



Linux Day 2

Services, Backdoors, and How to
Protect



Processes and Services

Definitions

- Processes
 - runs background or foreground on system
 - occurs as a result of a command/script being run
 - has a PID (process ID)
- Services
 - background
 - consists of 1+ processes
 - serves the user some sort of function
 - ssh, ftp, samba, etc.
- Daemons
 - Background
 - non-terminal

Processes Specified

- **ps -aux | less**
 - -a: info about all users
 - -u: additional information
 - -f: full information
 - -x: shows info about processes w/o terminals
- **kill -9 [PID]**
 - kills the specified process completely (force quits the process)

Relevance of Processes

- Processes- don't give you points necessarily in CyPat but they can lead you to discover more vulnerabilities happening on the system
- useful in real-life situations because you can see what is happening on the background

Services Specifically

- **service --status-all**
 - checks all running services and their current settings
 - [+] = active
 - [?] = masked
 - [-] = off
- Other helpful commands:
 - sudo [service_name] restart
 - sudo [service_name] stop
 - sudo [service_name] status

Service Hunting

- in CyberPatriot you should also find services that are malicious or are not specified in the README (for instance if they ask you to host openssh only then you can delete apache2 and stuff)
- don't delete critical services and everything should be all good :)

Backdoors



Backdoors



sneaky bois





How to prevent backdoors?

Just let them use the front door (duh)





What are backdoors?

Backdoors: a point of access where users (authorized or not) gain access to root privileges on a system, usually to exploit it

- can be used for good/bad stuff (either companies or hackers use)
- many instances are opened by bad peeps

THEY MUST STAY IN (persistence is key)

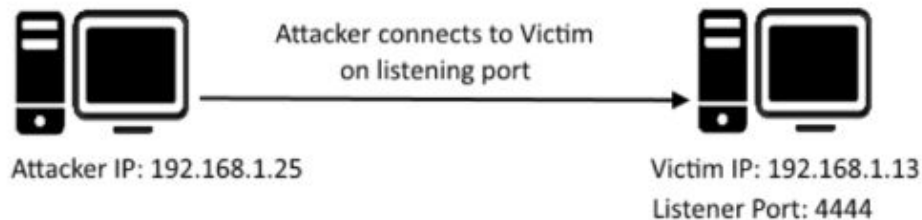


Types of Backdoors

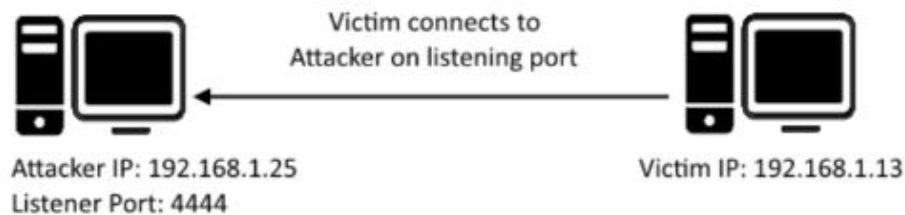
- **bind shell:** the attacker targets the user (victim) by listening on an open port and connecting to it (their shell)
 - THEY connect to YOU
 - inbound traffic
- **reverse shell:** the attacker opens up a listener port while the user connects to said port (access to the shell)
 - YOU connect to THEM
 - outbound traffic
- **insecure remote services:** pretty much just hackers exploiting services (SSH, Samba, Telnet) to connect onto device

QUIZ

bind shell/shell bind



reverse shell



How to Actually Remove/Detect Backdoors

```
user@ubuntu:~$ sudo netstat -tulpen
[sudo] password for user:
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
User      Inode          PID/Program name
tcp        0      0 0.0.0.0:1234        0.0.0.0:*               LISTEN
0          51041          23288/nc
tcp        0      0 0.0.0.0:1234        0.0.0.0:*               LISTEN
0          26698          2229/nc
tcp        0      0 0.0.0.0:1234        0.0.0.0:*               LISTEN
0          21477          1732/nc
tcp        0      0 0.0.0.0:1234        0.0.0.0:*               LISTEN
0          18753          1222/nc
tcp        0      0 0.0.0.0:1234        0.0.0.0:*               LISTEN
0          17631          847/dnsmasq
```

1. netstat
 - a. prints network connections, routing tables, bunch of other stuff
 - b. **netstat -tulpn**: displays a bunch of information about port connections and processes
 - i. -t = tcp
 - ii. -u = udp
 - iii. -l = listening ports
 - iv. -p = port
 - v. -n = numerical address

How to Actually Remove/Detect Backdoors, Part 2

1. Track down the processes that are using the ports using **lsof** or **fuser** or **ps aux**
 - a. **lsof**- checks all open files and the processes that are running them
 - b. **fuser**- displays PIDs of processes using the files
 - c. remember **ps -aux**?
2. kill said processes after (**kill -9 <pid>**) and delete files attached to them




