NOTE/DISCLAIMER:

These plans work great for locking it down all the way. Nothing got in or out, 0% pwnage here...but you need to actually keep services usable.

This plan/Fail2ban locks everything down, right away. Best not to get pwned, so deploy it (or a something better), and use it as an opportunity to get your services back up safely.

CHECKLIST - FIRST FIFTEEN MINUTES* *consider taking a snapshot first...

1. Log into Debian box from local machine

   ```
   localbox:~admin$ ssh root@172.20.241.39
   ```

2. Debian might not have sudo- install sudo, Snort, and Fail2Ban

   ```
   root@server:~#  apt-get update ; apt-get install sudo snort fail2ban
   root@server:~#  apt-get --only-upgrade install exim4 -OR- ??rmaildaemon
   ```

3. Create a new user, and give that user sudo privileges

   ```
   root@server:~#  adduser newUserName
   root@server:~#  adduser newUserName sudo -OR- sudo adduser foo sudo
   ```

4. Edit sudoers file; make sure newly created account is present

   ```
   root@server:~#  sudo nano /etc/sudoers
   ```

   Look for this part of the file, `newUserName` should match `root`

   ```
   # User privilege specification
   root        ALL=(ALL:ALL) ALL
   newUserName ALL=(ALL:ALL) ALL
   ```

   Make sure this line is in place

   ```
   # Allow members of group sudo to execute any command
   %sudo   ALL=(ALL:ALL) ALL
   ```

   Be sure to remove this line, in order to use sudo and be prompted for a password

   ```
   NOPASSWD:ALL
   ```

5. Secure Root, Lock SSL Root Access, Change SSH Port, and Restart SSH

   ```
   root@server:~#  sudo passwd -l root
   root@server:~#  nano /etc/ssh/sshd_config
   ```

   Uncomment these lines

   ```
   #PermitRootLogin no
   #22 (
   ```
   change the port, to one you'll remember, then you can access it later via ssh user@host -p (port whatever)

6. Add an AllowUsers line at the bottom of the file with a space separating usernames, save file/exit

   ```
   AllowUsers newUserName
   ```

   Restart SSH

   ```
   root@server:~#  service ssh restart -OR- /etc/init.d/ssh restart
   ```

7. Configure & restart Fail2Ban

   ```
   root@server:~#  nano /etc/fail2ban/jail.local
   root@server:~#  /etc/init.d/fail2ban restart -OR- service fail2ban restart
   ```

8. Log out and log back in with newly created account

   ```
   root@server:~#  exit
   localbox:~admin$ ssh newUserName@172.20.241.39
   ```

9. Update system *****This will likely have to wait, due to time constraints*****

```
root@server:~#  apt-get update && apt-get upgrade -y
```

Debian (will need to revise once we find out exact version)

root:P@ssw0rd and newuser:P@ssw0rd

1. Change root password: passwd
2. Fail2ban: apt-get install fail2ban -y
   Service fail2ban start
3. Update (may need to wait if firewall is blocking first 15 min): apt-get update
4. Lock ssh: vi /etc/ssh/sshd_config
   PermitRootLogin no
   PasswordAuthentication no
   Service ssh restart
5. Set up IP tables (Sam's CNIT 129 lab, may want to create separate doc for this)
   iptables -N TCP
   iptables -N UDP
   iptables -P FORWARD DROP
   iptables -P OUTPUT ACCEPT
   iptables -P INPUT DROP
   iptables -A INPUT -m conntrack --ctstate RELATED, ESTABLISHED –j ACCEPT
   iptables -A INPUT -I lo -j ACCEPT
   iptables -A INPUT -m conntrack --ctstate INVALID -j DROP
   iptables -A INPUT pp icmp --icmp-type 8 -m conntrack --ctstate NEW -j ACCEPT
   iptables -A INPUT -p udp --m contract --ctstate NEW –j UDP
   iptables -A INPUT -p tcp --syn -m conntrack --ctstate NEW -j TCP
   iptables -A INPUT -p udp --j REJECT --reject-with icmp-port-unreachable
   iptables -A INPUT -p tcp --j REJECT –reject-with tcp-reset
   iptables -A INPUT -j REJECT --reject-with icmp-proto-unreachable
   iptables -A TCP -p tcp --dport 80 -j ACCEPT
   iptables -A TCP -p tcp --dport 443 -j ACCEPT
   iptables -A TCP -p tcp --dport 22 -j ACCEPT
   iptables -A OUTPUT -p tcp --tcp-flags ALL SYN -m state --state NEW -j DROP
6. Running services: netstat –tulpn
7. Remove unnecessary services

