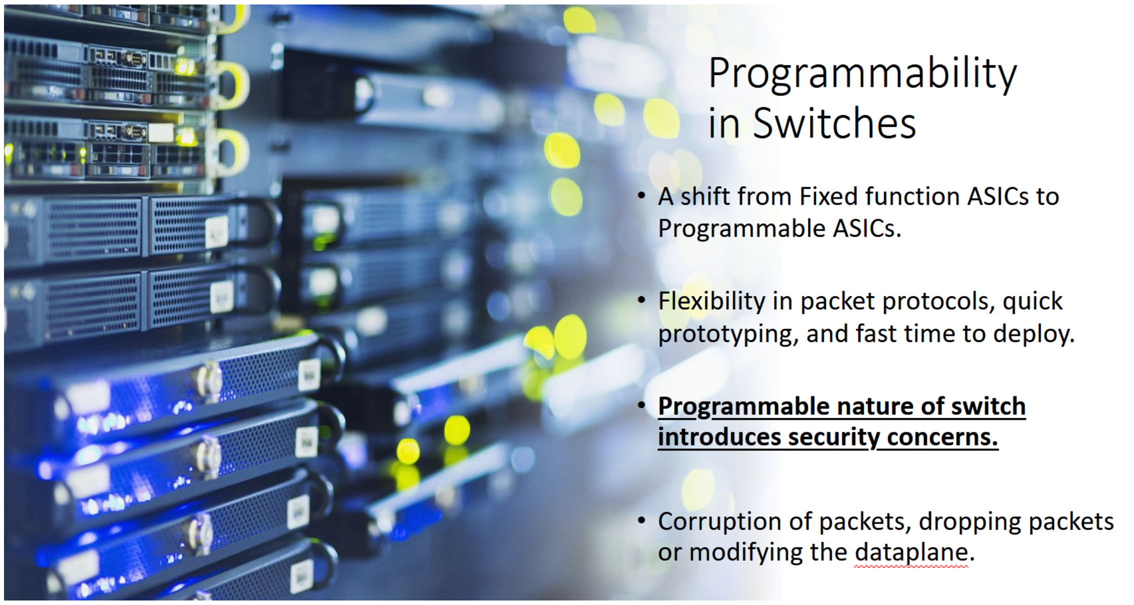
ra_switch

November 17, 2023

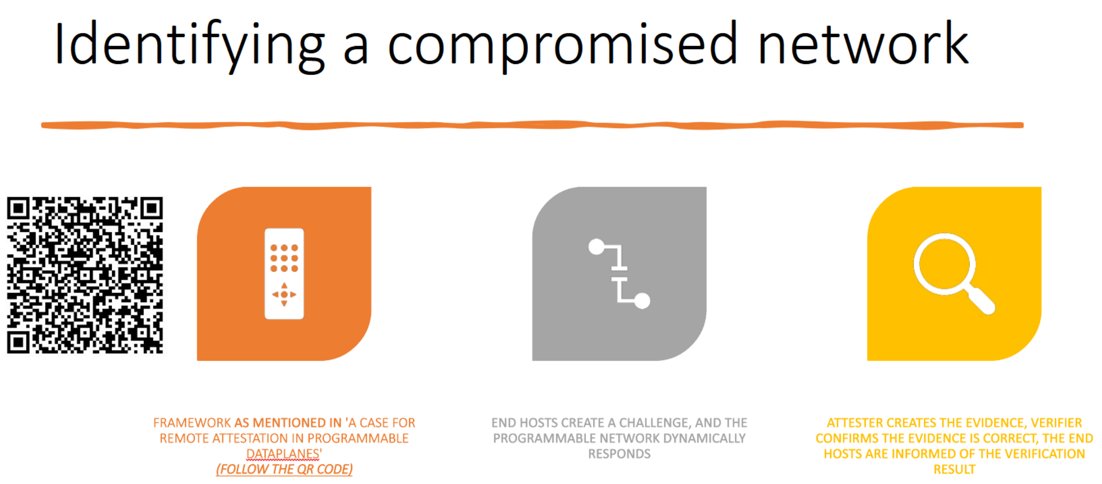
ATTESTATION CAPABLE PROGRAMMABLE SOFTWARE SWITCH

Alexander Wolosewicz, Nishanth Shyamkumar, Nik Sultana (ILLINOIS INSTITUTE OF TECHNOLOGY)