Crontabs

- located in /etc/crontab and /var/spool
 - task schedulers that schedule events to happen at a certain time/moment
- * = stands for “every”
 - ***** **[backdoor]**
 - each place stands for a time (every minute of every hour of every day of every month of every year, forever)





System Crontabs vs. User Crontabs

- System Crontabs:
 - /etc/crontab
 - /etc/cron.d/
 - /etc/cron.daily
 - /etc/cron.monthly
 - /etc/cron.weekly
 - /etc/cron.hourly
 - User Crontabs:
 - must be root (command?!?)
 - check the /var/spool/cron directory
 - empty = good
- 



rc.local

- /etc/rc.local
 - script that is run after other system services are running
 - Linux formerly ran on runlevels to determine the state of the system but has been deprecated
 - check to make sure it is empty so that nothing else is running besides what should be there (**exit 0**)



UFW

(Uncomplicated Firewall)

- Firewalls are important in filtering network traffic
 - You want to block things that harm your computer from coming in. UFW is the tool to do that.
- There is also gufw, which uses the GUI, and that appears in your system settings for configuration. But it's better to use ufw.

TO PREPARE:

```
sudo apt-get install ufw
```

More commands on the next slide.

UFW commands

- `ufw enable`
 - Enables firewall and enables firewall on boot
- `ufw disable`
 - Disable firewall and disable firewall on boot
- `ufw default allow|deny|reject [incoming|outgoing|routed]`
 - Sets the default for all incoming, outgoing, or routed packets (depending on option) to allow, deny, or reject (depending on the option)

UFW commands, continued

- `ufw reject [args]`
 - Reject any packet following these arguments (We will go over args in next slide)
- `ufw allow [args]`
 - Allow any packet following these arguments (We will go over args in next slide)
- `ufw deny [args]`
 - Deny any packet following these arguments (We will go over args in next slide)
- `ufw logging [on|off]`
 - Enables logging,
 - logs can be found at `/var/log/ufw.log`

Arguments in UFW commands

Example arguments

- 24/tcp
 - This is a port number and a protocol
- sshd
 - A service name
- apache2/udp
 - A service name and a protocol
- Deny means do not allow that package to travel in the specified direction (incoming,outcoming). Default direction is incoming
- Reject means do not allow that package to travel in specified direction and inform the sender that the packet was rejected
- Allow means do allow that packet to travel.



Important commands to use

- `sudo ufw enable`
- `sudo ufw reset`
- `sudo ufw default deny incoming`
- `sudo ufw default allow outgoing`
- `sudo ufw allow [service you need, like ssh]`





Important Configurations We Will Be Going Over

- SSH
- FTP
- Samba





SSH

- SSH: Secure Shell
 - Provides remote terminal connection to other people
- Config file: `/etc/ssh/sshd_config`



Important SSH Settings

- Protocol 2
- PermitEmptyPasswords no
 - Why would we disable this?
- PermitRootLogin no
 - Why would we disable root login?
- PasswordAuthentication no
 - Yes, I said no. Does anyone know why?
- RSAAuthentication yes
- PubkeyAuthentication yes
- X11Forwarding no



When you are
done configuring...

`sudo service sshd restart`





FTP

- File Transfer Protocol
 - Allows you to move files between computers
- FTP is not secure
 - Look up when it was created, and tell me why it isn't.
- To meet these vulnerabilities, vsftpd (very secure ftp daemon) and PureFTPD were created



Anonymous vs. Non-Anonymous Servers

- Some servers are built so anyone can use them
 - [Mozilla.firefox.org](https://mozilla.org) is an example
- Some are built for private usage
 - Companies have files on their servers that are only meant for employees
- You can access anonymous servers by providing your username and password as anonymous
- You have to access non-anonymous servers as a local user



Some Basic Steps Before We Configure

- Install the required FTP software
 - `sudo apt install vsftpd`
 - `sudo apt install pure-ftpd`
- Allow FTP through the firewall
 - `sudo ufw allow ftp`
- Create the appropriate local users



Vsftpd configuration

- Open /etc/vsftpd.conf
- anonymous_enable=NO
 - Given what you know about anonymous and non-anonymous, why is this setting important?
- chroot_local_user=YES
 - We'll go into chroot on the next slide
- local_enable=YES
- write_enable=YES

`sudo service vsftpd restart`

More About Chroot

- Go to <ftp.mozilla.org>
 - It says “Index of /”
 - Why would they give us access to the / directory? That seems insecure.
 - This is a chroot jail
- Chroot stands for change root
 - You make the folder the user can access appear to be the root directory, even if it's not
 - The users are then confined to that folder, because root is the highest directory, and you can't go out of it
 - You want users on an FTP server to only access the appropriate files, so a chroot directory stops them from accessing everything on the server

