2023 INTERNET2 TECHNOLOGY exchange

Ten Obscure Things I Learned Automating

Shannon Byrnes, Sr NetDevOps Engineer
Internet2 Infrastructure Systems and Software

Foreword

- ★ Most experiences come from a Cisco shop.
- ★ Some are obvious in hindsight. But that’s how hindsight works.
- ★ Many experiences rely entirely on memory.
- ★ Since they rely on memory and are from long ago, I don’t have as many pertinent visual aids as I’d like.

1. DoS Your TACACS Server

Responses to authentication requests slow down as the platform gets hit by so many threads at the same time.

Netmiko, on default settings, eventually does not tolerate the delay and will raise sporadic authentication issues.

<u>The worst:</u> Sporadic, unpredictable "Authentication Failed" errors.

This, but with multi-threading and many more devices.

```
Connected to device1.example.com
Connected to device2.example.com
Connected to device3.example.com
Connected to device4.example.com
Connected to device5.example.com
Connected to device6.example.com
Connected to device7.example.com
Connected to device8.example.com
Connected to device9.example.com
Connected to device10.example.com
```

>2000 Devices **+** 300 Netmiko threads
= TACACS server :-(

2. “show ver” and “show inv” are not always on the same page

In a Cisco shop, relying on “show version” can mislead you, particularly on old devices (of which universities have many)

WS-C3550-24PWR-SMI != **WS-C3550-24-PWR**

A real PID
Which *should* have been pulled from “show ver”

Not a real PID,
but is the one pulled from “show ver”

NAPALM

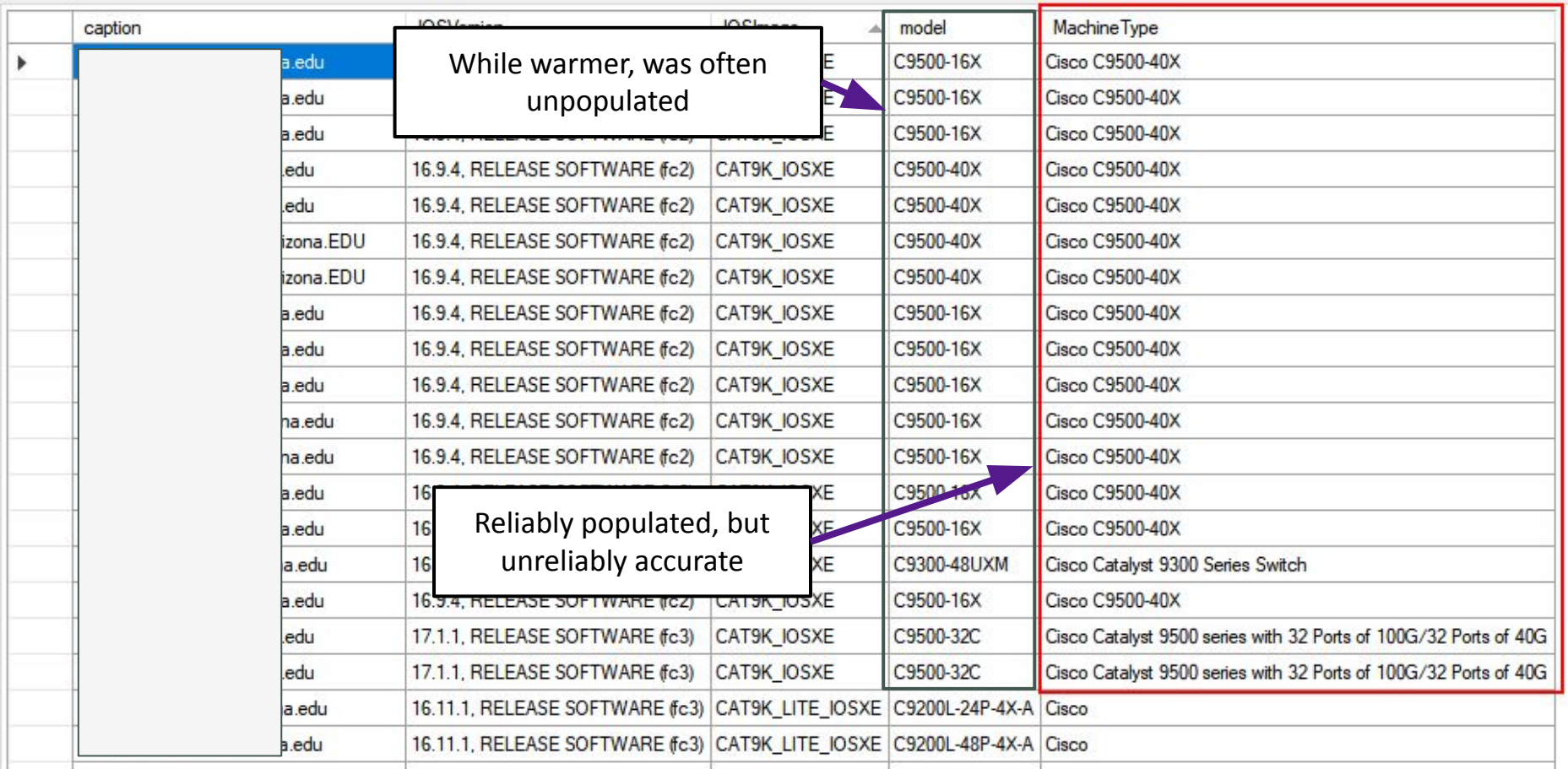
```
def get_facts(self):
    """Return a set of facts from the devices."""
    # default values.
    vendor = "Cisco"
    uptime = -1
    serial_number, fqdn, os_version, hostname, domain_name = ("Unkn

    # obtain output from device
    show_ver = self._send_command("show version")
    show_hosts = self._send_command("show hosts")
    show_ip_int_br = self._send_command("show ip interface brief")
```

