - and also depending on user requirements -->
  <!-- the pam service value value must be tuned. -->
  <pam>
    <enabled>true</enabled>
    <service>common-auth</service>
  </pam>
  <external-authentication>
    <enabled>false</enabled>
    <executable>my-test-auth.sh</executable>
  </external-authentication>

  <local-authentication>
    <enabled>true</enabled>
  </local-authentication>

</aaa>
```

AAA (default for local install) configuration in CDB

```
admin@ncs> show configuration aaa authentication users user admin
uid         65534;
gid         65534;
password
$6$EUPCDnuhJwIFZk9c$m/IKonkOTNm0KbeRb4BTIUsq9I6XrKNbvd3UKowava904mWdbVxvT7C/X8aAKgwb598mrHwS05ewyc5f
/pQfU1;
ssh_keydir /var/ncs/homes/admin/.ssh;
homedir    /var/ncs/homes/admin;
[ok][2020-01-09 12:30:59]
admin@ncs> show configuration aaa authentication users user oper
uid         65534;
gid         65534;
password
$6$Ey9GYFlUcF3TgkZY$rlx5OteS.bRXfzxouX7EEunzKgz5.xR2TWWxsYCR3wKwBKjOJhz7BN68OLulSEk8VRjHhynMskzFR/Sb
zyJYd1;
ssh_keydir /var/ncs/homes/oper/.ssh;
homedir    /var/ncs/homes/oper;
[ok][2020-01-09 12:35:28]
admin@ncs>
```

Verification of username/password authentication



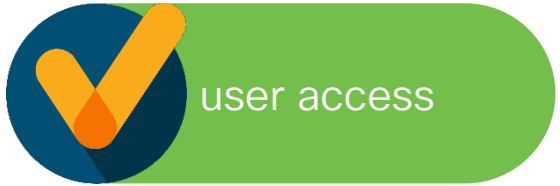

```
ncs> show configuration aaa authentication users user admin
password   $6$EUPCDnuhJwIFZk9c$m//pQfU1;
ssh_keydir /var/ncs/homes/admin/.ssh;
```

```
$ ssh admin@localhost -p 2024
admin@localhost's password:****

admin connected from 127.0.0.1 using ssh on FAAYVAZ-M-J0S2
admin@ncs>
```

```
[withheld]/0 login failed via cli from 127.0.0.1:57488 with ssh:
Couldn't read "/var/ncs/homes/admin/.ssh/authorized_keys2": "no
such file or directory"
admin/0 local authentication succeeded via cli from 127.0.0.1:57488
with ssh, member of groups: admin
admin/0 logged in via cli from 127.0.0.1:57488 with ssh using local
authentication
admin/50 assigned to groups: admin
admin/50 created new session via cli from 127.0.0.1:57488 with ssh
```

Disable public key for oper


```
admin@ncs% set aaa authentication users user oper ssh_keydir ""
```




```
$ ssh oper@localhost -p 2024
oper@localhost's password:****

oper connected from 127.0.0.1 using ssh on FAAYVAZ-M-J0S2
oper@ncs>
```


```
<INFO> .. audit user: oper/0 local authentication succeeded via
cli from 127.0.0.1:58713 with ssh, member of groups: oper
<INFO> .. audit user: oper/0 logged in via cli from
127.0.0.1:58713 with ssh using local authentication
<INFO> .. audit user: oper/54 assigned to groups: oper
<INFO> .. audit user: oper/54 created new session via cli from
127.0.0.1:58713 with ssh
```

Public Key Authentication

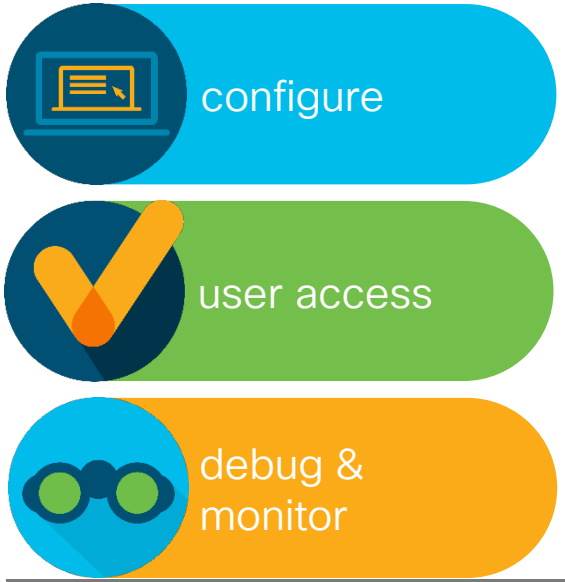
Enable public key for a “local user"

```
admin@ncs% set aaa authentication users user faayvaz ssh_keydir "/Users/faayvaz/.ssh"
Value for 'uid' (<int>): 65534
Value for 'gid' (<int>): 65534
Value for 'password' (<hash digest string>): ****
Value for 'homedir' (<string>): "/Users/faayvaz"
admin@ncs% commit
```

```
cat /Users/faayvaz/.ssh/id_rsa.pub >> /Users/faayvaz/.ssh/authorized_keys

FAAYVAZ-M-J0S2: faayvaz$ ssh faayvaz@localhost -p 2024
faayvaz connected from 127.0.0.1 using ssh on FAAYVAZ-M-J0S2
faayvaz@ncs>
```

```
<INFO> 9-Jan-2020::12:57:52.973 FAAYVAZ-M-J0S2 ncs[95259]: audit user: faayvaz/0 logged in via cli
from 127.0.0.1:60007 with ssh using publickey authentication
<INFO> 9-Jan-2020::12:57:52.981 FAAYVAZ-M-J0S2 ncs[95259]: audit user: faayvaz/58 assigned to groups:
<INFO> 9-Jan-2020::12:57:52.981 FAAYVAZ-M-J0S2 ncs[95259]: audit user: faayvaz/58 created new session
via cli from 127.0.0.1:60007 with ssh
```

Enable public key for a non-local user




```
admin@ncs% delete aaa authentication users user faayvaz
admin@ncs% commit
```

```
$ ssh faayvaz@localhost -p 2024

faayvaz connected from 127.0.0.1 using ssh on FAAYVAZ-M-J0S2
faayvaz@ncs>
```

```
<INFO> 9-Jan-2020::13:43:01.390 FAAYVAZ-M-J0S2 ncs[95259]: audit user: faayvaz/0 logged in via cli from
127.0.0.1:63124 with ssh using publickey authentication
<INFO> 9-Jan-2020::13:43:01.392 FAAYVAZ-M-J0S2 ncs[95259]: audit user: faayvaz/61 assigned to groups:
_appserveradm,staff,admin,_appserverusr,_lpadmin
<INFO> 9-Jan-2020::13:43:01.392 FAAYVAZ-M-J0S2 ncs[95259]: audit user: faayvaz/61 created new session
via cli from 127.0.0.1:63124 with ssh
```

External Authentication

External authentication

- LDAP, RADIUS, TACACS
- Python scripting

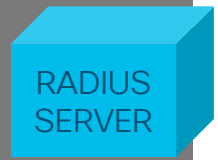
```
<aaa>
  <ssh-server-key-dir>${NCS_DIR}/etc/ncs/ssh</ssh-server-key-dir>
  <external-authentication>
    <enabled>true</enabled>
    <executable>/Users/faayvaz/NSO-5.3-20200108/external-authentication-for-demo.py</executable>
  </external-authentication>
  <auth-order>external-authentication local-authentication</auth-order>
</aaa>
```

