# PROPOSAL FOR AMAZIN INC PCI-COMPLIANT AWS INFRASTRUCTURE.

**NAME: NICHOLAS NGUGI NJIHIA**

**Name of the Project: PCI Compliant Ecommerce Infrastructure System Design**

**Name of company : Amazin Inc**

**Submission Date: 3rd July 2020.**

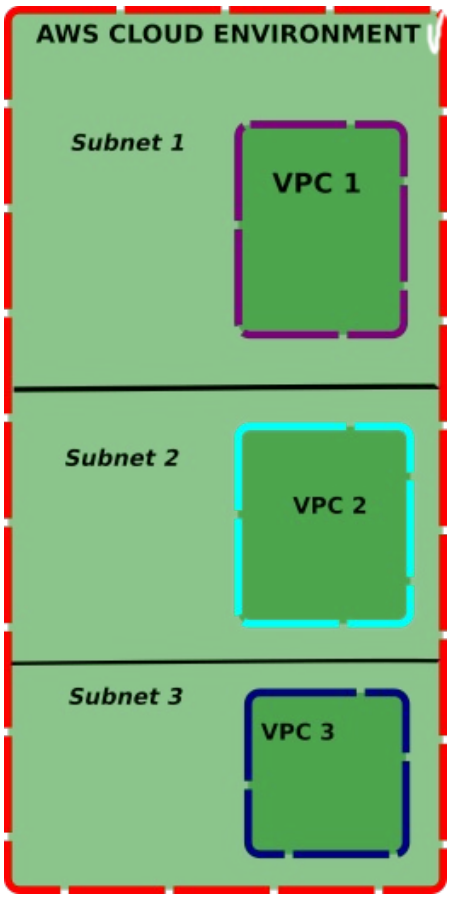
# SUMMARY:

Generally there were three routes I considered using in developing a PCI compliant infrastructure that incorporates Amazon Web Services (AWS).
As such there are multiple templates to choose from.
Since I am not a Qualified Security Assessor or QSA, I would definitely have to choose an architecture already deemed and proven to be PCI DSS compliant.
The 3 routes I considered were:

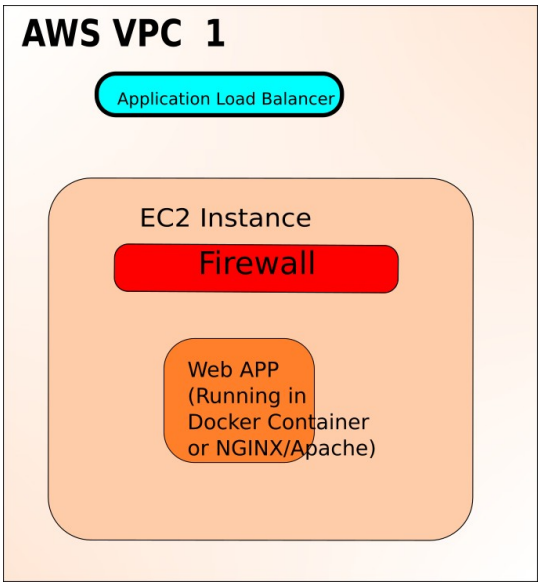
a) Using AWS ***infrastructure resources*** such a plain EC2 Vms and AmazonEKS.
b) Using ***containerized services*** such as AWS ECS, Amazon RDS and Amazon Fargate.
c) Using ***abstracted services*** such as AWS S3 and AWS Lambda.

The containerized services and abstracted services offer greater abstractionwith the abstracted services offering the highest. The use ofabstracted services has the advantages of reducing pentesting workloads which is a requirement of PCI DSS and the need for OS security hardening is gone.
This however does come at the cost of reduced flexibility and greater pricing.

