David Lambert

July 20, 2012

Assignment 6 - CNT4603, Langley

“Knock Before Entering: Securing SSHD”

The goal of this week’s project was to add some security to SecureSHell Server (SSHD). Since SSH is essentially a doorway into a system, one would want to protect is as if it were a door to their house. Of course, one of the best ways to secure a door is to lock it. On a hypothetically secure system, the only keyhole is on the inside of the door, but sometimes people still need to get in from the outside. One way to allow that access would be with a secret knock that only the user and the system know. Only after the user has knocked in the right way would the door be allowed to open. This scheme is roughly what was instituted in this week’s assignment: a daemon program was installed on the server machine which listens for a sequence of “knocks”, or pings, on pre-specified ports so that when the right sequence is executed, it allows SSH access through a different, pre-specified, non-default port.

According to the assignment’s webpage, I was to set up both the Debian machine and the CentOS virtual machine to require a secret “knock” before allowing SSH access to each of them. To get started, I decided to focus on setting up the CentOS machine as the SSH/knockd server first, and then it would hypothetically be easy to replicate the setup on the Debian machine by simply copying the configuration files over.

After verifying that *sshd* was installed on the CentOS machine, the first step was to change its listening port by modifying the appropriate line in */etc/ssh/sshd_config* to reflect my chosen port of **28910**.

The next step was to configure the firewall. I had worked with *iptables* before, and also in past assignment in this class, so this part wasn’t too difficult. I started with a clean slate by flushing the iptables rules and then used *iptables-save* and *iptables-restore* to write a new rule table from scratch. The assignment webpage instructed to restrict all ports except 80 (HTTP) and 443 (HTTPS).

After a fruitless attempt at executing *yum install knockd*, I discovered *knockd* was not in my default repositories on my CentOS machine. Fortunately, some diligent searching of Google yielded pleasant results: the appropriate RPM package was found at [*http://pkgs.org*]. After adding the associated repository, *yum* happily found the correct *knockd* package to install when I gave it the order this time.

A swift installation later, and it was time for configuration of *knockd*. With the webpage [*http://www.zeroflux.org/projects/knock*] for the creator of knockd conveniently hosting configuration instructions, I dove right into editing the *knockd.conf* file. The configuration was short and sweet - everything seemed pretty straightforward. I set the knock sequence to **4000,10000,5000,20000** and gave the instructions for iptables to allow unfiltered port **28910** traffic for twenty seconds if that knock sequence was received from any machine.

Now time for some testing. I wrote a short bash script, *testknock.sh*, on the Debian machine to execute the *knock* command with the appropriate knock sequence and then run *nmap* to test if the correct port opened and closed at the proper times. The script ran flawlessly, but the “knocks” were not getting to the CentOS machine for some reason. My first thought was “firewall issue” but some time with *Wireshark* told me that the packets were not getting blocked. Attempts at pinging the CentOS machine also went ordinarily. After much frustration, closer examination of the *Wireshark* live data capture and the *knockd.conf* file finally concluded that *knockd* was listening for packets only containing the “ACK” TCP flag - a misconfiguration by myself, set by the “tcpflags” parameter in the *knockd.conf* file. All it took was this simple correction, and now the “knocking” process works as it’s supposed to!

Thoroughness matters, so before checking the CentOS machine off the to-do list, I wanted to make sure I could actually connect to the SSH server. After trying to *ssh* to the

CentOS machine, a “cannot connect” error reinforced my premonition. A natural intuition to assume “firewall issue” persuaded me to consult “iptables -S”. After some consideration, it was obvious the error was that *knockd* was inserting the “open port 28910” rule *after* the rule to reject all other unspecified ports. It might has well have been talking to brick wall - or, a “firewall” (sorry, I couldn’t resist). A quick Google search showed me how to tell *knockd* to add the rule to the top of the list instead of the bottom (“-I INPUT 1...” instead of “-A INPUT...”, for *insert* instead of *append*).

A successful test followed by several more for redundancy confirmed that all my hard work and frustration had paid off. That should be the end of the learning curve for this project. The last polishing touch before tackling the Debian side was to make sure *knockd* was configured to run at startup via *chkconfig* - a nuance I had learned about Red Hat-derived distributions versus Debian-derived ones.

The Debian setup of *knockd* certainly went much quicker. I started out by installing *openssh-server* and *knockd* via *apt-get*. Fortunately they were already in the Debian repos. I then simply copied the *iptables* and *knockd.conf* files from the CentOS machine over to the Debian machine and made any adjustments to them as necessary.

Thinking about the requirement of the assignment for the setup to work after both machines are restarted, I questioned how *iptables* would know to be loaded during the startup process on the Debian machine (I had already dealt with *iptables* on the CentOS machine in the past and recalled that this did not necessarily happen automatically). Sure enough, a Google search confirmed that I had to create a script file to load the rule tables in */etc/network/if-pre-up.d/*.

Time to do some testing! An attempt to SSH to the Debian machine resulted in a “connection refused” error. *nmap* showed that port 28910 was “unfiltered” when it was supposed to be, so it was probably not a firewall problem. A few minutes of thought reminded me that I forgot to set the port that SSHD listens to on the Debian machine. An edit of */etc/ssh/sshd_config* promptly fixed this.

To wrap things up, I restarted both machines and ran my *testknock.sh* script file to confirm that everything worked as it was supposed to. Mission accomplished!

This was definitely an interesting and fruitful assignment. I had used SSH to connect to machines (even to my own personal machine) many times in the past and never really thought that "*Secure*" Shell had opportunities to be unsecure. It goes to prove that no matter how secure a system is ever developed, there will always be a way to compromise it - all in the game of hacker cat and mouse.