Programmability in Switches

- A shift from Fixed function ASICs to Programmable ASICs.
- Flexibility in packet protocols, quick prototyping, and fast time to deploy.
- **Programmable nature of switch introduces security concerns.**
- Corruption of packets, dropping packets or modifying the dataplane.



Identifying a compromised network

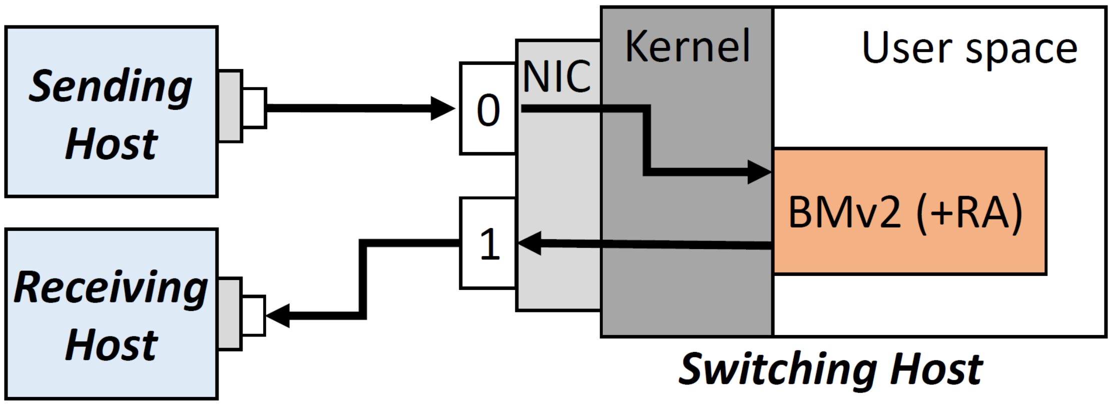
FRAMEWORK AS MENTIONED IN 'A CASE FOR REMOTE ATTESTATION IN PROGRAMMABLE DATAPLANES'
(FOLLOW THE QR CODE)

END HOSTS CREATE A CHALLENGE, AND THE PROGRAMMABLE NETWORK DYNAMICALLY RESPONDS

ATTESTER CREATES THE EVIDENCE, VERIFIER CONFIRMS THE EVIDENCE IS CORRECT, THE END HOSTS ARE INFORMED OF THE VERIFICATION RESULT

What you'll see in the demo

1. Setup resources in FABRIC and start the Switch.
2. Send packets from Sender to Receiver hosts via the Switch.
3. Show the Switch generating *evidence* based on *look-up table* contents ("rules").
4. Add and modify rules and display how the Attester's (i.e., Switch) evidence changes accordingly.
5. Show performance (throughput) of the prototype switch.



[1]:
```python
import json
import traceback
from fabrictestbed_extensions.fablib.fablib import fablib
```

Define constants

[2]:
```python
from ipaddress import ip_address, IPv4Address, IPv6Address, IPv4Network,␣
↪IPv6Network

# Slice
slice_name = 'BMv2_newesttopo'

# Switches
s1_name = "s1"

switch_cores = 8
```

```
switch_ram = 16
switch_disk = 100

# Hosts
h1_name = "h1"
h2_name = "h2"

h1_subnet=IPv6Network('fec0:db8:0:f000::/64')
h1_addr=IPv6Address('fec0:db8:0:f000::10')

h2_subnet=IPv6Network('fec0:db8:0:f001::/64')
h2_addr=IPv6Address('fec0:db8:0:f001::100')

host_cores = 2
host_ram = 8
host_disk = 10

net_h1_name = 'net_h1'
net_h2_name = 'net_h2'

net_h1_s1_name = 'net_h1_s1'
net_h2_s1_name = 'net_h2_s1'

# All node properties
image = 'default_ubuntu_20'
```

