PCI Compliance

PCI compliance is easily achievable. There are three steps to this.

1. Get more information from the client so that you can know which services they'll be OK with you disabling.
2. Deal with the SSL ciphers as set within the WHM, and disable any services that they would rather not run.
3. Update any old versions of software.
4. Deal with any issues within RedHat (e.g. backporting) by obtaining logs that show the CVEs being resolved.

Get more information from the client so that you can know which services they'll be OK with you disabling.

-Warn the client that this may affect connectivity to their services! Especially for email and FTP.

-Ask the client if they're using email or FTP on the server. If they aren't, feel free to disable it in the Service Manager! I usually try to ask about 'cPanel DAV Daemon' here as it is almost never used, as well - and I disable it in the Service Manager if it isn't being used. You want to lower the possible attack points. Once you've cleared this with the client, you can then go on to set the ciphers.

Deal with the SSL ciphers as set within the WHM, and disable any services that they would rather not run.

-Check the SSL Cipher Suite within the Apache Configuration and grab the top one. What I'm currently using is the following:

- ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:AES:CAMELLIA:DES-CBC3-SHA:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK:!aECDH:!EDH-DSS-DES-CBC3-SHA:!EDH-RSA-DES-CBC3-SHA:!KRB5-DES-CBC3-SHA

Then go into the individual configuration pages:

-Apache Configuration:

- Change the SSL Cipher Suite to that default.
  I would recommend checking this when you're adding it to ensure that you're getting and propagating the 'latest' one, instead of copy-pasting the one above. Before cPanel had one I usually used Mozilla: https://wiki.mozilla.org/Security/Server_Side_TLS#Modern_compatibility Both of these should work but I would default to the cPanel one unless it causes serious problems.
- Add -TLSv1 to SSL/TLS Protocols

-Mailserver Configuration:

- Change the SSL Cipher Suite to the default we have above.
- Add !TLSv1 to SSL Protocols
- Disable "Allow Plaintext Authentication (from remote clients)".
- **Please note that you also need the change in 'Exim Configuration' for this to properly secure everything**

-Exim Configuration

- In Basic Exim Configuration, enable 'Require clients to connect with SSL or issue the STARTTLS command before they are allowed to authenticate with the server'
- Go into 'Advanced Exim Configuration', find openssl_options, and change it to this (adding the last value): +no_sslv2 +no_sslv3 +no_tlsv1

-FTP Configuration:

- Change 'TLS Encryption Support' from 'Optional' to 'Required (Command/Data)'.
- Add !TLSv1 to SSL Protocols. You'll want to use either a cipher like this, that simply adds blocking TLSv1: HIGH:!TLSv1:!SSLv2:!SSLv3:!ADH:!aNULL:!eNULL:!NULL
- Or alternatively you can use something that specifies everything it allows and disables TLSv1 explicitly (this would be more secure but may also be more annoying to set up):
  ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:AES128-GCM-SH

A256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:AES:CAMELLIA:DES-CBC3-SHA:!TLSv1:!SSLv2:!SSLv3:!ADH:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK:!aECDH:!EDH-DSS-DES-CBC3-SHA:!EDH-RSA-DES-CBC3-SHA:!KRB5-DES-CBC3-SHA
- Make sure that Allow Anonymous Logins and Allow Anonymous Uploads are both turned off.

-cPanel Web Services Configuration:

- Replace the SSL cipher with the first one listed here.
- Add :!TLSv1 to SSL/TLS Protocols.

-ConfigServer Security and Firewall

- If you aren't using a service at all, disable the firewall ports.

-If they have LiteSpeed, you'll want to secure their LiteSpeed Admin page. You'll want to secure it with this list:

https://www.litespeedtech.com/support/wiki/doku.php/litespeed_wiki:config:admin-ssl

Basically:

- Go to http://(ip):7080/config/confMgr.php?m=altop
- Add a listener called 'AdminListenerSSL': make it listen on port 7081 and require SSL
- Go to http://(ip):7080/config/confMgr.php?m=al_adminListenerSSL&p=lsecure&t=L_SSL_CERT&a=e
- In the shell, go to /var/cpanel/ssl/cpanel/mycpanel.pem : this is your current root certificate. Copy the middle certificate into /usr/local/lsws/conf/cert/admin.crt, and the top RSA one into /usr/local/lsws/conf/cert/admin.key. You can do this with commands:
  sed -n '/-----BEGIN RSA PRIVATE KEY-----/,/-----END RSA PRIVATE KEY-----/p' mycpanel.pem > /usr/local/lsws/conf/cert/admin.key
  sed '/-----BEGIN CERTIFICATE-----/,/-----END CERTIFICATE-----/!d;/-----END CERTIFICATE-----/q' mycpanel.pem > /usr/local/lsws/conf/cert/admin.crt
- Set the private key file to /usr/local/lsws/conf/cert/admin.key and the certificate file to /usr/local/lsws/conf/cert/admin.crt
- Edit 'SSL Protocol', enable 'TLS v1.1   TLS v1.2; and use the first cipher as the cipher for the page. You can ignore the other variables.

That should be it!

Update any old versions of software.