Device

```
cis1#sh ver | i 3550
Cisco IOS Software, C3550 Software (C3550-IPBASEK9-M), Version 12.2(44)SE6, RELEASE SOFTWARE
(fc1)
ROM: Bootstrap program is C3550 boot loader
System image file is "flash:c3550-ipbasek9-mz.122-44.SE6.bin"
Cisco WS-C3550-24-PWR (PowerPC) processor (revision D0) with 65526K/8192K bytes of memory.
Model number: WS-C3550-24PWR-SMI
```

* Source: Wayback Machine Cisco 3350 Series Datasheet
* Source: Cisco PID Checker Tool

3. Monitoring software may be “best effort” at PID discovery



caption	IOSVersion	IOSImage	model	MachineType
a.edu		E	C9500-16X	Cisco C9500-40X
a.edu		E	C9500-16X	Cisco C9500-40X
a.edu		E	C9500-16X	Cisco C9500-40X
.edu	16.9.4, RELEASE SOFTWARE (fc2)	CAT9K_IOSXE	C9500-40X	Cisco C9500-40X
.edu	16.9.4, RELEASE SOFTWARE (fc2)	CAT9K_IOSXE	C9500-40X	Cisco C9500-40X
izona.EDU	16.9.4, RELEASE SOFTWARE (fc2)	CAT9K_IOSXE	C9500-40X	Cisco C9500-40X
izona.EDU	16.9.4, RELEASE SOFTWARE (fc2)	CAT9K_IOSXE	C9500-40X	Cisco C9500-40X
a.edu	16.9.4, RELEASE SOFTWARE (fc2)	CAT9K_IOSXE	C9500-16X	Cisco C9500-40X
a.edu	16.9.4, RELEASE SOFTWARE (fc2)	CAT9K_IOSXE	C9500-16X	Cisco C9500-40X
a.edu	16.9.4, RELEASE SOFTWARE (fc2)	CAT9K_IOSXE	C9500-16X	Cisco C9500-40X
na.edu	16.9.4, RELEASE SOFTWARE (fc2)	CAT9K_IOSXE	C9500-16X	Cisco C9500-40X
na.edu	16.9.4, RELEASE SOFTWARE (fc2)	CAT9K_IOSXE	C9500-16X	Cisco C9500-40X
a.edu	16	XE	C9500-16X	Cisco C9500-40X
a.edu	16	XE	C9500-16X	Cisco C9500-40X
a.edu	16	XE	C9300-48UXM	Cisco Catalyst 9300 Series Switch
a.edu	16.9.4, RELEASE SOFTWARE (fc2)	CAT9K_IOSXE	C9500-16X	Cisco C9500-40X
.edu	17.1.1, RELEASE SOFTWARE (fc3)	CAT9K_IOSXE	C9500-32C	Cisco Catalyst 9500 series with 32 Ports of 100G/32 Ports of 40G
.edu	17.1.1, RELEASE SOFTWARE (fc3)	CAT9K_IOSXE	C9500-32C	Cisco Catalyst 9500 series with 32 Ports of 100G/32 Ports of 40G
a.edu	16.11.1, RELEASE SOFTWARE (fc3)	CAT9K_LITE_IOSXE	C9200L-24P-4X-A	Cisco
a.edu	16.11.1, RELEASE SOFTWARE (fc3)	CAT9K_LITE_IOSXE	C9200L-48P-4X-A	Cisco

4. Sockets are closed! Come back tomorrow.

Attempting to sign into many contexts at once, on the same Cisco ASA hardware platform, can result in “Socket is closed” errors.