Setting up slice and submission

[ ]:
```
try:
    #Create Slice
    slice = fablib.new_slice(name=slice_name)

    # Add switch node s1
    s1 = slice.add_node(name=s1_name, image=image,
                        cores=switch_cores, ram=switch_ram, disk=switch_disk)
    s1.set_capacities(cores=switch_cores, ram=switch_ram, disk=switch_disk)
    s1_iface_h1 = s1.add_component(model='NIC_Basic', name="s1_local_nic1").
 ↪get_interfaces()[0]
    s1_iface_h2 = s1.add_component(model='NIC_Basic', name="s1_local_nic2").
 ↪get_interfaces()[0]

    # Add host node h1
    h1 = slice.add_node(name=h1_name, image=image,
                        cores=host_cores, ram=host_ram, disk=host_disk)
    h1_iface = h1.add_component(model='NIC_Basic', name="h1_nic").
 ↪get_interfaces()[0]
```

```
    # Add host node h2
    h2 = slice.add_node(name=h2_name, image=image,
                        cores=host_cores, ram=host_ram, disk=host_disk)
    h2_iface = h2.add_component(model='NIC_Basic', name="h2_nic").
 ↪get_interfaces()[0]

    #Add host networks
    host_net1 = slice.add_l2network(name=net_h1_s1_name,␣
 ↪interfaces=[s1_iface_h1, h1_iface])
    host_net2 = slice.add_l2network(name=net_h2_s1_name,␣
 ↪interfaces=[s1_iface_h2, h2_iface])

    #Submit Slice Request
    slice.submit()
except Exception as e:
    print(f"Error: {e}")
    traceback.print_exc()
```

```
[4]: config_threads = {}
```

Download libraries and tools from repository

```
[ ]: host_config_script = "sudo apt update -y -qq && sudo apt install -y␣
 ↪build-essential && sudo apt install -y net-tools && sudo apt install -y␣
 ↪iperf3"
switch_config_script = "sudo apt update -y -qq && sudo apt install -y␣
 ↪build-essential && sudo apt install -y net-tools && sudo apt install -y␣
 ↪python3-pip"
try:

    h1 = slice.get_node(name=h1_name)
    print(h1)
    if type(ip_address(h1.get_management_ip())) is IPv6Address:
        #h1.execute("sudo sed -i '/nameserver/d' /etc/resolv.conf && sudo sh -c␣
 ↪'echo nameserver 2a00:1098:2c::1 >> /etc/resolv.conf' && sudo sh -c 'echo␣
 ↪nameserver 2a01:4f8:c2c:123f::1 >> /etc/resolv.conf' && sudo sh -c 'echo␣
 ↪nameserver 2a00:1098:2b::1 >> /etc/resolv.conf'")
        h1.execute("sudo sh -c 'echo nameserver 2a00:1098:2c::1 >> /etc/resolv.
 ↪conf' && sudo sh -c 'echo nameserver 2a01:4f8:c2c:123f::1 >> /etc/resolv.
 ↪conf' && sudo sh -c 'echo nameserver 2a00:1098:2b::1 >> /etc/resolv.conf'")
    h1_os_iface = h1.get_interface(network_name=net_h1_s1_name)
    h1_os_iface.ip_addr_add(addr=h1_addr, subnet=h1_subnet)
    h1_config_thread = h1.execute_thread(host_config_script)
    config_threads[h1] = h1_config_thread

    h2 = slice.get_node(name=h2_name)
    if type(ip_address(h2.get_management_ip())) is IPv6Address:
        #h2.execute("sudo sed -i '/nameserver/d' /etc/resolv.conf && sudo sh -c␣
↪'echo nameserver 2a00:1098:2c::1 >> /etc/resolv.conf' && sudo sh -c 'echo␣
↪nameserver 2a01:4f8:c2c:123f::1 >> /etc/resolv.conf' && sudo sh -c 'echo␣
↪nameserver 2a00:1098:2b::1 >> /etc/resolv.conf'")
        h2.execute("sudo sh -c 'echo nameserver 2a00:1098:2c::1 >> /etc/resolv.
↪conf' && sudo sh -c 'echo nameserver 2a01:4f8:c2c:123f::1 >> /etc/resolv.
↪conf' && sudo sh -c 'echo nameserver 2a00:1098:2b::1 >> /etc/resolv.conf'")
    h2_os_iface = h2.get_interface(network_name=net_h2_s1_name)
    h2_os_iface.ip_addr_add(addr=h2_addr, subnet=h2_subnet)
    h2_config_thread = h2.execute_thread(host_config_script)
    config_threads[h2] = h2_config_thread

    s1 = slice.get_node(name=s1_name)

    if type(ip_address(s1.get_management_ip())) is IPv6Address:
        s1.execute("sudo sh -c 'echo nameserver 2a00:1098:2c::1 >> /etc/resolv.
↪conf' && sudo sh -c 'echo nameserver 2a01:4f8:c2c:123f::1 >> /etc/resolv.
↪conf' && sudo sh -c 'echo nameserver 2a00:1098:2b::1 >> /etc/resolv.conf'")

    s1_config_thread = s1.execute_thread(switch_config_script)
    config_threads[s1] = s1_config_thread

except Exception as e:
    print(f"Error: {e}")
    traceback.print_exc()
```

