CS 6480: Advanced Computer Networks - Fall 2015

Lab Assignment 1:
Mininet - Software Defined Networking

August 23, 2015

Overview

There are three goals associated with this assignment. First is to introduce students to the Mininet network emulation environment [1, 2]. The second is to use the Mininet environment to explore the OpenFlow architecture and protocol [3, 4] as the de facto realization of a concept called Software Defined Networking (SDN). The third is to use the Mininet environment to replicate some results from a recently published SDN paper. Specifically, you will be required to implement an SDN controller capable of realizing a form of random host mutation as described in a recent HotSDN workshop paper [5].

Assignment Details

Scanning and probing statically allocated IP address ranges and IP addresses associated with specific hosts, is a common reconnaissance technique whereby would be attackers scope out the structure of a network and/or look for vulnerabilities on hosts. In the current Internet, communication with an endpoint requires that endpoint to have an IP address that is reachable via the Internet and as such will be "visible" and subject to scanning/probing.[1]

An interesting *Random Host Mutation* approach which uses SDN to mitigate this concern has recently been proposed [5]. In brief, this proposed solution uses a combination of DNS and SDN to allow the visible IP address of a host to change frequently so as to foil scanning/probing attempts. A host protected by this approach maintains a single real address all the time, but its hostname maps (via DNS) to different externally visible IP addresses. This dynamic mapping informs an SDN network (controller) to establish appropriate flows in an SDN network that front-ends the host in question. Critically, this approach allows different external hosts to **simultaneously** communicate with the protected host using different IP addresses.

For this assignment you will use the Mininet emulation environment to implement a simple version of the Random Host Mutation approach [5].

Figure 1 depicts a minimal Mininet setup and simplified workflow that would satisfy the requirements for this assignment. Specifically, at a minimum, your submitted assignment should:

- Provide an SDN controller (application) and a Mininet script which can be executed on an "All-in-one SDN App Development Starter VM" to show your solution.
- Have at least three hosts in the topology and show how two of those hosts (acting as "external Internet based hosts") simultaneously interact with the third host (the protected host) using the same hostname that at different times resolve to different IP addresses.
- Since our focus here is on exploring Mininet and SDN, you may use a simple "algorithm", e.g., changing periodically, to decide when to change the DNS mapping for the protected host.
- Your controller might have to implement additional functionality, e.g., ARP resolution, to make this work.

[1] In practice firewalls and other middle boxes can be used to mitigate this concern.