```
$ ls -al /Users/faayvaz/NSO-5.3-20200108/external-authentication-for-demo.py
lrwxr-xr-x  1 faayvaz  staff  77 Jan 13 13:18 /Users/faayvaz/NSO-5.3-20200108/external-
authentication-for-demo.py -> /Users/faayvaz/Documents/CiscoLive/CL2020/external-authentication-for-
demo.py
```

External authentication with RADIUS

- Add NSO IP address as a RADIUS NAS client on RADIUS server
- Get the radius secret

```
Cisco Access Registrar 5.1.0.10 Configuration Utility
--> cd /radius/clients
--> add nso-15 "" radius <NSO-IP> cisco NAS
--> cd /radius/userlists/nsoUsers/
--> add test5 "" test5 TRUE "" prNSO1
--> ls -R test5
[ test5 ]
    Name = test5
    Password = <encrypted>
    BaseProfile~ = prNSO1
--> ls -R /radius/profiles/prNSO1
[ /Radius/Profiles/prNSO1 ]
    Name = prNSO1
    Description =
    Attributes/
          Callback-Id = "ncsoper 1010 500 501 502"
--> save
```




```
<external-authentication>
  <enabled>true</enabled>
  <executable>${NCS_RUN_DIR}/radauth.py</executable>
  rnal-authentication>
```

```
o-faayvaz:~$ ssh test5@<NSO-IP> -p 2024
test5@172.16.13.15's password:
test5 connected from 172.16.13.15 using ssh on nso-faayvaz
test5@ncs>


--- audit.log ---
<INFO> audit user: test5/0 Logged in over ssh using externalauth, member of groups: ncsoper
<INFO> audit user: test5/97 assigned to groups: ncsoper
```

radauth.py


```
def get_credentials():
  # read username and password from stdin
  # comes in [username;password;]\n format

  #Remove [ and ], split on ; and assign to username and password
  username = c.replace("\\n","").strip('"[]').split(";")[0]
  password = c.replace("\\n","").strip('"[]').split(";")[1]
  return (username, password)

def check_credentials(username, password):
   radhost = 'RADIUSSERVER:1812'
   radsecret = 'radius-secret'
   # build and send radclient command
   radcommand = 'echo \"User-Name=%s,User-Password=%s\" | radclient %s auth %s' %
(username,password,radhost,radsecret)
   radresponse = subprocess.Popen(radcommand, stdout=subprocess.PIPE, shell=True)
   (reply, err) = radresponse.communicate()

   if reply.find("Callback-Id") > 0 :
       authinfo = reply.split('"')
       accept = "accept %s /home/%s" % (authinfo[1],username)
       acceptstr = "Returning: %s\n" % accept
       return accept
```

External authentication with TACACS

- Add NSO IP address as a TACACS client
- Get the tacacs shared key

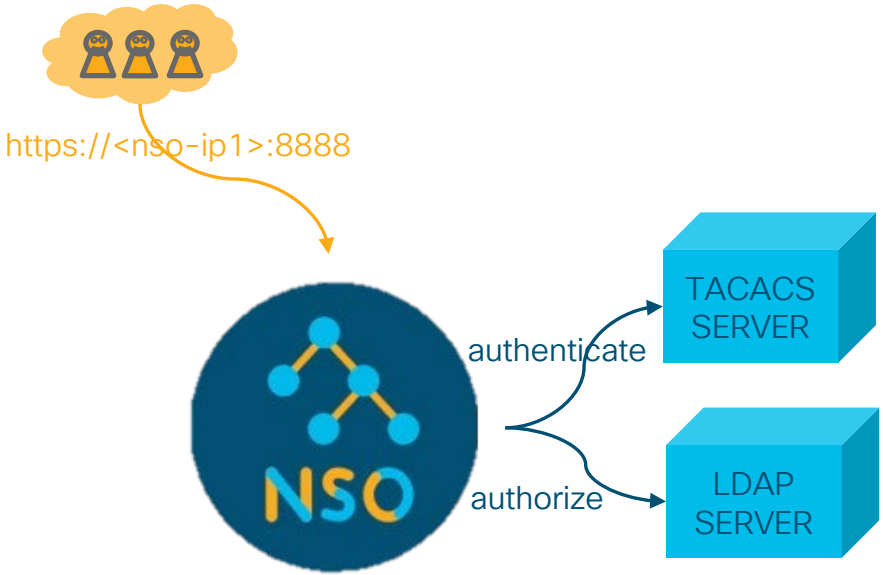
```
'cisco-av-pair=shell:domains = group1'
```


```
<external-authentication>
  <enabled>true</enabled>
  <executable>${NCS_RUN_DIR}/authTACACS.py</executable>
</external-authentication>
```

authTACACS.py

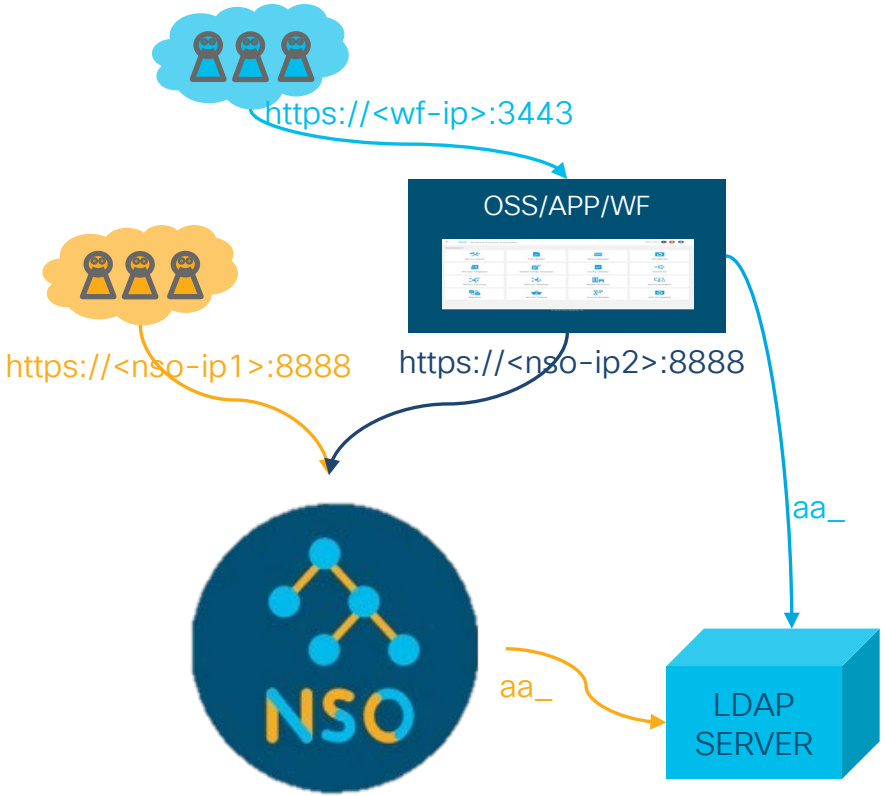