REFERENCES

http://firewallingit.blogspot.com/2015/04/ccdc-debian-hardening-guide.html?m=

What steps do you take to secure a Debian server?
http://serverfault.com/questions/11659/what-steps-do-you-take-to-secure-a-debian-server

- disable root login
- disable login by password (allow only login by public-key)
- change SSH port
- use denyhosts (or similar)
- write your own iptbles script (so you control exactly what to allow and can drop everything else)
- force the use of SSL/TLS secured communications and make sure to have valid, non-expired and signed certificates
- turn on strict certificate verification for all external services (for example when authenticating users with an LDAP server on another machine)

Obligatory:
- installation of system with expert mode, only packages that I need
- hand written firewall with default policy on iptables' input: drop, permitting access to SSH, HTTP or whatever else given server is running
- Fail2Ban for SSH [and sometimes FTP / HTTP / other - depending on context]
- disable root logins, force using normal user and sudo
- custom kernel [just old habit]
- scheduled system upgrade

Depending on level of paranoia, additionally:
- drop policy on output except for a couple of allowed destinations / ports
- integrit for checking if some parts of file system ware not modified [with checksum kept outside of the machine], for example Tripwire
- scheduled scan at least with nmap of system from the outside
- automated log checking for unknown patterns [but that's mostly to detect hardware malfunction or some minor crashes]
- scheduled run of chkrootkit
- immutable attribute for /etc/passwd so adding new users is slightly more difficult
- /tmp mounted with noexec
- port knocker or other non-standard way of opening SSH ports [e.g. visiting 'secret' web page on web server allows incoming SSH connection for a limited period of time from an IP address that viewed the page. If you get connected, -m state --satete ESTABLISHED takes care of allowing packet flow as long as you use a single SSH session]

Things I do not do myself but make sense:
- grsecurity for kernel
- remote syslog so logs cannot be overwritten when system gets compromised
- alerting about any SSH logins
- configure rkhunter and set it up to run from time to time

Disabling root user login:

http://www.howtogeek.com/howto/linux/security-tip-disable-root-ssh-login-on-linux/

https://www.liquidweb.com/kb/disabling-root-user-login/

https://wiki.debian.org/sudo#Verifying_sudo_membership https://wiki.debian.org/sudo

To disable root logins through ssh, edit the file /etc/ssh/sshd_config. Refer to your distribution's documentation in case the location of the file is different in your distribution. In this file, change the line that says "***PermitRootLogin yes***" to "***PermitRootLogin no***". If there is a # at the start of the line, remove it. Now save the file.

This hole might already be plugged in your specific distribution by default, but it doesn't hurt to check. Once the file has been modified and saved, you must restart the ssh daemon to use the new configuration. To do so, type ***"/etc/init.d/ssdh restart***" to make it use the new configuration file.

The /etc/securetty file defines the local terminals (Alt+Shift+<number from 1-6>) on the computer which are considered secure and can be allowed to have a root login. Simply having a blank securetty file ensures that all local terminals are considered insecure and will not allow anyone to login using the root account.

echo > /etc/securetty

This command will overwrite the contents of /etc/securetty with a blank file. Make sure to store a backup of the original file if you ever think you might need it.

Since this is an email server, it'll likely use SMTP/be vulnerable to SMTP-based attacks:

This article looks at changing the configuration file so that we can monitor the mail.log file and take action against suspect connections over smtp (port 25):

http://www.the-art-of-web.com/system/fail2ban-sendmail/

http://www.dummies.com/programming/networking/smtp-hacks-and-how-to-guard-against-them/

Exim (default Debian, likely to be the MTA): https://wiki.debian.org/Exim

https://nmap.org/nsedoc/scripts/smtp-vuln-cve2010-4344.html

Using Postfix for Secure SMTP Gateways: http://www.linuxjournal.com/article/4241

https://wiki.debian.org/Postfix

Fail2Ban:

https://debaday.debian.net/2007/04/29/fail2ban-an-enemy-of-script-kiddies/

http://www.the-art-of-web.com/system/fail2ban/

http://www.ducea.com/2006/07/03/using-fail2ban-to-block-brute-force-attacks/

http://www.the-art-of-web.com/system/fail2ban-sendmail/

https://www.unixmen.com/how-to-prevent-ssh-brute-force-attacks-with-fail2ban-on-debian-7/