“Reliably” get authentication issues, but at a varying degree per run.

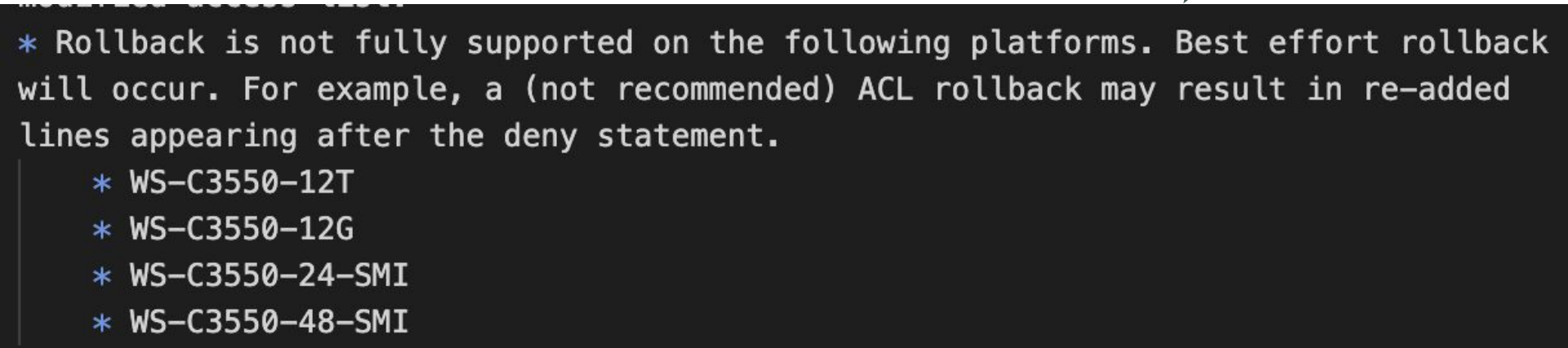
```
AWESOME-MPLS-FW/active/unit-2-1# show context
Context Name      Class                Interfaces           Mode         URL
*admin            default              Ethernet1/8          Routed       disk0:/admin.cfg
 ABC-MPLS         default              Port-channel30.422,  Routed       disk0:/ABC-MPLS.cfg
                                       1422
 DEF-MPLS            default              Port-channel30.129,  Transparent  disk0:/DEF-MPLS.cfg
                                       1129
 GHI-MPLS            default              Port-channel30.100,  Transparent  disk0:/GHI-MPLS.cfg
                                       1100
 JKL-MPLS            default              Port-channel30.113,  Transparent  disk0:/JKL-MPLS.cfg
                                       1113
 MNO-MPLS            default              Port-channel30.105,  Transparent  disk0:/MNO-MPLS.cfg
                                       1105
 PQR-MPLS            default              Port-channel30.114,  Transparent  disk0:/PQR-MPLS.cfg
                                       1114
 STU-MPLS            default              Port-channel30.104,  Transparent  disk0:/STU-MPLS.cfg
                                       1104
 VWX-MPLS            default              Port-channel30.116,  Transparent  disk0:/VWX-MPLS.cfg
```

5. Not All (Cisco) Rollbacks are Equal

In particular, ACLs may be “recreated” versus “re-applied” when rolling back.

At least on a few platforms where I could witness this first-hand, re-applied ACLs would be appended after the deny statement.

Shannon documentation



6. Juniper cRPD really, *really* wants to pass traffic

We gave a collaborative workshop at Community Exchange 2023, **“Get Started with Network Automation”**

Hosted a lab with two Cisco routers and one Juniper router, all virtual and containerized.

Could not shut interfaces as originally planned to demonstrate automated BGP config migration.

```
nodes:
  {%- set x = 2 %}
  cisco1:
    kind: cisco_xrd
    startup-config: startup-config/cisco1.conf
    {{- shared_node_settings(x) }}
  {%- set x = x+1 %}
  cisco2:
    kind: cisco_xrd
    startup-config: startup-config/cisco2.conf
    {{- shared_node_settings(x) }}
  {%- set x = x+1 %}
  juniper1:
    kind: juniper_crpd
    {{- shared_node_settings(x) }}
  {%- set x = x+1 %}
  ubuntu:
```

This guy.

```
[edit]
clab@juniper1# set interfaces eth1 ?
Possible completions:
+ apply-groups         Groups from which to inherit configuration data
+ apply-groups-except  Don't inherit configuration data from these groups
  description          Text description of interface
  mac                  Hardware MAC address
  mtu                  Maximum transmit packet size (1..16000)
  native-vlan-id       Virtual LAN identifier for untagged frames (0..4094)
> unit                 Logical interface
[edit]
```