```
credentialstring = sys.stdin.readline()[:-2][1:]
credentials = credentialstring.split(';')
user = credentials[0]
password = credentials[1]

tacacs_host = '10.A.B.C'
shared_key = '***'
cli = TACACSClient(tacacs_host, 49, shared_key,
                           timeout=60, family=socket.AF_INET)
authenticate = cli.authenticate(username, password)

if authenticate.valid:
   response = cli.authorize(username, arguments=cmds)
   regex = re.compile(r".*domains=(\S+)")
   groups += regex.search(arg).groups()[0] + " "
   accept = "accept {} 1004 1004 /opt/ncs/ncs-run\n".format(groups)
   print(accept)
else:
    print "reject invalid password\n"
```

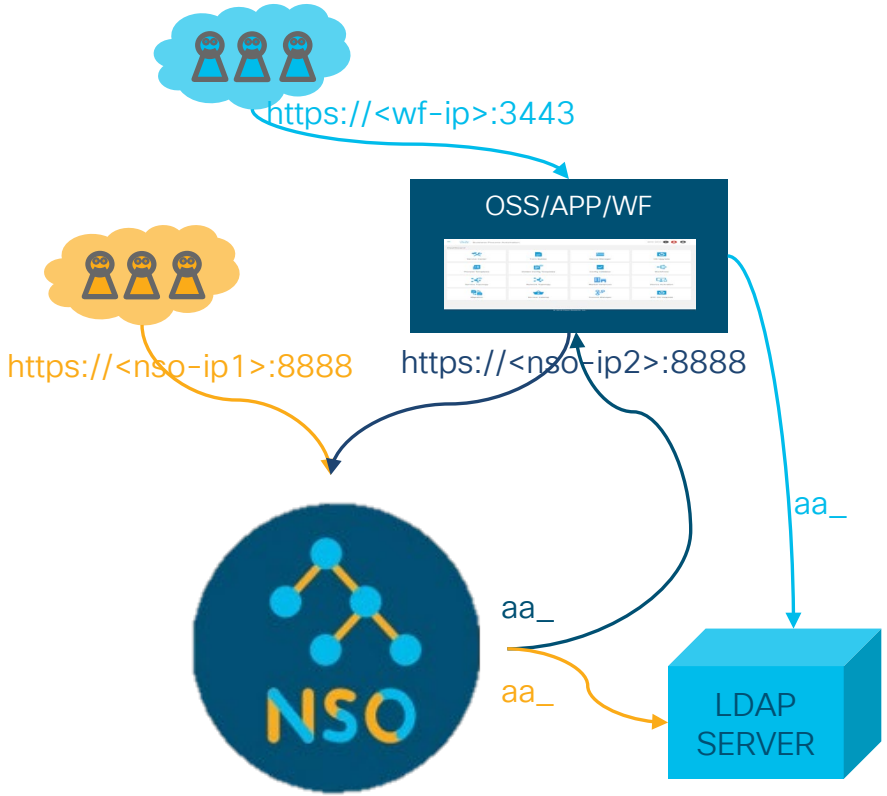
External authentication with multiple servers



- What is the requirement to authenticate/authorize with another server?
- When can this be useful?

How about this?





When authentication happens

- External authentication (with a custom python script) via LDAP, RADIUS, TACACS:
  - "accept $groups $uid $gid $supplementary_gids $HOME\n”
  - To return a token: "accept_token $groups $uid $gid $supplementary_gids $HOME $token\n”
  - Other options: accept_info, accept_warning, accept_token_info, accept_token_warning
- External token validation (RESTCONF only):
  - "accept $groups $uid $gid $supplementary_gids $HOME $USER\n”
  - To return a token: "accept_token $groups $uid $gid $supplementary_gids $HOME $USER $token\n”
  - Other options: accept_info, accept_warning, accept_token_info, accept_token_warning
- Monitor in audit.log file:
  - demouser1/0 logged in via netconf from 127.0.0.1:55779 with ssh using publickey authentication
  - nsoadmin/0 logged in via webui from 127.0.0.1:52135 with http using local authentication
  - demouser1/0 logged in via rest from 127.0.0.1:55363 with http using externalvalidation authentication

When authentication fails

- If authentication/validation fails:
  - “reject”: will try next method (in auth-order or validation-order)
  - “abort”: fails immediately!
  - /ncs-config/aaa/audit-user-name (always | known | never) [known]

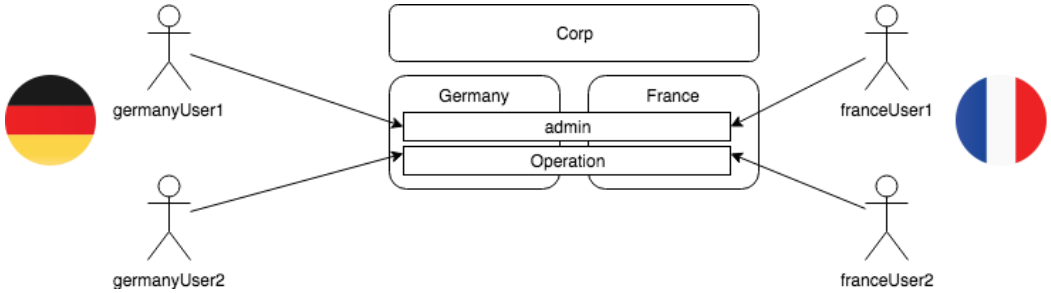
```
try:
   ldap_client.simple_bind_s(ldap_username, password)
except ldap.INVALID_CREDENTIALS:
    if username == ”privileged-user":
       print("reject INVALID_CREDENTIALS: privileged-user : invalid ldap credentials")
    else:
       print("abort INVALID_CREDENTIALS: invalid ldap credentials")
except ldap.CONSTRAINT_VIOLATION:
    print("reject CONSTRAINT_VIOLATION: ldap constraint violation")
except ldap.SERVER_DOWN:
    print("reject SERVER_DOWN: ldap server not accessible")
except ldap.LDAPError:
    print("abort LDAPError")
```

NSO Authorization

Authorization on NSO

- What is authorized and how?
- Authorization of commands (cmdrule)
  - CLI and WebUI commands and operations
- Authorization of data access (rule)
  - RPC
  - Notifications
  - Data nodes
- Group membership
  - Role based authorization
  - /nacm/groups
  - /etc/group
  - (+) any group from authentication
  - /ncs-config/aaa/default-group, if empty and set so!
- NACM (RFC 8341)
  - Hyperlink: https://tools.ietf.org/html/rfc8341
- Tail-f ACM
  - CLI commands
  - Support for ”context”
- Tail-f NCS ACM
  - NACM options for services

Example scenario: multi-tenant users/groups



LDAP Structure

Group	User
Corp	corpUser1 corpUser2
Germany	germanyUser1 germanyUser2
France	franceUser1 franceUser2

Group membership: /nacm/groups

```
set nacm groups group Corp user-name [ corpUser1 corpUser2 ]
set nacm groups group Germany gid 1003 user-name [ germanyUser1 germanyUser2 ]
set nacm groups group France gid 1004 user-name [ franceUser1 franceUser2 ]
```

```
<config xmlns="http://tail-f.com/ns/config/1.0">
  <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-
netconf-acm">
  <groups>
  <group>
    <name>Corp</name>
    <user-name>corpUser1</user-name>
    <user-name>corpUser2</user-name>
  </group>
  </groups>
  </nacm>
</config>
```

```
<config xmlns="http://tail-f.com/ns/config/1.0">
  <nacm xmlns="urn:ietf:params:xml:ns:yang:ietf-
netconf-acm">
  <groups>
  <group>
    <name>France</name>
    <user-name>franceUser1</user-name>
    <user-name>franceUser2</user-name>
    <gid xmlns="http://tail-f.com/yang/acm">1004</gid>
  </group>
  </groups>
  </nacm>
</config>
```

/nacm/groups/group is augmented!

tailf-acm.yang

```
module tailf-acm {
  namespace "http://tail-f.com/yang/acm";
  prefix tacm;

  import ietf-netconf-acm {
    prefix nacm;
  }

  organization "Tail-f Systems";

  description
    "This module augments ietf-netconf-acm with additional
     access control data.