Hence I chose to use mostly Amazon's lower level infrastructure, namely AWS EC2 (Elastic Compute Cloud). This enables me to use many open-source tools like linux firewall Iptables or another option like pfSense, SELinux, PAM (Pluggable Authentication Modules), OpenVPN, OpenLDAP, cron jobs, Linux auditing system,
This architecture is composed of the following:
The overall ***high-level architecture*** is as follows:



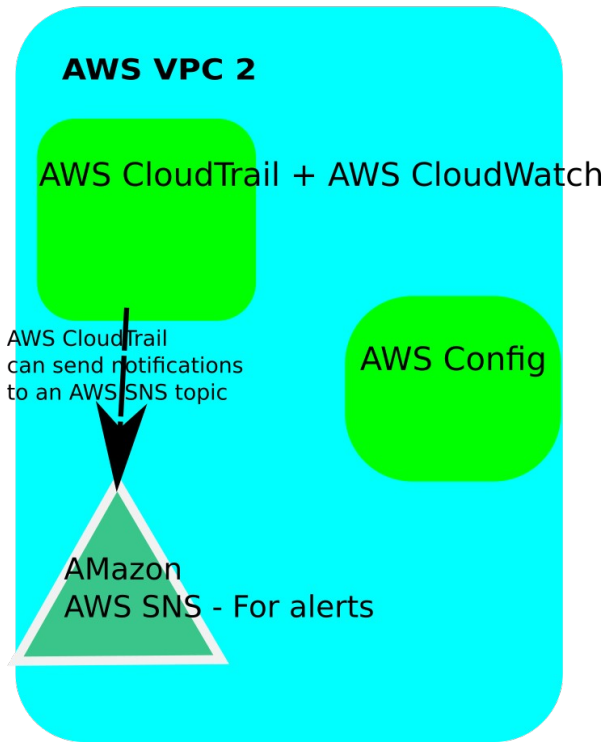
1) **Main production web application Instances(EC2) VPC 1:**




The above EC2 instances have an Applications ELB which sends requests to the appropriate and healthy instances thus sharing the network load.
These instances are Auto-scaled EBS-backed instances.

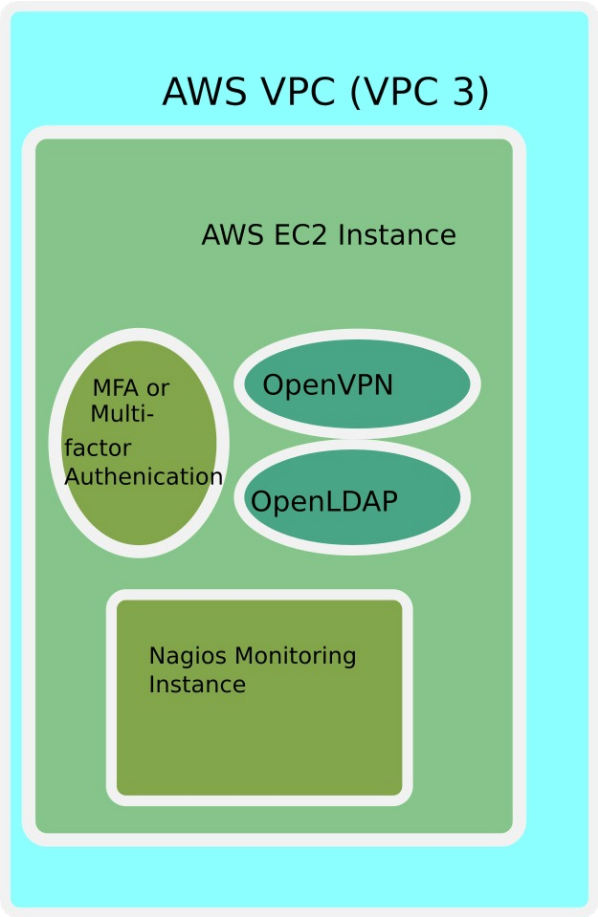
**2. Centralised Logging architecture VPC 2:**

Monitoring and Metrics:



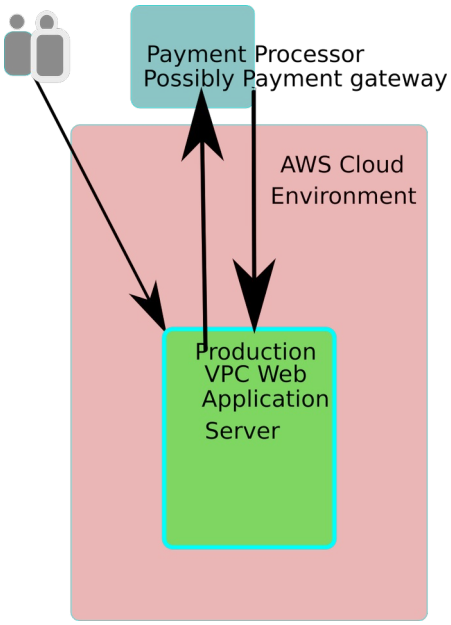
- ***Amazon CloudWatch*** monitors in real time, AWS resources and applications running on AWS. It collects and tracks metrics, metrics being variables used to measure resources and applications. It is referred to as a metrics repository and in our scenario, the auto-scaled production EC2 instances running our applications put metrics into this repository and statistics are generated. CPU usage, disk reads and writes of the EC2 instances will be monitored and the monetary cost of this is not high.
  In addition to monitoring AWS resources, Amazon Cloud Watch can be used to monitor data produced from applications, scripts and services. Amazon Cloudwatch also has Amazon CloudWatch Alarms used to initiate automatically an action in response to a predefined condition, for instance when a certain metric reaches a certain threshold. The actions Amazon CloudWatch Alarms can do include triggering an AutoScaling policy if network/memory/disk resources are overused, or publishing to an Amazon SNS topic or updating the CloudWatch dashboard. It can handle EC2 instances like loss of network connectivity or system power, software issues on the physical host, exhausted memory, corrupted file system, number of bytes read or written, among others. Even more important are Amazon CloudWatch Events and Amazon CoudWatch Logs the former of which delivers near real-time information stream about changes to the AWS resources, for instance API calls and the latter monitors applications and systems using their log data and also performs log rotation of log files like /var/log/*.
- ***Amazon AWS CloudTrail:*** This service collects and tracks metrics to monitor AWS resources and the applcations that run on it. Integration with ***Amazon CloudWatch Logs*** enables AWS CloudTrail to send events containing API activity to an Amazon CloudWatch Logs log group and these can then trigger alarms.Amazon CloudWatch offers performance monitoring while Amazon CloudTrail adds depth to the monitoring capabilities by focussing on API activity.
- ***Amazon AWS Config:*** This service anables administrators to exercise better governance over resource configurations, to do frequent audits such as on the history of configurations of the resources, security analysis such as analyzing IAM permissions history or the Amazon Security group rules. Also through AWS Config Rules, we can define the desired configurations for out AWS resources and detect changes to the resources' configurations.
- ***Amazon AWS SNS:*** This service coordinates and manages the delivery and sending of messages

**3. Management VPC VPC3:**




The above image shows the secured management server which is an AWS EBS-backed EC2 instance. It has an OpenVPN instance server which has OpenVPN running to allow secure access to the Amazon VPC that has the production servers and client applications for administrative tasks.
**Alternatively** this could be done via a secured AWS ***bastion*** host that facilitates command-line Secure Shell access to the EC2 instances for troubleshooting and systems administrations activities.
In addition, the instance has Nagios monitoring the main production instance designated in this proposal as VPC 1.
It also has OpenLDAP to restrict and control user permissions. This is not a replacement for IAM and security groups however and it is restricted to controlling Linux EC2 virtual machine instances.
Nagios is a monitoring service for our instances and augments the work done by Amazon CloudWatch, Config and CloudTrail and enables granular and very extensive control.

**CUSTOMER INTERACTION WITH ARCHITECTURE:**




It is important to ***reduce the scope of PCI compliance*** by not storing cardholder data as often having these details can lead to greater overhead and is a liability.
This liability extends beyond the risk of losing a card number to the internet and the associated public relations damage.
Properly protecting cardholder data slows down business operations and if it doesn't earn the business money is not worth it.
The costs of having a payment aggregator or a payment gateway (such as Stripe) or a merchant account handling the process of storing this cardholder data and tokenizing the PAN(Personal Account Number) as well as doing things like fraud analysis and detection is not that high as to make managers make the decision to do all this processing on their servers and store this very sensitive data on their databases.

## AWS SERVICES USED (BRIEF OVERVIEW):

i. ***AWS CloudTrail:*** AWS CloudTrail records AWS API calls and delivers log files that include caller identity management, time, source IP address, request parameters, and response elements. These details enable security analysis, resource change tracking and compliance auditing.
ii. ***AWS CloudWatch:*** This is a monitoring service for AWS cloud resources and the applications running on AWS. Amazon CloudWatch collects and tracks metrics, collects and monitors log files, sets alarms, and automatically reacts to changes to your AWS cloud landscape/resources.
iii. ***AWS Config:*** This provides resource inventory, configuration history and configuration change notifications to enable security and governance.
iv. ***Amazon EC2 instances:*** These enable the launch of virtual machine instances from a variety of various operating systems. They can be either Amazon EBS-backed EC2 instances or Amazon Instance Store-backed EC2 instances, each with different features.
v. ***AWS Elastic Load Balancing:*** This automatically distributes traffic across multiple EC2 instances to help achieve better fault tolerance and high availability.

vi. ***Amazon VPC:*** The Amazon Virtual Private Cloud lets its users provision a private, logical isolated section of the AWS Cloud where you can launch AWS services and other resources in a virtual network that you define. Hence it enables complete control over your virtual networking environment, including the selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways.

vii. ***(Optional) Amazon CloudFront :*** Given that Amazin is deploying a web application, it can be very useful depending on pricing requirements and location of customers, to have CloudFront Content Delivery Network set up to cache web content across distributed areas that are located far from the originating servers and also to offload the origin servers.

## OpenSource Linux Tools Used:

i. ***OpenVPN:*** A VPN enables an administator to create a local network between hosts on different networks/segements, often distant from each other across the internet.Network traffic flowing over a VPN is often referred to as being inside a VPN tunnel. When a VPN is used, the traffic inside the tunnel is no longer identifiable.

   OpenVPN is a robust Open source VPN tool that uses the SSL/TLS protocol to secure the connection. It also uses HMAC in combination with a digest or hashing algorithm for ensuring the integrity of the packets delivered. It is often more secure than other closed-source VPNs as its code is continually inspected by different people and is in the public. Also there's very little chance of secretback doors being built into OpenVPN.

ii. ***OpenLDAP:*** This is a network directory system that stands for Open Lightweight Directory Access Protocol. It is related to Microsoft Windows Active Directory which also uses the LDAP protocol. This service uses a centralized hierachical database to store information regarding network objects, and these objects in our case are usernames, passwords and permissions associated with users, for instance who is allowed to connect to the EC2 instances that entail the cardholder environment.

   To install OpenLDAP on Red-Hat based distributions we install packages `openldap-servers and openldap-clients` respectively.

   ```
   sudo yum install openldap-servers openldap-clients
   ```

   To install on Debian/Ubuntu and their derivatives:

   ```
   sudo apt install slapd ldap-utils
   ```

iii. ***Linux/UNIX Firewalls:*** Here there is a great deal of choice of firewalls such as Iptables offering the greatest amount of flexibility, though at the cost of taking the time to learn it. Though, there are firewall clients like ufw and firewalld which use Iptables and use simpler commands. Another excellent choice for a firewall is pfSense.

   These firewalls can also be used to log traffic and this can be done while rate limiting what hits the server.

iv. ***Open source Host-based Intrusion Detection systems(Open source HIDS):***

These include Open Source Tripwire, OSSEC and Samhain which uses cryptographic hashing to monitor any file changes on your system. This is done by generating and rcording the hashes of any file visible on the filesystem initially during the software's first run.

Here I have chosen ***OSSEC*** and ***md5deep***. The latter's hash list has to be stored elsewhere such as in an s3 bucket which has encryption to avoid attackers overwriting the MD5sum list if they compromise the instance.

v. ***Network-based IDS:*** Here I have used ***Snort(***https://www.snort.org/***)***. Snort is not just a Network Intrusion Detection System (NIDS) but a packet logger and a packet sniffer as well.

As a packet logger it can be useful in reading packets off the network and reading them to disk, preferrably in a binary format which can then be read back usig snort or tcpdump.

However, undoubtedly it's greatest use is as a NIDS. Snort is configured to log packets that trigger rules predefined in snort.conf. It is highly configurable and doesn't just log but performs actions like alerts, activates dynamic rules among other things. However, it is to be noted that snort can be overshadowed by some commercial IDS in terms of features.

***vi. Antivirus and rootkit hunters:*** For the latter, there's an excellent tool called a ***RootKit Hunter*** also called rkhunter that protects files from rootkits. The Rootkit Hunter developers describe it as a "host-based, passive, post-incident path-based tool". The 'passive' reference means that the software has to be run manaully or scheduled via cron or systemd service units. The "path-based" reference means that it doesn't operate heuristically like a virus chacker might and only operates on files

Thus we also need an anti-virus and malware detecting software.

The best and most robust Linux based open-source anti-virus is ***ClamAV***.

***LMD*** or ***Linux Malware Detect*** is a popular sophisticated software package from R-fx Networks(https://www.rfxn.com) that helps to mitigate malware threats on a Linux system. It focusses exclusively on malware. The good thing about LMD is that LMD pulls in updates regularly from which it generates malware signatures(signatures receive an upate aproximately once a day or more).

***vii. Nagios: Nagios*** is an open source tool for system monitoring. It watches servers and other devices on your network and makes sure they're working properly. It also verifies that services on those machines are working properly. In addition, Nagios can accept information from other processes or machines regarding their status, for instance an anti-virus or web server, hence if a web-server is overloaded, Nagios will detect this.

**Note:** I still use ***AWS CloudWatch, AWS CloudTrail, AWS Config and AWS SNS*** as they're excellent and top-notch in monitoring different metrics on running EC2 instances, RDS instances (with CloudWatch), monitoring and logging API calls to AWS resources (AWS CloudTrail), sending alerts (AWS SNS), and resource inventory and configuration history (AWS Config). Hence Nagios is an add-on to these already very capable services.

***viii. Netcat:*** Netcat has a lot of features and one of this is the ability to add ACLs or Access Control Lists. For example the following command will allow all machines access to a given host and deny only the one specified in the command:

```
sudo ncat -l –-deny 1222:cb88::3b
```

# GENERAL OVERVIEW OF ARCHITECTURE:

- ***Customer checkout:*** The web application will have a checkout or pay button that takes the merchant's (Amazin) customer to the checkout form to fill in data such as his/her personal account number, name on the card, expiry date of card and possibly the CID/CAV2(found at the back of the card or at the front(for American Express/AMEX)).
- ***Segmented secure cloud infrastructure:*** In the cloud, this web app lies in a segmented and secured cloud environment. This segmentation is achieved through the use of Amazon AWS VPC or Virtual Public Clouds which have associated private/public subnets with them. There could be various web apps running on various Amazon EBS-backed EC2 instances each connected to an Application or network load balancer. The application load balancer is preferred in this case. The EC2 instances have autoscaling enabled to grow with increased customer traffic.
- ***Firewall(IPtables, pfSense or AWS WAF) (PCI DSS Requirement 1):*** For security, these EC2 instances need to be configured with a firewall that disables all connections apart from that with the load balancer and they're assigned private IP addresses.
  The ALB (Application Load Balancer) will accept HTTPS connections from customers and hence will have a public IP unlike the EC2 instances running the application code.
  AWS offers AWS WAF or Web Application Firewall but there are also other Open source alternatives such as Iptables which uses netfilter, as well as pfSense.
- ***Reducing PCI Compliance Scope by using a payment gateway/payment aggregator/merchant account:*** It is important that none of this cardholder data is stored in Amazin's servers or in any database and as such only verification of payment is needed from the payment gateway. After the payment gateway processes the transaction(e.g. shoes bought on the ecommerce platform), it sends all the information for approval to the issuing bank via the acquiring bank and relevant card schemes (VISA, Mastercard, AMEX, etc). After approval, once the payment has been confirmed by the card schemes, the payment gateway sends the approved transaction back to the merchant's website (Amazin in this case) and then the merchant informs the customer that the purchase has been successfully completed. If approval fails, this information is also shown on the website to the customer.
- ***OpenVPN:*** Administrative tasks such as management of the instances and configuration editing, patch management are done through an OpenVPN server running on an EC2 Instances that is highly secured.
- ***Logging and monitoring with AWS CloudTrail, CloudWatch and Config as well as with Open Source tools like Nagios, IPTables and SELinux(PCI DSS Requirement 10):*** Logging, monitoring, alerts and metrics are done through AWS CloudTrail, CloudWatch and AWS Config. There are various OpenSource tools which also allow even more fine-grained or granular visibility. Which AWS services may lack. These have greater visibility into the applications running in the resources.

- **Access Controls(PCI DSS Requirement 7):**For access controls or ACLs, the first thing that is needed is to protect the AWS account. This is done using the Identity and Access management (IAM) and multifactor authentication.
  The root AWS account should never be used, and should be protected by multifactor authentication as well, in addition to strong passwords.
  OpenLDAP provides a good choice as a open source Access control software within the running Linux Virtual machines. Another stack on this pyramid of user access and permissions control is Netcat and SELinux. SELinux is especially crucial in the Linux VMs.
- ***Anti-virus Software:*** The OpenSource ClamAV antivirus is a good choice for an antivirus. It is augmented by the malware detection tool LMD as well as the rootkit detector Rootkit Hunter, which are both top-notch.
- ***Intrusion Detection Systems(IDS):*** For intrusion detection, there are numerous open-source solutions such as OSSEC, MD5deep and Samhain(https://www.la-samhna.de/samhain/).
  OSSEC is a host-based intrusion detection system (IDS) with file system integrity monitoring and is backed by Trend Micro.
  Snort is a good choice for a network intrusion detection system (IDS).
- ***Proper Key Management:*** To protect keys and ensure key rotation, AWS KMS is a very good choice and is affordable. It also supports powerful audit trail through AWS CloudTrail.
- (Optional) It is possible to have AWS CloudFront to perform tasks like caching web content and reducing the origin's servers workload. AWS would then connect via HTTP to an S3 bucket or an Amazon EC2 web server that has the web content.

# SECURING THE EC2 INSTANCE LINUX VMS:

- PCI DSS 1.1 requires that a merchant establish firewall and router configuration that includes a formal process for approving and testing all network connections and changes to the firewall and router configuration changes, current network diagram that identifies all connections between the cardholder data environment and other networks, description of groups, roles and responsibilities for management of network components, among other specifications.
  Iptables is often available in most Linux systems and is the most ubiquitous Linux firewall. Another great firewall is pfSense.
  PCI 1.1.6b states that there should be documentation for all security features implemented for those protocols considered insecure. Hence we need to identify insecure services and disable these services. Where possible, insecure services should be replaced with a more secure alternative. For example, all Linux distributions use now SSH by default for remote administration. A protocol like telnet should therefore no longer be used. The "r" services like rexec, rlogin, rcp, and rsh, are insecure as well.
- Unencrypted protocols include FTP, HTTP, IMAP, POP3, SNMP v1/v2 and Telnet which should all be replaced by their secure alternatives, namely FTPS/SFTP/SCP, HTTPS, IMAPS, POP3S, SNMP v3 and SSH or Mosh respectively.
  To determine which protocols are running in an instance, netstat or ss utilities can be used.
  For netstat(On Ubuntu):
  ```
  sudo apt install net-tools
  netstat -nlp
  ```
  On RHEL, Fedora:
  dnf install net-tools
  netstat -nlp
  On Centos/RHEL:

sudo yum install net-tools
netstat -nlp
Systems which have no netstat utility available can use ss instead. For example to display listening TCP connections:
`ss -lnt`
For UDP: `ss -lnu`
Disabling unnecessary listening services is important to reduce the attack surface of the system.Vendor-supplied defaults and security parameters

- **Vendor-supplied defaults and security parameters**

PCI section 2.2.3 specifically states that SSLv3 and early TLS versions should no longer be used.
On Apache installations, SSLv2 and SSL v3 should be disabled by adding the SSLProtocol option, specifying which protocols NOT to use. This is done in the Apache configuration file which is /etc/apache2/httpd.conf in Debian flavours like Ubuntu and /etc/httpd/conf/httpd in Red-Hat based distributions.
On NGINX too, only the newest TLS versions should be enabled by using the *ssl_protocols directive.* This is added to the configuration file (e.g. /etc/nginx/nginx.conf), and applied to the *http* context. See more details in the nginx SSL module.
ssl_protocols TLSv1.2 TLSv1.3;

- **Vulnerability management**

***Requirement 5*** emphasizes the use of software that is capable of detecting, removing and protectig against all known types of malicious software, and that for systems that are not considered to be susceptible to malware, there should be periodic evaluations to confirm that they continue not to be susceptible.
To ensure this item of the PCI DSS compliance verification is completely done, the following steps are done:
To keep ClamAV updated, the utility known as **freshclam i**s used. Additionally monitoring can be set to check that signatures are regularly updated. One option is to use the **clamconf** utility and determine the date of the signatures.
Determine how freshclam is running (freshclam in daemon mode or manual). Additionally check if it is properly logging to **/var/log/clamav/freshclam.log**. Determine if freshclam encountered any issues, like outdated definitions.
ClamAV runs in either daemon mode or as a manual scanner.
Processes can interface with the Clam daemon and request a file descriptor to be tested. The daemon then will return if the file is OK, or a different code (infected, no permissions etc).The manual scanner utility, clamscan can be used to manually check files. This is also a great way of running a regular scan on Linux machines, by using clamscan in a cronjob.
Check if clamscan is scheduled via a cronjob. Additionally check if clamd is running and available for other software components to use it (e.g. mailbox scanning via MTA).
*Logs are to be retained in accordance with PCI DSS Requirement 10.7, h*ence log files have to be checked in /var/log/clamav and the configuration files (/etc/clamav/clamd.conf and /etc/clamav/freshclam.conf) should have a log file defined. The **clamconf** utility can also be used to quickly determine if logging is enabled by running:
clamconf | grep log
*PCI DSS 5.3 states that anti-virus configurationsa are to be examined, including the master installation of the software and a sample of system components, to verify that the anti-virus software cannot be disabled or altered by users.* Hence the config files should be held only by priviledged users.

- ***Requirement 8*(Authentication)**:
  Systems need to be maintained, which requires legitimate users to have access to them. Within the PCI DSS requirements, there are several controls which highlight the need for proper access, protection, and logging changes. This includes system configuration files (e.g. to PAM or SSH), but also to the logging itself. In other words, we should take appropriate measures to safeguard these configurations itself as well.
  PCI describes in section 8.1.4 that accounts older than 90 days and are unused, should be removed. Unused or inactive accounts on the system might be an unneeded security risk.
  To determine the last time a user logged in, the *last* command can be used.
  Information is stored in binary files **/var/log/wtmp** or rotated files like **/var/log/wtmp.1**.
  Requirement 10.7 specifies that information should be stored/logged for at least 3 months. Enough copies are to be stored on the system itself, or be availed on a central logging server.
  PAM or Pluggable Authentication Modules and SELinux can be combined to ensure the Linux authentication process is highly robust.
  SELinux automatically tries to log and do audits, using the audit subsystem when available or falling back to the regular system logging. For instance, SELinux access vector cache(AVC) denials are logged by default. Denials will be shown in the audit log file */var*log/udit/audit.log but this can be configured explicitly in the */etc/*audit/auditd.conf file.
  Two-factor authentication can also be achieved with PAM. Depending on the solution, the related module needs to be installed, configured and tested. Modules include pam_google_authenticator.so.
- ***Requirement 10:***
  PCI section 10.7 covers the need for an audit trail. On Linux we have normal logging, audit events and accounting. Normal logging is done using syslog which stores varying information such as boot information and kernel events. Of critical importance to PCI compliance is proper log rotation, without destroying(removing or overwritting) previously stored data. The systemd utility, journald which is a journal logging utility will output logs that are of importance to us.
  By default, most log files on Linux based systems will be stored in /var/log. We can do a quick\ check for any files which are world readable, by using *find*.
  ```
  find /var/log -perm -o=r ! -type l
  ```
  It is easy to restrict log file viewing of these entries by changing file permissions using chmod.
  We need to enable remote audit logging(to centralize audit events from multiple hosts) and we can do this by either enabling syslog forwarding or enabling the audispremote plugin.
  This is an SELinux feature. Hence the file */etc/audisp/plugins.d/syslog.conf* will have its active setting set to yes.
  System loggers are unencrypted and often not even guarenteed, data delivery. With the audisp-remote plugin, audit events can be sent encrypted and with guaranteed delivery.

**CONCLUSION:**

The PCI DSS Compliance process is a continuous process.
The fact that all these applications in use have to be updated and themselves vetted for usefulness and potential vulnearabilities clearly shows this. In addition, patches to software have to made.
This is the reason early versions of SSL and TLS are no longer considered safe to use.
Some of the tools used in my architecture are redundant. My motivation for this is that not all tools can do the job wholesome, so they may need to be augmented with other tools.
In practice, the aforementioned architecture may be replaced by a simplified architecture using abstracted or containerized services like Amazon Fargate or Amazon Lambda to simplify the PCI Compliance and requirements even further. But I am skeptical with regards to flexibility, granularity and costs associated with going that way. A template that can be used even by startups can prove useful.