[ ]:
```
try:
    slice = fablib.get_slice(name=slice_name)
    for node in slice.get_nodes():
        print(f"{node.get_name()}: {node.get_ssh_command()}")
except Exception as e:
    print(f"Exception: {e}")
```

Copy scripts to the host VMs

[ ]:
```
h1_ssh = ""
h2_ssh = ""
s1_ssh = ""
h1_scp = ""
h2_scp = ""
s1_scp = ""
script_h1 = "ip6_bringup_h1.sh"
script_h2 = "ip6_bringup_h2.sh"
script_s1 = "switch_script.sh"
script_export_mac = "export_mac.sh"
script_addrule = "add_rules.sh"
```

```
out_file = "ascript.sh"

try:
    slice = fablib.get_slice(name=slice_name)
    h1_ssh = slice.get_node(name=h1_name).get_ssh_command()
    h2_ssh = slice.get_node(name=h2_name).get_ssh_command()
    s1_ssh = slice.get_node(name=s1_name).get_ssh_command()
    '''
    creating appropriate scp command for host h1
    '''
    h1_param = h1_ssh.split('ssh ')[1]
    h1_scp = "scp " + h1_param
    scp_addr = h1_scp.split('@')[1]
    scp_config = h1_scp.split('@')[0]
    scp_addr_brkted = "[" + scp_addr + "]"
    h1_scp = scp_config + "@" + scp_addr_brkted
    h1_scp = h1_scp.split('ubuntu')[0] + " " + script_h1 + " ubuntu" + h1_scp.
  ↪split('ubuntu')[1] + ":~"
    print(h1_scp)
   # h1.execute(h1_scp)
    '''
    creating appropriate scp command for host h2
    '''
    h2_param = h2_ssh.split('ssh ')[1]
    h2_scp = "scp " + h2_param
    scp_addr = h2_scp.split('@')[1]
    scp_config = h2_scp.split('@')[0]
    scp_addr_brkted = "[" + scp_addr + "]"
    h2_scp = scp_config + "@" + scp_addr_brkted
    h2_scp = h2_scp.split('ubuntu')[0] + " " + script_h2 + " ubuntu" + h2_scp.
  ↪split('ubuntu')[1] + ":~"
    print(h2_scp)

    '''
    creating appropriate scp command for switch s1
    '''
    s1_param = s1_ssh.split('ssh ')[1]
    s1_scp = "scp " + s1_param
    scp_addr = s1_scp.split('@')[1]
    scp_config = s1_scp.split('@')[0]
    scp_addr_brkted = "[" + scp_addr + "]"
    s1_scp = scp_config + "@" + scp_addr_brkted
    s1_scp = s1_scp.split('ubuntu')[0] + " " + script_s1 + " ubuntu" + s1_scp.
  ↪split('ubuntu')[1] + ":~"
    print(s1_scp)
    '''
    s1_param1 = s1_ssh.split('ssh ')[1]
    s1_scp1 = "scp " + s1_param1
    scp_addr = s1_scp1.split('@')[1]
    scp_config = s1_scp1.split('@')[0]
    scp_addr_brkted = "[" + scp_addr + "]"
    '''
    s1_scp1 = scp_config + "@" + scp_addr_brkted
    s1_scp1 = s1_scp1.split('ubuntu')[0] + " " + script_export_mac + " ubuntu"␣
↪+ s1_scp1.split('ubuntu')[1] + ":~"
    print(s1_scp1)

    s1_scp2 = scp_config + "@" + scp_addr_brkted
    s1_scp2 = s1_scp2.split('ubuntu')[0] + " " + script_addrule + " ubuntu" +␣
↪s1_scp2.split('ubuntu')[1] + ":~"
    print(s1_scp2)
    with open(out_file, "w") as out:
        out.write("#!/bin/bash" + "\n")
        out.write(h1_scp + "\n")
        out.write(h2_scp + "\n")
        out.write(s1_scp + "\n")
        out.write(s1_scp1 + "\n")
        out.write(s1_scp2 + "\n")
        out.close()
except Exception as e:
    print(f"Exception: {e}")
```

