Very hard to even do a replay attack on a controller do to TCP sequence numbers (will just drop all previous flows) so if encrypted and can not break encryption, replay attack difficult.

(Good explanation here: <http://packetlife.net/blog/2010/jun/7/understanding-tcp-sequence-acknowledgment-numbers/> )

Maybe we focus solely on a switch that is compromised (i.e. you have a switch's private key and a controller's public key)

One secondary effect is we no longer need to mess with encryption since we are assuming we can correctly represent a single switch.

Our thesis: In an SDN environment, can a single, compromised switch severely degrade your entire network? (essentially SDN environments are more vulnerable than traditional environments because one compromised switch can have a much large effect)

Part 1: Research:

1.a. Make sure no one has tried to emulate a fake switch before. Someone else want to look at google Scholar (I am not so good at it). If we are good with this approach, I will email Jen and ask her.

1.b. Look into compromising switches (demonstrate switches currently have vulnerabilities that allow a signal switch to be compromised):

<http://www.infoworld.com/d/security/cisco-patches-vulnerabilities-in-some-security-appliances-switches-and-routers-228551>

<http://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20120711-ctms>

Any published docs on this? What about cold boot attack, vulnerabilities if get physical access to a switch - just need to show switches are vulnernable.

Part 2: Brainstorm ways to effect the controller:

Overload controller: What types of messages can the switch send to the controller to repeatedly to effect it?

Can the switch misrepresent traffic to get more data (i.e. send that mac address xxx belong to him when it does not)?

First --- Comments on this? Any issues? I think simplifying the project is good. We are honestly looking for a 2 page write up so this makes thing easier.

So what does our test set-up look like?

--- I brain stormed about this for a bit and decided the easiest thing to do was not to steal a TCP connection from an existing switch, but rather to make our own switch in python. I have wiresharked the TCP handshake between a controller and mininet (simply start your controller, start wireshark (filter = of), and then run sudo mn). I am including the wireshark of a single session in 4 formats - use whatever is easier for you. The entire handshake is very small (approx 10 packet sends from the switch). There then is a periodic liveness poll that we need to respond to. Each time I believe the packet contents are the same - I am hoping the only difference is the seq/ack numbers.

Processing forward:

Part 3: Fake a Switch

3.a: Record exact packets from switch to a file

     -- I wrote (found) sniffer.py to do this. I am having a time looking (making readable) the data contents. I write out packets from port=6633 to a file, but the file (packet records) is empty in some cases.

    -- Pycap might help with this: [http://pycap.sourceforge.net](http://pycap.sourceforge.net/)

3.b: Create simple socket to send messages to controller (done)

Part 4: Emulate the single mn topo=minimal switch:

a. Test replaying same packets as recorded with new Seq / ACKs

b. Determine new messages to send based on Part 2

          ---no progress on this

c. See how a single bad switch (no others) can affect a controller

Part 5: Develop a more realistic testing scenario.

---What should our topology look like

---Use iperf

Part 6. Test on multiple controllers(Ryu, Opendaylight,etc)

---Does the switch commands change for controllers?

---Does one do worse than others?

Where I am currently:

I created two files. One is called sniffer.py.