Hello co-instructor friends! (if you’re in the audience)

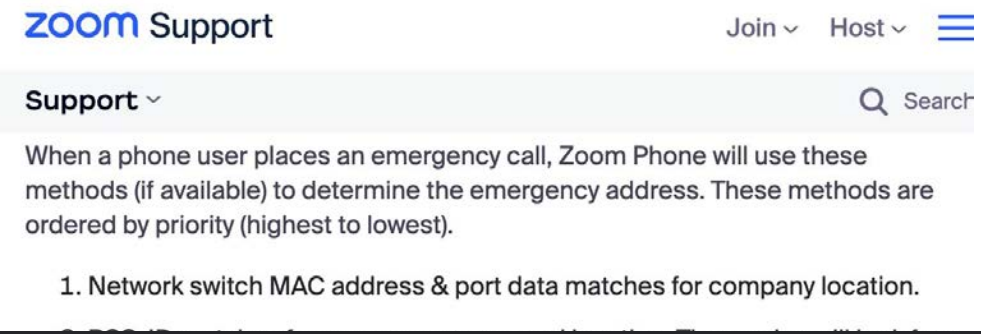
7. Don’t forget to close your automated tty sessions!

If using CLI-based automation, make sure to code so that your TTY sessions are always cleaned up, or you may hit your cap.

```
myswitch-north-cis10#sh users
    Line       User       Host(s)              Idle       Location
   0 con 0     slbyrnes idle                 00:02:09
   2 vty 0     slbyrnes idle                 00:00:22
                                               on-campus-112-46.net.coolu.edu
   3 vty 1     slbyrnes idle                 00:00:00
                                               on-campus-112-46.net.coolu.edu
*  4 vty 2     slbyrnes idle                 00:00:00 mybastion.net.coolu.edu
```

8. Which base MAC address reports via LLDP? Chassis + Switches

E911 platforms, such as Zoom nomadic emergency services, may depend on a database that ties three things together:



zoom Support

Join Host

Support Search

When a phone user places an emergency call, Zoom Phone will use these methods (if available) to determine the emergency address. These methods are ordered by priority (highest to lowest).

1. Network switch MAC address & port data matches for company location.

```
[redacted]#sh switch
Switch/Stack Mac Address : 5c5a.c77f.6880 - Local Mac Address
Mac persistency wait time: Indefinite
                                             H/W   Current
Switch#   Role    Mac Address     Priority Version  State
------------------------------------------------------------
*1       Active   5c5a.c77f.6880     15     V01     Ready
 2       Standby  78bc.1ab2.f100     1      V01     Ready
```

Potential base MAC locations:

- **`sh ver`**
  - “mac” (parsed)
  - “Base ethernet MAC Address:” (raw)
- **`sh module`**
- **`show chassis detail`**
  - It can be the burned-in chassis MAC, not either hypervisor, that appears via LLDP.
- **`sh switch detail`**
  - You need all stack members.
- **`sh spanning-tree bridge address`**

Bonus work: C4510R-E and C3550-12G required us to write custom TextFSM templates.

9. E911 Base MAC Address Fun: The AP Sequel

Rather that switch base MAC + switchport pairs, APs can be tied to physical addresses via BSSIDs.

A network management tool (**cough* Cisco Prime *cough**) may not necessarily pull and store this for you, but at least it can provide you a list of AP names to poke the controllers with (if you call against each AP individually).

However, a rate limit of 5 calls/sec makes for a very long-running job, and tweaking it isn't ideal when your WiFi engineers observe things slowing down at that default. So, nightly 3 hours it is.

```
with ThreadPoolExecutor(max_workers=5) as executor:
    fn = partial(
        get_accesspoint_bssids,
        controller_dict,
        accesspoints,
        wlc_creds["username"],
        wlc_creds["password"],
    )
    results = executor.map(fn, controller_dict, timeout=3600)
```

10. Parsing tables from WLC could have varying lengths

```
(Cisco Controller) >show ap wlan 802.11b AP01
Site Name..................................... MY_AP_GROUP1
Site Description.............................. MY_AP_GROUP1
WLAN ID           Interface           BSSID
-------           -----------        ---------------------------
1                 management           00:1c:0f:81:fc:20
2                 dynamic              00:1c:0f:81:fc:21
```

Ew.

```
for line in wlan_lines:
    row = line.split()
    mac = None
    if len(row) == 6:
        mac = format_mac(row[2].strip())
    elif len(row) == 5:
        mac = format_mac(row[1].strip()[-17:])
    elif len(row) == 3:
        mac = format_mac(row[2].strip())
    else:
        logger.error(f"{ap} Error: Unable to determ
```

Thank you!