Exercise: Hacking Mr Robot - Defending with OSSEC

Steps

1. This is the second MrRobot exercise. If you haven't done the others yet, please use the attack path from the first to secure `root` access on this system. If you don't have time for that, use the Getting SSH Access on the MrRobot Virtual Machine instructions to do this.

   We'd like to break the `wfuzz` run. There's no reason for legitimate users to need to make quite so many authentication attempts.

   Here's the command we ran on Kali to guess usernames:

   ```
   wfuzz -c -z file,wordlist.txt --hs "Invalid username" -d "log=FUZZ&pwd=unlikelypass" ht
   ```

2. Let's configure OSSEC to put temporary firewall blocking rules in place when a Wordpress login rule fires, concerning the same source IP address, too many times in a specific time window.

   The MrRobot virtual machine has OSSEC already installed in its default location: `/var/ossec`

   First, let's tell OSSEC where to find Apache's `access_log`, which is in a strange location on this Bitnami system:

   ```
   /opt/bitnami/apache2/logs/access_log
   ```

3. Edit `/var/ossec/etc/ossec.conf`:

   ```
   cd /var/ossec/
   nano etc/ossec.conf
   ```

4. Place this near the end of the file, just before the `</ossec_config>` line:

   ```
   <localfile>
       <log_format>syslog</log_format>
       <location>/opt/bitnami/apache2/logs/access_log</location>
   </localfile>
   ```

5. Next, we would normally create a decoder in `/var/ossec/etc/decoder.xml`, but there's one already there called `web-accesslog`.

6. Take a look at the `web-accesslog` decoder.

   ```
   grep -A 7 \"web-accesslog\" etc/decoder.xml
   ```

7. We also don't even need to create a new rule. There's already one that detects access attempts on the Wordpress login:

   ```
   grep -A 5 id=\"31509\" rules/web_appsec_rules.xml
   ```

8. Next, we want to create a rule that fires when rule `31509` fires more than, say, 9 times in 5 minutes. There's already a rule that does this, but with a different threshhold. Let's tune it to our timing. Here's the rule before you make a change:

   ```
   <rule id="31510" level="8" frequency="6" timeframe="30">
       <if_matched_sid>31509</if_matched_sid>
       <same_source_ip />
       <description>CMS (WordPress or Joomla) brute force attempt.</description>
   </rule>
   ```

9. This rule (`31510`) triggers if rule `31509` fires eight (8) times in 30 minutes. That's not a typo. OSSEC counts frequencies thresholds strangely, whereby you have to add 2 to the listed frequency to understand the triggering frequency. For us to make this rule trigger when `31509` fires 10 times in 5 minutes, we'd need to write:

   ```
   <rule id="31510" level="8" frequency="8" timeframe="5">
   ```

10. Edit rule `31510`:

    ```
    nano rules/web_appsec_rules.xml
    ```

11. Make the rule read like so:

    ```
    <rule id="31510" level="8" frequency="8" timeframe="5">
        <if_matched_sid>31509</if_matched_sid>
        <same_source_ip />
        <description>CMS (WordPress or Joomla) brute force attempt.</description>
    </rule>
    ```

12. Now, let's configure OSSEC to put a firewall rule in place when rule 31510 triggers.

    We'll edit `/var/ossec/etc/ossec.conf` – find the `<active-response>` section that includes a `firewall-drop` command:

    ```
    <active-response>
        <!-- Firewall Drop response. Block the IP for
        - 600 seconds on the firewall (iptables,
        - ipfilter, etc).
        -->
        <command>firewall-drop</command>
        <location>local</location>
    ```

```
    <level>6</level>
    <timeout>600</timeout>
</active-response>
```

13. We're going to replace the `<level>` line, so that we trigger on specific rules, rather than rule severities, with escalating lockout periods.

14. Edit the `/var/ossec/etc/ossec.conf` file:

    ```
    nano etc/ossec.conf
    ```

15. Change the block to read like so (adding the `<rules_id>` and `<repeated_offenders>` fields):

    ```
    <active-response>
        <!-- Firewall Drop response. Block the IP for
        - 600 seconds on the firewall (iptables,
        - ipfilter, etc).
        -->
        <command>firewall-drop</command>
        <location>local</location>
        <rules_id>31510</rules_id>
        <timeout>600</timeout>
        <repeated_offenders>600,3600,86400</repeated_offenders>
    </active-response>
    ```

16. Let's put this in place by starting up OSSEC:

    ```
    /var/ossec/bin/ossec-control stop
    /var/ossec/bin/ossec-control start
    ```

17. Now, on the Kali terminal, try running that `wfuzz` one more time:

    ```
    wfuzz -c -z file,wordlist.txt --hs "Invalid username" -d "log=FUZZ&pwd=unlikelypass" ht
    ```

18. Your attack should get through 10 attempts, but then stall. OSSEC has inserted an `iptables` rule that will block all traffic from your attacking IP address for 10 minutes.

19. You'll notice that any SSH sessions you have to the MrRobot virtual machine appear to be hung. OSSEC's iptables rule is blocking the SSH sessions from your Kali system.

20. Optionally, if you'd like to see the rules that are doing this block, log into the MrRobot system on the `virt-manager`'s MrRobot console, using the username `robot` and the password `abcdefghijklmnopqrstuvwxyz`.

21. Use `sudo` to get `root` access:

    ```
    sudo su -
    ```

22. Observe the `iptables` rules and ossec's reason for implementing them using these commands:

```
iptables-save  | grep INPUT
cat /var/ossec/logs/active-responses.log
cat /var/ossec/logs/alerts/alerts.log  | grep -A3 -B2 -E '31509|31510'
```

23. On your Kali system, please kill your netcat listener so it doesn't interfere with future exercises.

    ```
    pid=`ps -ef | grep "n[c] 10" | awk '{print $2}'`
    kill $pid
    ```

24. Change your Slack status to `:thumbsup:`.

25. Suspend the virtual machines:

    ```
    sudo /scripts/suspend-all-vms.sh
    ```