     Copyright 2019 Cisco Systems, Inc.
     All rights reserved.
     Permission is hereby granted to redistribute this file without
     modification.";

  revision 2013-03-07 { ... }

  revision 2012-11-08 { ... }

  augment /nacm:nacm { ... }

  augment /nacm:nacm/nacm:groups/nacm:group {
    leaf gid {
      type int32;
      description
        "This leaf associates a numerical group ID with the group.
         When a OS command is executed on behalf of a user,
         supplementary group IDs are assigned based on 'gid' values
         for the groups that the use is a member of.";
    }
  }
```

```
augment /nacm:nacm/nacm:rule-list {

  list cmdrule {
    key "name";
    ordered-by user;
    description
      "One command access control rule. Command rules control access
       to CLI commands and Web UI functions.

       Rules are processed in user-defined order until a match is
       found.  A rule matches if 'context', 'command', and
       'access-operations' match the request.  If a rule
       matches, the 'action' leaf determines if access is granted
       or not.";

    leaf name {
      type string {
        length "1..max";
      }
      description
        "Arbitrary name assigned to the rule.";
    }

    leaf context {
      type union {
        type nacm:matchall-string-type;
        type string;
      }
      default "*";
      description
        "This leaf matches if it has the value '*' or if its value
         identifies the agent that is requesting access, i.e. 'cli'
         for CLI or 'webui' for Web UI.";
    }

    leaf command {
      type string;
      default "*";
      description
        "Space-separated tokens representing the command. Refer
         to the Tail-f AAA documentation for further details.";
    }

    leaf access-operations {
      type union {
        type nacm:matchall-string-type;
        type nacm:access-operations-type;
      }
      default "*";
      description
        "Access operations associated with this rule.

         This leaf matches if it has the value '*' or if the
         bit corresponding to the requested operation is set.";
    }

    leaf action {
      type nacm:action-type;
      mandatory true;
      description
        "The access control action associated with the
         rule.  If a rule is determined to match a
         particular request, then this object is used
         to determine whether to permit or deny the
         request.";
    }

    leaf log-if-permit {
      type empty;
      description
        "If this leaf is present, access granted due to this rule
         is logged in the developer log. Otherwise, only denied
         access is logged. Mainly intended for debugging of rules.";
    }

    leaf comment {
      type string;
      description
        "A textual description of the access rule.";
    }
  }
}
```

Group membership: /etc/passwd

```
[nsoadmin@nso-01 ncs]$ grep nsoadmin /etc/passwd
nsoadmin:x:1000:1000::/home/nsoadmin:/bin/bash
[nsoadmin@nso-01 ncs]$ grep 1000 /etc/group
nsoadmin:x:1000:
[nsoadmin@nso-01 ncs]$ id
uid=1000(nsoadmin) gid=1000(nsoadmin) groups=1000(nsoadmin)
```

```
[nsoadmin@nso-01 ncs]$ ncs_cli
nsoadmin connected from 10.1.1.1 using ssh on nso-01
nsoadmin@nso-s1> conf
                 ^
% Invalid input detected at '^' marker.
nsoadmin@nso-s1> <TAB>
Possible completions:
  exit - Exit the management session
  quit - Exit the management session
nsoadmin@nso-s1>
-- audit.log --
<INFO> 29-Jan-2020::06:46:50.621 nso-01 ncs[1305]: audit user: nsoadmin/471065 assigned to groups: nsoadmin
```

Group membership: /etc/passwd & /nacm/groups

```
[nsoadmin@nso-01 ncs]$ ncs_cli -u admin
admin connected from 10.1.1.1 using ssh on nso-01
admin@nso-s1> show configuration nacm groups group ncs
Possible completions:
  ncsadmin  ncsoper
admin@nso-s1> show configuration nacm groups group ncsadmin
user-name [ admin private root system ];
```

```
-- audit.log --
<INFO> 29-Jan-2020::06:46:50.621 nso-01 ncs[1305]: audit user: nsoadmin/471065 assigned to groups: nsoadmin

<INFO> 29-Jan-2020::06:49:49.243 nso-01 ncs[1305]: audit user: admin/471112 assigned to groups: ncsadmin,nsoadmin
<INFO> 29-Jan-2020::06:50:24.818 nso-01 ncs[1305]: audit user: admin/471112 CLI 'show configuration nacm groups group
ncsadmin'
```

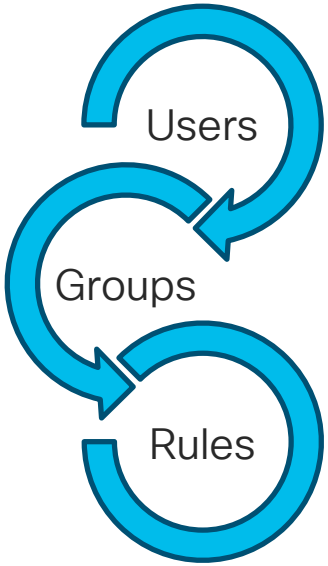
External group assignments can be disabled

```
admin@nso-s1% set nacm enable-external-groups false
admin@nso-s1% commit
```

```
[nsoadmin@nso-01 ncs]$ ncs_cli
nsoadmin connected from 10.1.1.1 using ssh on nso-01
nsoadmin@nso-s1> <TAB>
Possible completions:
  exit - Exit the management session
  quit - Exit the management session
nsoadmin@nso-s1> exit
[nsoadmin@nso-01 ncs]$ ncs_cli -u admin
admin connected from 10.1.1.1 using ssh on nso-01
admin@nso-s1> exit

-- audit.log --
<INFO> 29-Jan-2020::07:04:42.344 nso-01 ncs[1305]: audit user: nsoadmin/471345 assigned to groups:
<INFO> 29-Jan-2020::07:04:52.737 nso-01 ncs[1305]: audit user: admin/471348 assigned to groups: ncsadmin
```

NACM Overview and Examples

Cisco NSO's AAA Model



- AAA Users
- NACM Groups
- NACM Rule-lists
- NACM Rule Structure
  - Enablement
  - Rule-lists
  - Path statements

NACM Rule Types

- Module Rule: e.g. -> module-name = id-allocator
  - Controls access for definitions in a specific YANG module, identified by its name.
- Protocol Operation Rule: e.g. -> rpc-name = edit-config
  - Controls access for a specific protocol operation, identified by its YANG module and name.
- Data Node Rule: e.g. -> path = /devices/device[name='devIOS-0']
  - Controls access for a specific data node and its descendants, identified by its path location within the conceptual XML document for the data node.
- Notification Rule: e.g. -> notification-name = sys-config-change
  - Controls access for a specific notification event type, identified by its YANG module and name.

NSO cmdrule

- Where do these apply?
  - CLI commands