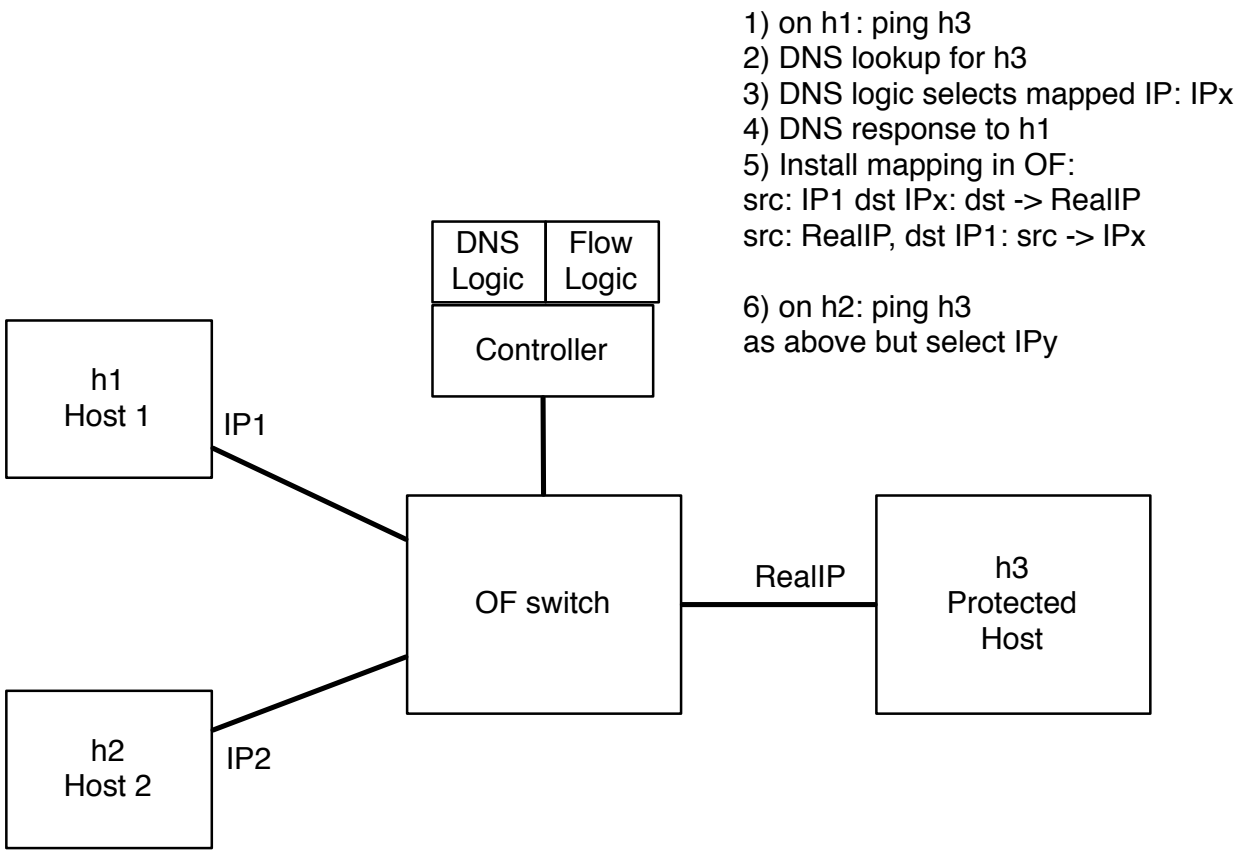

Figure 1: Minimal LA1 Mininet setup and workflow

Approach

This is a non-trivial assignment which will acquaint you with a variety of concepts, several of which might be new. It is strongly recommended that you start early and follow a staged approach to work through all the needed materials. Suggested approach:

- Get familiar with Mininet:
  - Read through the Mininet CoNEXT paper [2].
  - Download and install the "All-in-one SDN App Development Starter VM" by following the instructions on the SDN Hub page [6]. (Importing this appliance image into VirtualBox on your laptop is by far the easiest solution.)
  - Work through (not just read through) the "Mininet Walkthrough" [7] and the "Introduction to Mininet" [8]. Main objectives are to get a practical understanding of Mininet, its namespace isolation, how it fits into the SDN VM, how to communicate with the VM from your laptop, how to use Mininet commandline options, how to script Mininet commands, how to run an external controller (e.g., one running on your laptop), how to clean things up when something goes wrong, how to monitor both data plane and control plane interaction using Wireshark etc.

    Note that these tutorials assume a different Mininet VM image and an earlier version of OpenFlow etc. So there are a couple of places where specific examples in the tutorials don't work on the SDN Hub VM:

    * Use "openflow_v1" or "openflow_v4" in the Wireshark filter (not "of").
    * The builtin OVS controller assumed in this tutorial does not appear to use the loopback interface. To see, via Wireshark, an example of OpenFlow messages being exchanged, use the python remote controller as explained at the bottom of the tutorial.
    * You might have to build openflow to get a working version of the dpctl utility. In the openflow directory, run:

      ```
      ./configure
      make
      ```

- Get familiar with OpenFlow:
  - Read the OpenFlow whitepaper [3].
  - Read (browse) through a recent version of the OpenFlow Specification [4].
  - Work through the SDN Hub OpenFlow Tutorial [9]. (The SDN Hub tutorials seem to be fairly self contained, but you might also want to look at an earlier version of the OpenFlow Tutorial page [10].)
- Get familiar with your OpenFlow Controller options:
  - Work through the "POX Controller Tutorial" on SDN Hub [11].
  - Work through the "RYU Controller Tutorial" on SDN Hub [12].
- **Submit the first progress report.**
- Start executing the core of the assignment:
  - Read the random host mutation paper [5]. Pay particular attention to Sections 4 and 5.
  - Select and get comfortable with the controller you will want to use for implementing the assignment. This includes working through some of the example controller applications provided as part of the tutorials. (POX and RYU are popular controllers. RYU has excellent documentation and support for OpenFlow version 1.3: `http://osrg.github.io/ryu/`. POX has more extensive protocol library support. Specifically, support for DNS which will be needed for this assignment.)
  - Familiarize yourself with some of the example Mininet scripts (~/mininet/examples) which you will need to mimic in order to package your experiment.
  - Start working on the details of the final assignment. A reasonable approach might be to first deal with how the controller would set up the required flows, assuming that DNS has provided different mappings. Then figure out how to add a simple DNS resolver capable of realizing your remapping strategy. Then work on packaging everything up into a Mininet script which would demonstrate your implementation.
- **Submit the second progress report.**
- Finalize assignment:
  - Finish assignment and package it up for submission.
  - Submit assignment report.

Grading and evaluation

What to hand in

First Progress Report The progress report needs to be submitted via Canvas (on the due date stated in Canvas). The first progress report should briefly summarize what you have done up to that point to familiarize yourself with Mininet, Openflow and the OpenFlow Controller(s). The progress report submission should be in pdf and in addition to the progress summary might include screenshots and/or snippets of code to illustrate what you have done.

Second Progress Report The progress report needs to be submitted via Canvas (on the due date stated in Canvas). The second progress report should describe your planned approach for completing the assignment. E.g., what controller are you planning to use, how you plan to realize the IP to DNS mapping etc. This progress report submission should be in pdf and in addition to the progress summary might include screenshots and/or snippets of code to illustrate what you have done.

Completed Assignment Your completed assignment will consist of **two parts**: (i) An assignment report, submitted via Canvas, formatted as a short paper (including citations) and (ii) a tarball, submitted via `handin` on Cade, which contains your code and a script that can be run on an "All-in-one SDN App Development Starter VM" to reproduce your results.

Assignment report Your assignment report should be formatted as a "short paper" with citations. One to two pages should be adequate. Your report should include the following sections (see latex template):

- Introduction: Where you will motivate what problem you addressed and why that is important. **Here the problem being addressed is not to teach you about Mininet and/or SDN, but the security problem addressed by the Random Host Mutation approach [5].** Since we can not claim the idea, however, you'd have to reference that work in the introduction and say something to effect that you reproduced a part of the paper for this assignment.
- Design: Here you would briefly describe details of your approach *at the architectural level*. How did you do the DNS mapping? Did you use an external controller? How did you implement the DNS functionality?

  This would be a good section in which to provide a simple figure to explain what you have done.
- Implementation: Here you would briefly describe the actual implementation. For example, you could explain which controller platform you used, what language the implementation was in etc.
- Results: Here you will present some results to show that your implementation was functional. Figures, e.g., time series, distribution plots etc., and tables are in general good ways to present your results. Be sure to have accompanying text that describe each result that you present. Actual results will of course depend on the content of your report.
- Discussion: Here you would discuss the pros and cons of your approach. Specifically, you might want to point out limitations of the architectural approach in general (i.e., essentially what was proposed in [5]), as well as limitations of your own implementation. This would also be a good place to suggest "future work". I.e., how some of these limitations might be addressed etc.

Tarball Your tarball should contain a "README" file that explains how to run your submission on the "All-in-one SDN App Development Starter VM". For example, assuming you used POX as the basis for your controller, your instructions might be along the lines of:

```
1. In a terminal on Mininet VM, copy my SDN controller to pox/ext:
cp mycontroller.py ~/pox/ext
2. Run my controller:
cd ~/pox
./pox.py log.level --DEBUG mycontroller
3. In a second terminal on Mininet VM, clear Mininet and run
my mininet script:
sudo mn -c
sudo ./mymnscript.py
4. Results will be generated in ./results/
5. (Optional) Run script to regenerate results:
./processresults.py
6. Processed results are contained in ./results/final.txt
```

You might look at the student projects referenced in [2] for examples of "experiment repeatability instructions".