Intrusion detection:

http://serverfault.com/questions/2783/how-do-i-know-if-my-linux-server-has-been-hacked

http://serverfault.com/questions/650/how-can-i-detect-unwanted-intrusions-on-my-servers/800#800

Debian Goodies:

1. dglob – Produce a list of package names which match a pattern
2. dgrep – Search all files in given packages for a regex
3. dpigs – Display which installed packages taken the most disk space
4. debget – Obtain a .deb for a package in APT's database
5. debmany – Choose manpages of installed or removed packages
6. **checkrestart** – Finds and restart processes which are using outdated versions of upgraded files***
7. popbugs – Show a customized release-critical bug report based on packages you use
8. which-pkg-broke – Catch which package might have broken another

Referenced below: https://www.debian.org/doc/manuals/securing-debian-howto/ch4.en.html

4.4 Set a LILO or GRUB password

Anybody can easily get a root-shell and change your passwords by entering <name-of-your-bootimage> init=/bin/sh at the boot prompt. After changing the passwords and rebooting the system, the person has unlimited root-access and can do anything he/she wants to the system. After this procedure you will not have root access to your system, as you do not know the root password.

To make sure that this cannot happen, you should set a password for the boot loader. You can choose between a global password or a password for a certain image.

For LILO you need to edit the config file /etc/lilo.conf and add a password and restricted line as in the example below.

```
      image=/boot/2.2.14-vmlinuz
          label=Linux
          read-only
```

```
password=hackme
restricted
```

Then, make sure that the configuration file is not world readable to prevent local users from reading the password. When done, rerun lilo. Omitting the restricted line causes lilo to always prompt for a password, regardless of whether LILO was passed parameters. The default permissions for /etc/lilo.conf grant read and write permissions to root, and enable read-only access for lilo.conf's group, root.

If you use GRUB instead of LILO, edit /boot/grub/menu.lst and add the following two lines at the top (substituting, of course hackme with the desired password). This prevents users from editing the boot items.timeout 3 specifies a 3 second delay before grub boots the default item.

```
timeout 3
 password hackme
```

To further harden the integrity of the password, you may store the password in an encrypted form. The utility grub-md5-crypt generates a hashed password which is compatible with GRUB's encrypted password algorithm (MD5). To specify in grub that an MD5 format password will be used, use the following directive:

```
timeout 3
 password --md5 $1$bw0ez$tljnxxKLfMzmnDVaQWgjP0
```

The --md5 parameter was added to instruct grub to perform the MD5 authentication process. The provided password is the MD5 encrypted version of hackme. Using the MD5 password method is preferable to choosing its clear-text counterpart. More information about grub passwords may be found in the grub-doc package.

4.5 Disable root prompt on the initramfs

Note: This applies to the default kernels provided for releases after Debian 3.1

Linux 2.6 kernels provide a way to access a root shell while booting which will be presented during loading the initramfs on error. This is helpful to permit the administrator to enter a rescue shell with root permissions. This shell can be used to manually load modules when autodetection fails. This behavior is the default for initramfs-tools generated initramfs. The following message will appear:

```
"ALERT!  /dev/sda1 does not exist.  Dropping to a shell!
```

In order to remove this behavior you need to set the following boot argument:*panic=0*. Add this to the variable GRUB_CMDLINE_LINUX in /etc/default/grub and issue update-grub or to the append section of/etc/lilo.conf.

4.9 Restricting the use of the Magic SysRq key

The *Magic SysRq key* is a key combination that allows users connected to the system console of a Linux kernel to perform some low-level commands. These low-level commands are sent by pressing simultaneously *Alt+SysRq* and a command key. The SysRq key in many keyboards is labeled as the *Print Screen* key.

Since the Etch release, the Magic SysRq key feature is enabled in the Linux kernel to allow console users certain privileges. You can confirm this by checking if the /proc/sys/kernel/sysrq exists and reviewing its value:

```
$ cat /proc/sys/kernel/sysrq
 438
```

The default value shown above allows all of the SysRq functions except for the possibility of sending signals to processes. For example, it allow users connected to the console to remount all systems read-only, reboot the system or cause a kernel panic. In all the features are enabled, or in older kernels (earlier than 2.6.12) the value will be just 1.

You should disable this functionality ifaccess to the console is not restricted to authorised users: the console is connected to a modem line, there is easy physical access to the system or it is running in a virtualised environment and other users access the console. To do this edit the /etc/sysctl.conf and add the following lines:

```
# Disables the magic SysRq key
 kernel.sysrq = 0
```