```
cmdrule c-logout action deny
  command logout
!
cmdrule j-logout action deny
  command "request system logout"
 !
```

  - WebUI functions and operations

```
cmdrule permit-jsonrpc-action action permit
  command "::jsonrpc:: action"
cmdrule permit-jsonrpc-run_action action permit
  command "::jsonrpc:: run_action"
cmdrule permit-jsonrpc-logout action permit
  command "::jsonrpc:: logout”
cmdrule permit-jsonrpc-delete action permit
  command "::jsonrpc:: delete"
```

```
augment /nacm:nacm/nacm:rule-list {
    list cmdrule {
      key "name";
      ordered-by user;

      leaf name {

      leaf context {
        default "*";

      leaf command {
        default "*";

      leaf access-operations {
         default "*";

      leaf action {
         mandatory true;

      leaf log-if-permit {

      leaf comment {
    }
  }
```

Authorization order

- For cmdrule
- Check /nacm/enable-nacm
- Traverse rule-list to match groups
  - Traverse cmdrule rules
- if read; check /nacm/cmd-read-default
- if exec; check /nacm/cmd/exec-default

- For rule
- Check /nacm/enable-nacm
- Traverse rule-list to match groups
  - Traverse rule rules
- Check NACM extensions in data models:
  - *: data model: nacm:default-deny-all
  - CUD: data model: nacm:default-deny-write
- if read; check /nacm/read-default
- if CUD; check /nacm/write-default
- if exec; check /nacm/exec-default

Order is important!

```
set nacm rule-list rdemogroup35 group demogroup3

set nacm rule-list rdemogroup35 cmdrule request-message-deny command "request message" action deny
access-operations exec

set nacm rule-list rdemogroup35 cmdrule all-cmd-any action permit context * log-if-permit
```

```
demouser3@ncs> request message nsoadmin hi
[ok][2020-01-29 14:19:06]
demouser3@ncs>
nsoadmin@ncs>
Message from demouser3@FAAYVAZ-M-J0S2 at 2020-01-29 14:19:06...
hi

<DEBUG> 29-Jan-2020::14:19:06.232 User: demouser3[demogroup3,demogroupN] Command Rule "rdemogroup3/all-cmd-any" triggered
full_match accept for "request message nsoadmin hi" op execute

nsoadmin@ncs% move nacm rule-list rdemogroup35 before rdemogroup3
nsoadmin@ncs% commit
demouser3@ncs> request message nsoadmin hi
Aborted: permission denied
[error][2020-01-29 14:22:29]

<DEBUG> 29-Jan-2020::14:22:29.114 User: demouser3[demogroup3,demogroupN] Command Rule "rdemogroup35/all-cmd-any" triggered
full_match accept for "request message" op read
<DEBUG> 29-Jan-2020::14:22:29.116 User: demouser3[demogroup3,demogroupN] rejected command "request message nsoadmin hi" op
execute by full_match Command Rule "rdemogroup35/request-message-deny"
```

NACM Default Rule Behavior

When no groups are found (no rule-lists to check) or no matching rules …

- nacm:default-deny-all
- nacm:default-deny-write
- read-default [permit]
- write-default [deny]
- exec-default [permit]
- *cmd-read-default [permit]
- *cmd-exec-default [permit]
- *log-if-permit-default

NSO NACM Rule Format

- Module Name [*]
  - The name of the YANG module where the requested data node is defined.
- Rule Type
  - rpc-name / notification-name / path: If data-node, then path must be checked.
  - “path” (yang:xpath1.0;}: The leaf "path" is an instance-identifier. You should not refer to non-key leafs.
- Access Operations [*]
  - create, *read, update, delete, **exec
    - *read: MUST be permitted, if a **notification** is tied to the node.
    - **exec: MUST be permitted, if an **action** is requested.
- Action
  - permit, deny
- Comment
- *Context
- *Log-if-permit

Module rule example

- The modules loaded:

  admin@ncs> **show status netconf-state capabilities capability**

  capability http://cisco.com/ciscoutils?module=ciscoutils;

  capability http://com/example/l3vpn?module=l3vpn;

  ...

- Deny l3vpn module:
  - **set nacm rule-list rdemogroup3 rule l3vpn-module-deny module l3vpn action deny access-operations ***
  - or:

```
<rule-list>
    <name>rdemogroup3</name>
      <rule>
        <name>l3vpn-module-deny</name>
        <module-name>l3vpn</module-name>
        <access-operations>*</access-operations>
        <action>deny</action>
      </rule>
  </rule-list>
```

```
demouser3@ncs> show vp{hit TAB to auto-complete for vpn}
- - devel.log- -
<DEBUG> 29-Jan-2020::01:21:56.945 FAAYVAZ-M-J0S2 ncs[24451]: devel-aaa User:
demouser3[demogroup3,demogroupN] rejected data access path /l3vpn:vpn op read due to rule
"rdemogroup3/l3vpn-module-deny"
```

RPC rule example

- Check rpc from netconf capabilities. E.g.: edit-config, delete-config, kill-session,
- Deny get-config:

  set nacm rule-list rdemogroup33 group demogroup3

  set nacm rule-list rdemogroup33 rule get-config-deny rpc-name get-config action deny access-operations *

  set nacm rule-list rdemogroup33 cmdrule all-cmd-any action permit context * log-if-permit

```
faayvaz$ netconf-console -u demouser3 -p cisco --host localhost --get-config -x "/devices/device/authgroups"
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
  <rpc-error>
    <error-type>protocol</error-type>
    <error-tag>access-denied</error-tag>
    <error-severity>error</error-severity>
  </rpc-error>
</rpc-reply>
FAAYVAZ-M-J0S2:yang-explorer faayvaz$