Very important: Your README file should also contain your name and uNID. It is not always possible to map your Cade username to your actual name.

Turning in assignments using handin on Cade Your submission tarball must be submitted on CADE machines using the handin command. To electronically submit files while logged in to a CADE machine, use:

`% handin cs6480 assignment_name name_of_tarball_file`

where `cs6480` is the name of the class account and `assignment_name` (la1, la2, or la3) is the name of the appropriate subdirectory in the handin directory. **Use la1 for this assignment.**

Grading

Criteria	Points
Progress reports	10
Assignment report	40
Code structure and inline documentation	10
Code executes as described and (re)generates results	40
Total	100

Note that to receive an A grade (90% or higher) for this assignment, the DNS mapping functionality need to be implemented. Without that, the highest grade possible will be a B+ (89%)

Notes

Random Mininet related notes

- Adding the host only network to your "All-in-one SDN App Development Starter VM" does simplify interaction with the Mininet environment. E.g., allowing you to run your controller on your laptop, while interacting with Mininet in the VM.
- DNS on Mininet: Mininet creates hosts by using network namespaces which allow each process (host in this context) to have it's own networking stack (ARP tables, routing tables, etc) and interfaces. However, all processes still share a host OS and a file system, so they all share the same resolv.conf file, and all use the same resolver methods for resolving names. (Note that when you do something like "h1 ping h2", Mininet internally maps that hostname to an IP address without using resolver methods.)

The simplest, if not the most elegant, way to temporarily "enable" DNS in Mininet is to add a name server directive to /etc/resolv.conf. E.g., adding the following:

```
nameserver 10.0.0.10
```

to /etc/resolv.conf would cause hosts in Mininet to attempt to connect to a DNS server at 10.0.0.10 in order to resolve names. Of course you'd have to run a DNS server at that address, or intercept the DNS requests in your controller to make it work. But this setup will cause Mininet hosts to attempt name resolution. **For grading purposes you can assume that the Mininet environment used for evaluation will be configured (as above) to expect a DNS server at 10.0.0.10.**

References

[1] Mininet Team, "Mininet - An Instant Virtual Network on your Laptop (or other PC)," http://mininet.org.

[2] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation." in *CoNEXT*. ACM, 2012.

[3] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling Innovation in Campus Networks," http://www.openflow.org/documents/openflow-wp-latest.pdf.

[4] Open Networking Foundation, "OpenFlow Switch Specifications," https://www.opennetworking.org/sdn-resources/onf-specifications/openflow.

[5] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: transparent moving target defense using software defined networking," in *Proceedings of the first workshop on Hot topics in software defined networks*, ser. HotSDN '12. New York, NY, USA: ACM, 2012, pp. 127–132. [Online]. Available: http://doi.acm.org/10.1145/2342441.2342467

[6] "All-in-one SDN App Development Starter VM," http://sdnhub.org/tutorials/sdn-tutorial-vm/.

[7] Mininet Team, "Mininet Walkthrough," http://mininet.org/walkthrough/.

[8] B. Lantz, N. Handigol, B. Heller, and V. Jeyakumar, "Introduction to Mininet," https://github.com/mininet/mininet/wiki/Introduction-to-Mininet.

[9] "OpenFlow version 1.3 tutorial," http://sdnhub.org/tutorials/openflow-1-3/.

[10] "OpenFlow Tutorial," http://archive.openflow.org/wk/index.php/OpenFlow_Tutorial.

[11] "POX Controller Tutorial," http://sdnhub.org/tutorials/pox/.

[12] "RYU Controller Tutorial," http://sdnhub.org/tutorials/ryu/.