[ ]:
```
%%sh
chmod u+x ascript.sh
./ascript.sh
```

[ ]:
```
thrift_nnpy_script = "cd ~/behavioral-model/ci; chmod u+x install-nanomsg.sh␣
↪install-nnpy.sh install-thrift.sh; ./install-nanomsg.sh && ./install-nnpy.sh␣
↪&& ./install-thrift.sh"
bmv2_build = "cd behavioral-model/ ; ./autogen.sh && ./configure 'CXXFLAGS=-g␣
↪-O3' 'CFLAGS=-g -O3' --disable-logging-macros --disable-elogger && make -j2␣
↪&& sudo make install"
try:
    s1 = slice.get_node(name=s1_name)
    s1.execute_thread(thrift_nnpy_script)
    s1.execute_thread(bmv2_build)
except Exception as e:
    print(f'Exception: {e}')
```

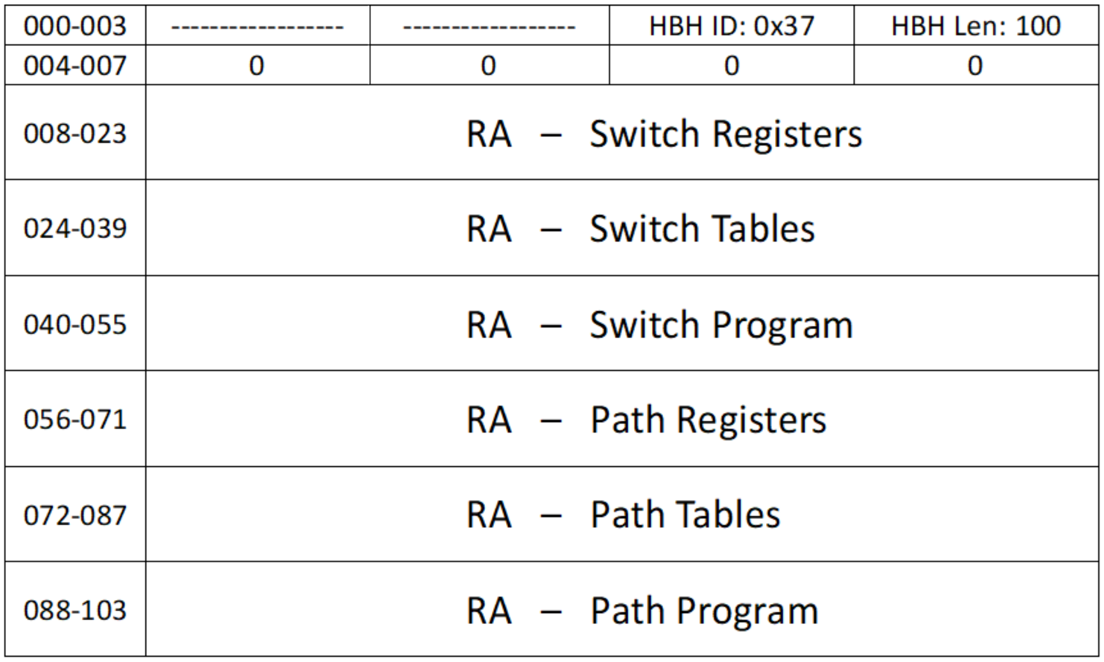
[39]:
```
ra_build = "cd ~/bmv2-remote-attestation/ ; ./autogen.sh && ./configure␣
↪'CXXFLAGS=-g -O3' 'CFLAGS=-g -O3' --disable-logging-macros --disable-elogger␣
↪&& make -j2"
```

```
ra_to_base = "cd ~/bmv2-remote-attestation/ ; mv /usr/share/p4c/p4include/
↪v1model.p4 /usr/share/p4c/p4include/v1model_orig.p4 && sudo ./ra_to_base.sh"
try:
    s1 = slice.get_node(name=s1_name)
    s1.execute_thread(ra_build)
    s1.execute_thread(ra_to_base)
except Exception as e:
    print(f'Exception: {e}')
```

Execute copied scripts

```
[ ]: script_h1 = "ip6_bringup_h1.sh"
script_h2 = "ip6_bringup_h2.sh"
script_s1 = "switch_script.sh"
script_export_mac = "export_mac.sh"

try:
    s1 = slice.get_node(name=s1_name)
    h1_s1_iface = s1.get_interface(network_name=net_h1_s1_name)
    h2_s1_iface = s1.get_interface(network_name=net_h2_s1_name)
