Title goes in report.tex

Bonnie Eisenman, Michael Kranch and Anna Kornfeld Simpson

COS 597E Software Defined Networks
14 January 2014

Abstract

Here's some space for our abstract, which also is in report.tex ... compile this with make in the report directory. The makefile runs the latex compiler pdflatex 3 times in order to get the references and citations correct; if it says you have undefined references in the output at the end scroll up in the compiler output to find them!

1 Introduction

We probably want to introduce things, right now that's in report.tex. We might want to cite something in the introduction, we can do that using

```
{\cite{refname}
```

where the *refname* is defined in bibliography.tex in the

```
{\bibitem}
```

command. For example citing dpkt looks like this: [4].

2 Related Work

The related work might be long, so let's put it in a separate file, related.tex This is related.tex - I've already added all the links from the email thread to the bibliography, we can remove the ones we won't plan to use. TODO for Anna... look up the two security papers we read in class, add them to the bibliography and talk about them here.

3 Faking a Switch

I'm making up sections and order... there's a file called fake.tex where we can write about faking a switch. The verbatim environment is useful for preformatted text - there's an example in the introduction! Super awesome fake switch made in Python here's how we did it.

4 Disrupting an SDN with a Fake Switch

There's a file called attacks.tex for describing our various attacks; we could combine the results of the attacks we implemented here, or put them separately. This is attacks.tex and the below is copied from the google doc:

4.1 Dropping traffic

A compromised switch could simply drop packets sent to it, thus creating a service interruption. However, a controller would probably notice this quickly – it would appear as though the switch had failed, and automated recovery mechanisms would be initiated.

4.2 Cloning or diverting traffic

A compromised switch could ensure that traffic would be routed through an adversarys middlebox, or could clone traffic and send the cloned stream to the adversary. This would be more difficult for the controller to detect, unless it caused considerable slowdown. This attack could conceivably be used for purposes such as the NSAs MUSCULAR program, which inter-

cepts traffic as it flows between private data centers [5].

4.3 DoS other switches

By injecting traffic, a switch could deliberately attempt to overload neighboring switches TCAMs or otherwise DoS them.

4.4 DoS the controller / the control channel

By sending requests to the controller, the switch could attempt to overload it. One could fine-tune the control messages to maximize churn or CPU consumption in the controller.

However, our results show that a TCP congestion control successfully prevents a single switch from denying service to the controller; instead the extra packets from the switch are dropped.

4.5 DDoS the controller by spoofing many switches

What happens if in a network with n actual switches, the controller receives hello messages from n^2 switches, for example? This works around the throttling problem where a controller may simple rate-limit each switch.

4.6 Advertising false host attachments

A compromised switch can send the controller packet-in events with false packets, which falsely claim to have a particular MAC address attached to them. This could lead the controller to believe that the compromised switch should receive traffic intended for the host. While this would soon lead to packet loss and a noticeable disruption, it would nevertheless allow a switch to eavesdrop when it would not normally be in a position to do so.

4.7 Falsify measurement reports

A switch may return false results in response to a read-state measurement message, thus causing the controller to behave irrationally. For example, a switch could falsify or hide a DoS attack, elephant flows, etc.

4.8 Ignoring rules

A switch could simply ignore flow-table modification requests. For example, dropping packets will be noticed quickly; but a switch could allow packets to pass through that should have been dropped. Because of the difficulty of querying flow table state, the controller may not become aware of this. A compromised switch could thus operate in stealth mode and the inconsistent flow table might only be noticed once the switch allows a DoS attack to pass through, for example.

4.9 Modifying VLAN tags

4.10 Reporting flow mods to an adversary

An adversary could hope to learn about network traffic by observing flow mods. Certain flow mods might indicate that the controller has, for example, detected an intrusion, or that a specific host has connected to the network in a certain location. Thus, just knowing what flow mods are being issued by the controller could be a source of interesting information for an adversary.

4.11 Even more...

5 Conclusion

The conclusion might be short enough to be in report.tex - we can add other sections before it as necessary, and the bibliography (in bibliography.tex) will follow.

References

[1] Benton et. al. "OpenFlow Vulnerability Assessment" *ACM SIGCOMM Hot Topics in Software Defined Networking (HotSDN)*, 2013.

[2] Cisco. "Multiple Vulnerabilities in Cisco TelePresence Multipoint Switch" 11 July 2012 `tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20120711-ctms`

[3] Constantin, Lucian. "Cisco patches vulnerabilities in some security applications, switches, and routers." *InfoWorld* 10 October 2013, http://www.infoworld.com/d/security/cisco-patches-vulnerabilities-in-some-security-appliances-switches-and-routers-228551

[4] dugsong et. al. "dpkt" code.google.com/p/dpkt

[5] Gellman et. al. "How we know the NSA had access to internal Google and Yahoo cloud data" *Washington Post, The Switch*, 4 November 2013, http://www.washingtonpost.com/blogs/the-switch/wp/2013/11/04/how-we-know-the-nsa-had-access-to-internal-google-and-yahoo-cloud-data/

[6] Hecker, Artur. "Carrier SDN: Security and Resilience Requirements" 15 November 2013 http://www.ikr.uni-stuttgart.de/Content/itg/fg524/Meetings/2013-11-15-Muenchen/12_ITG524_Muenchen_Hecker.pdf

[7] "OpenFlow Packet Format." *OpenFlow*, archive.openflow.org/wk/images/c/c5/Openflow_packet_format.pdf

[8] Rowe, Mark. "Python Packet Capture and Injection Library" pycap.sourceforge.net

[9] Shin and Gu. "Attacking Software-Defined Networks: A First Feasibility Study" *ACM SIGCOMM HotSDN*, 2013.

[10] Stretch, Jeremy. "Understanding TCP Sequence and Acknowledgement Numbers" 7 June 2010, http://packetlife.net/blog/2010/jun/7/understanding-tcp-sequence-acknowledgment-numbers/