Pure-ftp configuration

- vsftpd is more secure, but CyberPatriot might require this
- It is harder to configure, because there isn't one file. It's a whole bunch, each with its own configuration.
- TLS: degree of SSL enforcement (0-2)
- NoAnonymous: force private
- AnonymousOnly: force public
- UnixAuthentication: unencrypted authentication
- PamAuthentication: use PAM, uses /etc/pam.d/pure-ftp
- ChrootEveryone: chroot everyone to their homes

systemctl restart pure-ftp

Samba

- Unix apps that provide secure, stable file and print services
 - Mostly for file sharing
- Follows SMB protocol (Server Message Block)
- Useful for sharing between Windows and Linux
- Runs on port 445

Doing Samba:

- Install it
- Make a share folder on the server.
- Configure `/etc/samba/smb.conf`
- Setup user accounts w/ passwords
- Access it



Steps 1 and 2

- `sudo apt-get install samba`
- `cd /mnt`
- `mkdir sambashare`
- Add files in there

You have now installed samba and set up the directory on the server.



Basics of /etc/samba/smb.conf

- Go to /etc/samba/smb.conf
- It has a lot of parts: homes, printers, global, shares, personalized share configs
 - [global] configures behavior of server in general
 - [printers] is for the behavior of printers
 - [shares] configures file shares
 - [homes] and personalized shares are the same as [shares], but more specific.

Basics, continued

- 3 types of shares:
 - Global: what happens when others access you
 - Private: not accessible unless you have been given access
 - Public: shares that people can specifically request. Public shares usually make the most sense.

Adding smbashare to the file

- At the end of the file, add a section for sambashare using [sambashare]

Add the following lines

- comment = Samba on Ubuntu
- path = /mnt/smbashare
- read only = no
- browseable = yes

Now restart the smdb service



Testing It

To test if smb.conf has errors, use the command testparm

- If it returns nothing, you're good! It works.



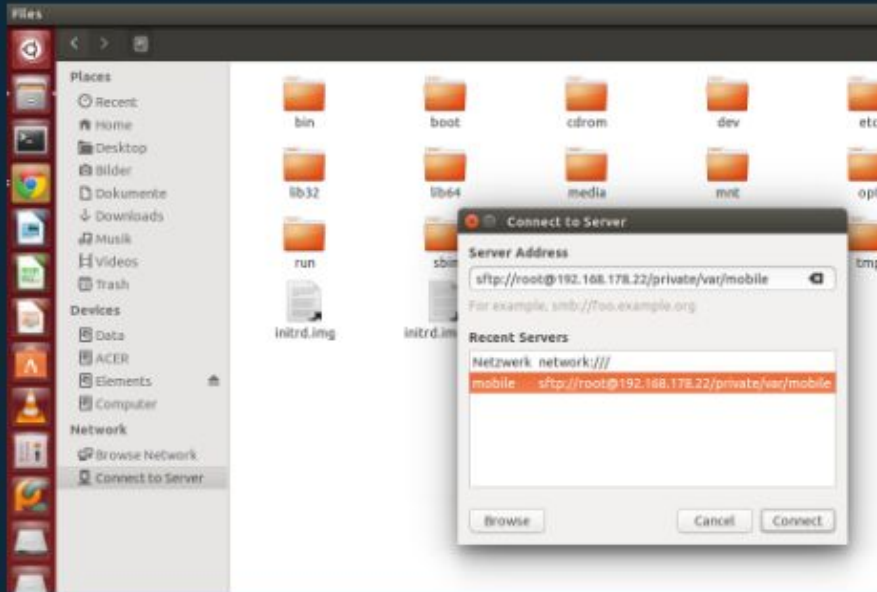


Connecting to a Share

- Add a user for samba
 - `sudo adduser sambauser`
 - Make sure to give a password
- Add the user to samba
 - `sudo smbpasswd -a sambauser`
 - -a means add the user
 - -x removes the user



How to Connect



- Use the ifconfig command to look for your IP Address
 - It should say 127.0.0.1

Cool. That's done. Now:

- Run the command nautilus (file explorer)
- Click on connect the server, in the bar on the left
- In the "Server Address" bar put `smb://sambouser@127.0.0.1/sambashare`
- Put in your password, and now you can connect and share files!



Other Samba Configs

write ok - editable files/directory (writeable)


read only - can only read files

valid users - list of users ok to login (or a @group)

guest ok - login password not needed (public vs private share)

browseable - can explore other directories in share

copy/template - copy configs from a template share






Things to look out for in Samba

Watch out for things that seem suspicious.
For example:

`[sharing]`

`path = /`

Now anyone on the share can view anything
on your system. That's bad.



Tips for Samba

1. Look at the shares and make sure to watch out for anything suspicious.
2. Delete unnecessary shares.
3. Be careful on what is commented. In Samba, both semicolons and # signs start comments.