-- devel.log -
<DEBUG> 29-Jan-2020::01:43:36.813 FAAYVAZ-M-J0S2 ncs[24451]: devel-aaa User:
demouser3[demogroup3,demogroupN] rejected data access path /nc:get-config op execute due to rule
"rdemogroup33/get-config-deny"
```

Usage of “path”

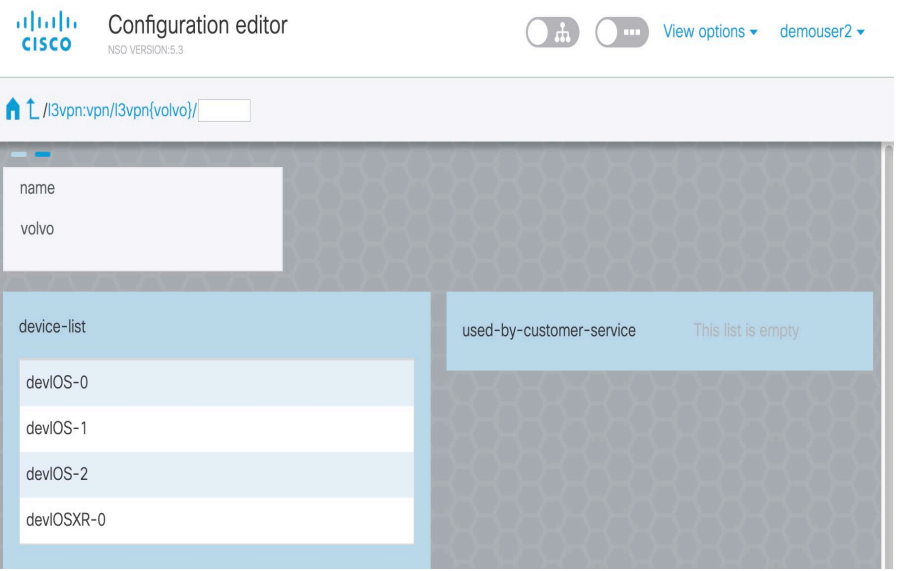
- Tagpaths that are not containing any keys
  - E.g., /ncs/live-device/live-status
- Instantiated key
  - E.g., /devices/device[name=“devIOS-0”]/config/interface
  - E.g., /device/device/config/interface[name=”eth0”]
- Wildcard at end
  - E.g., /services/web-site/*

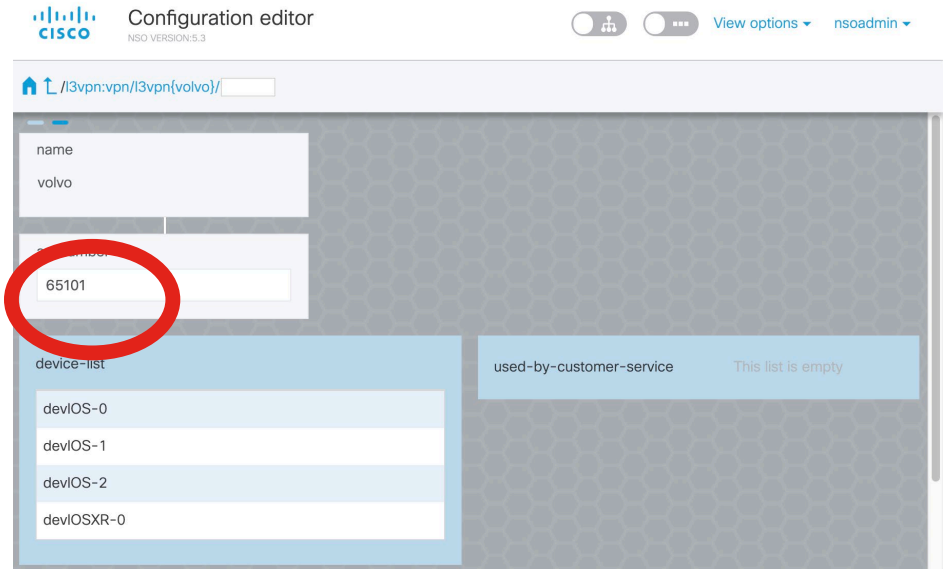
Example for path statement with no-key

- With no-key:

  set nacm rule-list rdemogroup2 group demogroup2

  set nacm rule-list rdemogroup2 rule l3vpn-asn-deny action deny context webui path /vpn/l3vpn/as-number access-operations read





Example for path statement with key

- A specific data node:

  set nacm rule-list device_devIOS rule d_TE412mtu_P_R path /devices/device[name='devIOS-0']/config/ios:interface/TenGigabitEthernet[name='4/1/2']/mtu access-operations read action permit log-if-permit

<DEBUG> 19-Jan-2020::12:53:29.923 FAAYVAZ-M-J0S2 ncs[4816]: devel-aaa User: demouser1[demogroup1,demogroupN] Rule "device_devIOS/d_TE412mtu_P_R" triggered data access accept for path /ncs:devices/device{devIOS-0}/config/ios:interface/TenGigabitEthernet{4/1/2}/mtu op read

Example for path statement with wildcard

- A wildcarded list of elements:

  set nacm rule-list rdemogroup2 group demogroup2

  set nacm rule-list rdemogroup2 rule l3vpn-asn-deny action deny context webui path /vpn/l3vpn/as-number access-operations read

  set nacm rule-list rdemogroup2 rule l3vpn-ford-wc-permit action permit context webui path /vpn/l3vpn[name='ford']/* access-operations * log-if-permit

  set nacm rule-list rdemogroup2 rule l3vpn-ford-permit action permit context webui path /vpn/l3vpn[name='ford'] access-operations * log-if-permit

```
<DEBUG> 28-Jan-2020::22:54:41.264 FAAYVAZ-M-J0S2 ncs[24451]: devel-aaa User:
demouser2[demogroup2,demogroupN] Rule "rdemogroup2/l3vpn-ford-permit" triggered data access accept for path
/l3vpn:vpn/l3vpn{ford}/qos op read
<DEBUG> 28-Jan-2020::22:54:41.265 FAAYVAZ-M-J0S2 ncs[24451]: devel-aaa User:
demouser2[demogroup2,demogroupN] Rule "rdemogroup2/l3vpn-ford-wc-permit" triggered data access accept for
path /l3vpn:vpn/l3vpn{ford}/qos/qos-policy op read
```

RESTCONF Examples

RESTCONF - Device Rules (deny)

- DELETE
- GET
- PATCH
- POST
- PUT

```
GET /restconf/data/tailf-ncs:devices/device=devIOS-0/config/tailf-
ned-cisco-
ios:interface/TenGigabitEthernet=4%2F1%2F2/encapsulation/dot1Q
HTTP/1.1
```

```
    <rule>
        <name>d_TE_D_A</name>
        <path>/devices/device[name='devIOS-
0']/config/ios:interface/TenGigabitEthernet</path>
        <access-operations>*</access-operations>
        <action>deny</action>
        <log-if-permit xmlns="http://tail-f.com/yang/acm"/>
    </rule>
```

```
devel-aaa User: demouser1[demogroup1,demogroupN] rejected data access path
/ncs:devices/device{devIOS-0}/config/ios:interface/TenGigabitEthernet{4/1/2}/encapsulation op
read due to rule "device_devIOS-0/d_TE_D_A"
```

RESTCONF - Device Rules (permit)

- DELETE
- GET
- PATCH
- POST
- PUT

```
GET /restconf/data/tailf-ncs:devices/device=devIOS-0/config/tailf-
ned-cisco-ios:interface/TenGigabitEthernet=4%2F1%2F2/description
HTTP/1.1
```

```
    <rule>
        <name>d_TEdesc_P_R</name>
        <path>/devices/device[name='devIOS-
0']/config/ios:interface/TenGigabitEthernet/description</path>
        <access-operations>read</access-operations>
        <action>permit</action>
        <log-if-permit xmlns="http://tail-f.com/yang/acm"/>
    </rule>
```

```
devel-aaa User: demouser1[demogroup1,demogroupN] Rule "device_devIOS-0/d_TEdesc_P_R" triggered
data access accept for path /ncs:devices/device{devIOS-
0}/config/ios:interface/TenGigabitEthernet{4/1/2}/description op read
```

A rule-list (device operations for restconf)

```
show configuration nacm rule-list device_devIOS-0_restconf
group [ demogroup1 ];
rule d_TEdesc_P_D {
    path              /devices/device[name='devIOS-
0']/config/ios:interface/TenGigabitEthernet/description;
    access-operations delete;
    action            permit;
    context           rest;
    log-if-permit;
}
rule d_TEdesc_P_U {
    path              /devices/device[name='devIOS-
0']/config/ios:interface/TenGigabitEthernet/description;
    access-operations update;
    action            permit;
    context           rest;
    log-if-permit;
}
rule d_TEdesc_P_C {
    path              /devices/device[name='devIOS-
0']/config/ios:interface/TenGigabitEthernet/description;
    access-operations create;
    action            permit;
    context           rest;
    log-if-permit;
}
```

```
rule d_TEdesc_P_E {
    path              /devices/device[name='devIOS-
0']/config/ios:interface/TenGigabitEthernet/description;
    access-operations exec;
    action            permit;
    context           rest;
    log-if-permit;
}
rule d_TEdesc_P_R {
    path              /devices/device[name='devIOS-
0']/config/ios:interface/TenGigabitEthernet/description;
    access-operations read;
    action            permit;
    context           rest;
    log-if-permit;
}
rule d_TEname_P_R {
..
rule d_TE_D_A {
..
rule d_D_CUD {
..
[ok][2020-01-21 13:22:17]
nsoadmin@ncs>
```

HTTP messages for various cases

- Non-existing object: PUT (read, create) ”201 Created”
- Existing object: PUT (read, update) ”204 No Content”
- Existing object: DELETE (delete) ”204 No Content”
- Non-existing object: DELETE (delete) “404 Not Found”
- Non-existing object: PATCH (read) “404 Not Found”
- Existing object: PATCH (read, update) ”204 No Content”
- Non-existing object: GET (read) “404 Not Found”
- Existing object: GET (read) “200 OK”
- Existing object: POST (delete) ”409 Conflict”
- Non-existing object: POST (create) ”201 Created”

NETCONF Examples

Example for rpc get-config

- A specific data node:

  set nacm rule-list rdemogroup33 group demogroup3

  set nacm rule-list rdemogroup33 rule get-config-deny rpc-name get-config action deny access-operations *

  set nacm rule-list rdemogroup33 cmdrule all-cmd-any action permit context * log-if-permit

```
netconf-console -u demouser3 -p cisco --host localhost --get-config -x "/devices/device/authgroups"

- - devel.log - -
<DEBUG> 29-Jan-2020::01:43:36.813 FAAYVAZ-M-J0S2 ncs[24451]: devel-aaa User:
demouser3[demogroup3,demogroupN] rejected data access path /nc:get-config op execute due to rule
"rdemogroup33/get-config-deny"
```

Example for context netconf

```
<rule>
        <name>d_TEdesc_P_D</name>
        <path>/devices/device[name='devIOS-
0']/config/ios:interface/TenGigabitEthernet/description</path>
        <access-operations>delete</access-operations>
        <action>permit</action>
        <context xmlns="http://tail-f.com/yang/acm">netconf</context>
        <log-if-permit xmlns="http://tail-f.com/yang/acm"/>
</rule>
```

Example for rpc edit-config

```
<rule>
        <name>d_TE_P_EDIT</name>
        <rpc-name>edit-config</rpc-name>
        <path>/devices/device[name='devIOS-0']/config/ios:interface/TenGigabitEthernet</path>
        <action>permit</action>
        <context xmlns="http://tail-f.com/yang/acm">netconf</context>
        <log-if-permit xmlns="http://tail-f.com/yang/acm"/>
</rule>
```

WEBUI Examples

NACM examples for WEBUI access

- Don’t allow showing hashed passwords on WEBUI, and allow for self password change:
  - Allow usernames on WEBUI

```
set nacm rule-list rsecuredadmin rule webui-usernames-allow module-name tailf-aaa path
/aaa/authentication/users/user/name access-operations read action permit context webui log-if-permit
```

  - Allow self-password change on WEBUI:

```
set nacm rule-list rsecuredadmin rule webui-self-password-change-allow module-name tailf-aaa path
/aaa/authentication/users/user[name='$USER']/change-password access-operations create action permit context
webui log-if-permit
```

  - Deny access to see other user’s aaa information:

```
set nacm rule-list rsecuredadmin rule webui-users-deny module-name tailf-aaa path
/aaa/authentication/users/user access-operations * context webui action deny
```

  - Allow running actions on WEBUI:

```
set nacm rule-list rsecuredadmin cmdrule webui-cmd-runaction-allow command "::jsonrpc:: run_action" access-
operations exec action permit context webui log-if-permit
```

```
set nacm rule-list rsecuredadmin cmdrule webui-cmd-action-allow command "::jsonrpc:: action" access-
operations exec action permit context webui log-if-permit
```

Example for webui rule-list

```
set nacm rule-list rsecuredadmin rule webui-usernames-allow module-name tailf-aaa path
/aaa/authentication/users/user/name access-operations read action permit context webui log-if-permit

set nacm rule-list rsecuredadmin rule webui-self-password-change-allow module-name tailf-aaa path
/aaa/authentication/users/user[name='$USER']/change-password access-operations create action permit context
webui log-if-permit

set nacm rule-list rsecuredadmin rule webui-users-deny module-name tailf-aaa path
/aaa/authentication/users/user access-operations * context webui action deny

set nacm rule-list rsecuredadmin rule webui-read-allow path / access-operations read action permit context
webui log-if-permit

set nacm rule-list rsecuredadmin cmdrule webui-cmd-logout-allow command "::jsonrpc:: logout" access-
operations exec action permit context webui log-if-permit

set nacm rule-list rsecuredadmin cmdrule webui-cmd-runaction-allow command "::jsonrpc:: run_action" access-
operations exec action permit context webui log-if-permit

set nacm rule-list rsecuredadmin cmdrule webui-cmd-action-allow command "::jsonrpc:: action" access-
operations exec action permit context webui log-if-permit

set nacm rule-list rsecuredadmin cmdrule webui-cmd-any action permit context webui log-if-permit
```

CLI Examples

Example for cli rule-list

- Rules:

```
set nacm rule-list rsecuredadmin rule cli-self-password-allow module-name tailf-aaa path
/aaa/authentication/users/user[name='$USER']/password access-operations * context cli action permit

set nacm rule-list rsecuredadmin rule cli-password-deny module-name tailf-aaa path
/aaa/authentication/users/user/password access-operations * context cli action deny

set nacm rule-list rsecuredadmin rule cli-aaa-allow module-name tailf-aaa path
/aaa/authentication/users access-operations * context cli action permit