    h1_s1_mac = h1_s1_iface.get_mac()
    h2_s1_mac = h2_s1_iface.get_mac()
    print(h1_s1_mac)
    print(h2_s1_mac)

 #   s1.execute("chmod u+x" + script_s1)
 #   s1.execute_thread("./" + script_s1)

    h1 = slice.get_node(name=h1_name)
    h1_iface = h1.get_interface(network_name=net_h1_s1_name)
    h1_ifname = h1_iface.get_device_name()
    h1_mac = h1_iface.get_mac()
    print(h1_ifname)

    h1.execute("chmod u+x " + script_h1)
    h1.execute_thread("./"+ script_h1 + " " + h1_ifname + " " + h1_s1_mac)

    h2 = slice.get_node(name=h2_name)
    h2_iface = h2.get_interface(network_name=net_h2_s1_name)
    h2_ifname = h2_iface.get_device_name()
    h2_mac = h2_iface.get_mac()
    print(h2_ifname)

    h2.execute("chmod u+x " + script_h2)
    h2.execute_thread("./"+ script_h2 + " " + h2_ifname + " " + h2_s1_mac)

    s1.execute("chmod u+x " + script_export_mac)
    s1.execute("./" + script_export_mac + " " + h1_mac + " " + h2_mac + " " +␣
↪h1_s1_mac + " " + h2_s1_mac )


except Exception as e:
    print(f'Exception: {e}')
```

HBH extension format



000-003	-------------------	-------------------	HBH ID: 0x37	HBH Len: 100
004-007	0	0	0	0
008-023	RA – Switch Registers			
024-039	RA – Switch Tables			
040-055	RA – Switch Program			
056-071	RA – Path Registers			
072-087	RA – Path Tables			
088-103	RA – Path Program			

[ ]:

Feel free to get in touch if you would like to know more or have any questions, email me at nshyamkumar@iit.edu

[ ]: