From IDS to IPS

Assignment due: 2/20/16

On Tuesday, you started an assignment that will, once completed, leave you with a virtual machine that can be deployed on a network and used to monitor that network for signs of intrusion. However, an intrusion defense system, by definition, does not include capabilities for actually preventing that intrusion from occurring.

Today, we'll be adding an intrusion prevention system, along with a firewall, to your growing collection of useful VMs. Make two more clones of your original Ubuntu VM (see the link provided in Blackboard (Assignments → Week 5) if you need help doing this on the lab computers), make sure they're updated (**sudo apt-get update && sudo apt-get dist-upgrade**) and get started!

This time things will be a bit more difficult: I'll be giving you a list of tasks to accomplish, but no direct instructions on how to accomplish those tasks. I've placed a number of resources in the "Additional Content" section of the Blackboard page (Course Content → Week 5) that provide enough information to complete the tasks. You are also free to employ any other methods you may be more familiar with to complete the tasks.

For this assignment, on one VM, you'll need to:

- Configure, build and install Suricata as an IPS from its source code
- Use ethtool to disable LRO and GRO on your virtual network interface (see section 8 of Snort setup guide PDF)
- Configure your Suricata installation to automatically download and install rules using Oinkmaster
- Configure Ubuntu to automatically update the rules daily using cron
- Configure Ubuntu to automatically start Suricata on boot using cron

Then, on your other VM, you'll need to:

- Create a script to be executed at startup that uses IPtables to:
    - Flush the current firewall rules from the system
    - Flush each chain of firewall rules individually (for good measure ☺)
    - Add a policy to:
        - Drop incoming packets
        - Accept outgoing packets
        - Drop forwarded packets
    - Append rules to the INPUT chain such that:
        - Traffic on the local network interface jumps to an "accept" state
        - Traffic from a related or already established connection jumps to an "accept" state
        - TCP traffic headed to port 22 should jump to an "accept" state
        - TCP traffic is logged
        - All other packets should be rejected with the error code "icmp-host-prohibited" sent back to the source of the rejected packet

It's important that you script performs these actions in this exact order; if we put the rule that rejects incoming packets at the front of the INPUT chain, none of our other rules would have a chance to be checked and we would have no inbound network traffic.

At this point, your firewall will allow SSH access to the system hosting the firewall and nothing else. In order for it to be useful, it would need to be configured to allow more traffic through. For example, if this firewall were to be deployed in order to protect a web server, we would want to create additional rules to allow HTTP, HTTPS, and DNS traffic. In order to actually deploy the IDS and IPS for use, you'd need to change the HOME_NET and EXTERNAL_NET variables to match the network you are protecting. Additionally, you'd need to change the "host-os-policy" section of suricata.yaml to reflect the hosts on the network. Furthermore, you'd need to setup all three VMs such that ingress traffic first hits the firewall, then the IPS, then the IDS, and finally the destination host.

To receive credit for this assignment, turn in a compressed archive containing your firewall script as well as the following screenshots:

For your IPS VM:

- Output of **sudo suricata --build-info | grep -i 'nfq'**
- Output of **ethtool -k eth0 | grep receive-offload**
- Output of **cat /etc/oinkmaster.conf | grep 'emerging'**
- Output of **cat /etc/crontab | grep -i 'oinkmaster'**
- Output of **cat /etc/crontab | grep -i 'suricata'**

For your firewall VM:

- Output of **stat -c "%a %n" ./firewall-script.sh**
  - Replace “firewall-script” with whatever you named your script
- Output of **cat /etc/crontab | grep -i 'firewall-script'**
  - Replace “firewall-script” with whatever you named your script