This is fairly self-explanatory. If they're running an older version of PHP that has been flagged within the scan, inform them that this has happened and ask if it's all right for you to upgrade them. When they assent, upgrade them - either through EasyApache (for PHP or Apache) or through our MySQL upgrade guide.

Deal with issues within RedHat (e.g. backporting) by getting logs that show the CVEs being resolved.

Since all of our servers are running CentOS, it's important to understand how that functions:

https://access.redhat.com/security/updates/backporting

After receiving the PCI report, here is what to tell the customer:

"While the scanning company detected a service version that is an older version, it does not necessarily mean that the services are vulnerable. The server is running CentOS, which is based on Red Hat Enterprise Linux (RHEL). RHEL utilizes a system called backporting for all of their software packages. Backporting is the action of taking parts from a newer version of a software system or software component and porting them to an older version of the same software. It forms part of the maintenance step in a software development process, and it is commonly used for fixing security issues in older versions of the software and also for providing new features to older versions.

I will review this report and check on the 'High/Critical' compliance status results which are causing the server to fail the scan. Once I have compiled a response, then you can use the information to file for an exception based on the status of the server."

With the report open, you will need to work through each 'High/Critical' compliance status line-by-line.

Make sure you run a 'yum -y update'

It's also important to provide the following information to the customer to rely back to the scanning company:

```
cat /etc/redhat-release

uname -a
```

Examples of common CVEs and how to address them:

> This is not an exhaustive list of possible issues. Just use the indicated CVE and Google it for more information. Also keep in mind that most reports have detailed information in the 'Full Report' section toward the bottom.

OpenSSH / OpenSSL

Example: OpenSSH Wildcards on AcceptEnv Vulnerability, CVE-2014-2532

```
rpm -qa |grep openssh

openssh-server-5.3p1-112.el6_7.x86_64

openssh-clients-5.3p1-112.el6_7.x86_64

openssh-5.3p1-112.el6_7.x86_64
```

```
rpm -q --changelog openssh | grep -B 5 -A 5 CVE-2014-2532
- better fork error detection in audit patch (#1028643)
- fix openssh-5.3p1-x11.patch for non-linux platforms (#1100913)

* Wed Jun 18 2014 Petr Lautrbach <plautrba@redhat.com> 5.3p1-96
- prevent a server from skipping SSHFP lookup (#1081338) CVE-2014-2653
- ignore environment variables with embedded '=' or '\0' characters CVE-2014-2532
- backport ControlPersist option (#953088)
- log when a client requests an interactive session and only sftp is allowed (#997377)
- don't try to load RSA1 host key in FIPS mode (#1009959)
- restore Linux oom_adj setting when handling SIGHUP to maintain behaviour over restart
(#1010429)
```

Once you've compiled the required information for the customer, you can send it back line-by-line, e.g. this example:

"Hello $CUSTOMERNAME

Please see my responses in-line:

> #1) OpenSSH Wildcards on AcceptEnv Vulnerability, CVE-2014-2532

This was patched via backporting:

```
rpm -qa |grep openssh

openssh-server-5.3p1-112.el6_7.x86_64
```

openssh-clients-5.3p1-112.el6_7.x86_64

openssh-5.3p1-112.el6_7.x86_64

```
rpm -q --changelog openssh | grep -B 5 -A 5 CVE-2014-2532
- better fork error detection in audit patch (#1028643)
- fix openssh-5.3p1-x11.patch for non-linux platforms (#1100913)

* Wed Jun 18 2014 Petr Lautrbach <plautrba@redhat.com> 5.3p1-96
- prevent a server from skipping SSHFP lookup (#1081338) CVE-2014-2653
- ignore environment variables with embedded '=' or '\0' characters CVE-2014-2532
- backport ControlPersist option (#953088)
- log when a client requests an interactive session and only sftp is allowed (#997377)
- don't try to load RSA1 host key in FIPS mode (#1009959)
- restore Linux oom_adj setting when handling SIGHUP to maintain behaviour over restart
(#1010429)
```

>#2) SSL/TLS Weak Encryption Algorithms, CVE-2013-2566 CVE-2015-2808

This was resolved by changing the 'Cipher Suite' used within the Apache configuration to the following and disabling TLSv1:

```
ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:AES:CAMELLIA:DES-CBC3-SHA:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK:!aECDH:!EDH-DSS-DES-CBC3-SHA:!EDH-RSA-DES-CBC3-SHA:!KRB5-DES-CBC3-SHA
```

>#3) PHP Exploit, CVE-1111-1111

This was resolved by updating PHP to the latest version on the server:

```
root@host [1151 10:16:48 ~]# php -v

PHP 5.5.21 (cli) (built: Feb  4 2015 16:17:28)

Copyright (c) 1997-2014 The PHP Group

Zend Engine v2.5.0, Copyright (c) 1998-2014 Zend Technologies

    with the ionCube PHP Loader v4.6.1, Copyright (c) 2002-2014, by ionCube Ltd.
```

You will want to pass along the above information to the PCI scanning company and have them rescan the server now."

Extra Options (that generally aren't required by PCI Compliance but can be in some cases)

-You can tweak SSH ciphers in /etc/ssh/ssh_config (not /etc/ssh/sshd_config!).