set nacm rule-list rsecuredadmin cmdrule cli-cmd-any action permit context cli log-if-permit
```

Service Examples

Example for service module l3vpn:vpn

- Rules around service module:

  set nacm rule-list rdemogroup2 group demogroup2

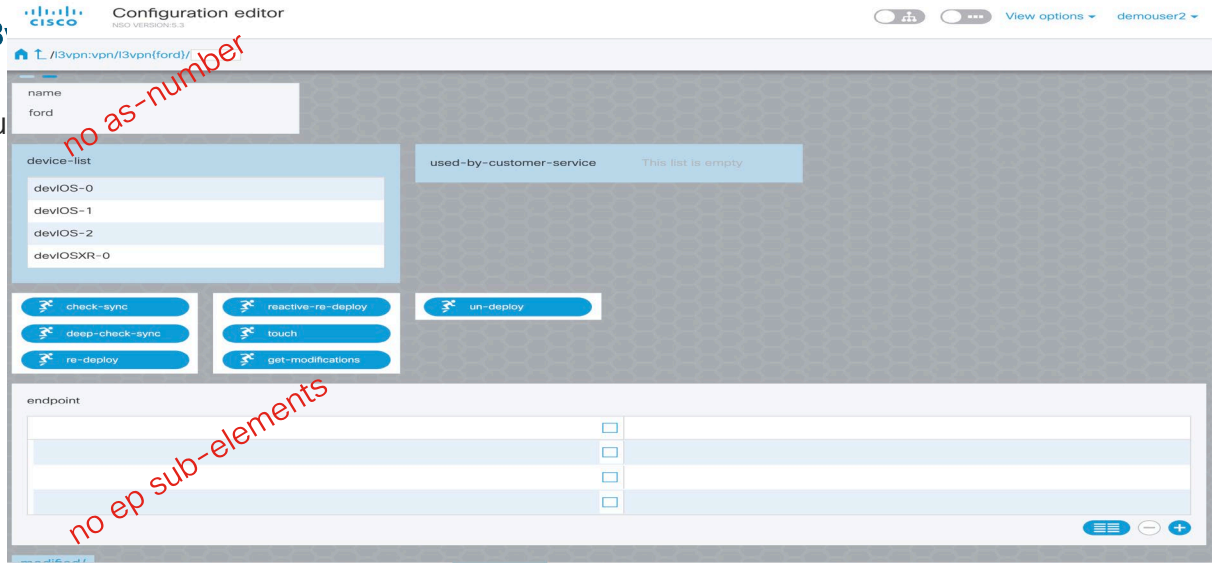
  set nacm rule-list rdemogroup2 rule l3vpn-asn-deny action deny context webui path /vpn/l3vpn/as-number access-operations read

  set nacm rule-list rdemogroup2 rule l3vpn-ford-ep-deny action deny context webui path /vpn/l3vpn[name='ford']/endpoint/* access-operations *

  set nacm rule-list rdemogroup2 rule l3vpn-ford-wc-permit action permit context webui path /vpn/l3vpn[name='ford']/* access-operations * log-if-permit

  set nacm rule-list rdemogroup2 rule l3
  access-operations * log-if-permit

  set nacm rule-list rdemogroup2 cmdru



Example for service with composite key

- Rules around service module:

```
rule elan-epl_device1_11003
  path /ncs:services/elan-epl:elan-epl[edu-device=device1][vc-id = 11003]
  access-operations *
  action permit
rule etree-epl_device1_11004
  path /ncs:services/etree-epl:etree-epl[edu-device=device1][vc-id = 11004]
  access-operations *
  action permit
```

Multi-tenancy Examples

NACM Rule Example – multi-tenant

```
rule-list tenant_Germany {
    group [ tenant_Germany ];
    rule tenant_Germany_nacm {
        path   /nacm/rule-list[name='tenant_Germany']/*;
        action permit;
    }
    rule dryrun {
        path              /services/commit-dry-run;
        access-operations *;
        action            permit;
    }
    rule permitMacAddressLookup {path /mef-common:icl-config/*; action permit;}
    rule permitGetTenantAction {path /mef-common:get-devicetenant; action permit;}
    rule device rules
      // devices/device path rules for this tenant
    rule service rules
      // service instance path rules for this tenant
 {
…
```

Testing and Monitoring

Testing and Monitoring Tips

- Log files: audit-log, devel.log, ncs.log, xpath-trace
  - localhost:8080.access for RESTCONF and WebUI
  - netconf.log for NETCONF
  - ncserr.log: ncs --printlog ncserr.log.1
- Consider log-if-permit in NACM rules
- ssh <user>@localhost -p 2024;or netconf-console --get-config -x '/nacm/groups'
  - Authentication method
  - Authentication order
  - Health and behavior of external authentication script
  - Verification of group assignments
- Review data models
  - ietf-netconf-acm, tailf-aaa, tailf-acm, tailf-ncs-acm, tailf-ncs-ssh, etc.
- Check NSO documentation
  - nso_man, nso_admin, nso_northbound, nso_webui, nso_development, etc.

Suggested path to create rules

- Determine the “context” (webui, netconf, cli, rest, etc.)
- Create whitelist of resources (devices, services, modules, etc.) with log-if-permit
  - Use cmdrule or rule
  - Where can you check that?
- Determine access operations
  - Perform the operation (to be allowed or denied)
  - Monitor devel.log (could be multiple: {read,exec}) or {create}, etc.
  - Match the path for data node rules.

```
<DEBUG> 29-Jan-2020::14:59:19.985 FAAYVAZ-M-J0S2 ncs[24451]: devel-aaa User:
demouser3[demogroup3,demogroupN] Command Rule "rdemogroup35/all-cmd-any" triggered full_match accept for
"::jsonrpc:: logout" op execute
```

- Refine the generic rules towards more specific rules
- Re-order the rules appropriately!
- Adjust the default behaviour accordingly! E.g. cmd-exec-default [permit]

Preventing HTML pattern in device names

- Allowing HTML pattern can be security vulnerability

```
nsoadmin@ncs% set devices device "<img\ src=\"not_real\"\
onerror=\"alert\(1\)\">"  address 2.2.2.2 device-type netconf ned-id
netconf
nsoadmin@ncs% commit dry-run
cli {
    local-node {
        data  devices {
             +    device "<img\ src=\"not_real\"\
onerror=\"alert\(1\)\">" {
             +        address 2.2.2.2;
             +        authgroup netsim;
             +        device-type {
             +            netconf {
             +                ned-id netconf;
             +            }
             +        }
             +    }
              }
    }
}
```

- NSO policy to prevent

```
nsoadmin@ncs% set policy rule nohtml expr
not(contains(/devices/device/name,'<')) error-message "No HTML
allowed for device name"
nsoadmin@ncs% commit
nsoadmin@ncs% set devices device <img address 2.2.2.2 device-type
netconf ned-id netconf
nsoadmin@ncs% set devices device <img authgroup netsim
nsoadmin@ncs% commit
Aborted: No HTML allowed for device name
```

Discussions

- Can this topic be better delivered in a lab session with hands-on practice?
- RESTCONF token authentication example
- Notification rule example
- *Automation of NACM rules (based on CSV input)
- Device authgroups and access control to devices layer, audit
- Accounting options: e.g. syslog, files, send to an application, etc.
- Limitations:
  - Using leafref can skip the NACM rule to access nodes. E.g.: restricting NACM rules on authgroups will not prevent user to see all while onboarding a device.
  - /devices/device/test/authgroup < is a leafref
  - path statement restrictions (must be node-instance-identifier; non-key leaf reference is not good!)

Complete your online session survey

- Please complete your session survey after each session. Your feedback is very important.
- Complete a minimum of 4 session surveys and the Overall Conference survey (starting on Thursday) to receive your Cisco Live t-shirt.
- All surveys can be taken in the Cisco Events Mobile App or by logging in to the Content Catalog on ciscolive.com/emea.

Cisco Live sessions will be available for viewing on demand after the event at ciscolive.com.

cisco Live!

Continue your education

Demos in the Cisco Showcase

Walk-In Labs

Meet the Engineer 1:1 meetings

Related sessions

Thank you

You make **possible**