There's a hole in my infrastructure? The road to PCI Compliance

Jonathan Chaitow

GSEC Gold Certification

Author: Jonathan Chaitow

Adviser: Charles Hornat

Accepted: April 28th, 2008

© SANS Institute 2008, Author retains full rights.

Executive Summary

This paper addresses some of the issues faced in working towards a deadline of PCI (Payment Card Industry) Compliance at a major international corporation. – including the key challenges we faced and the current progress as a set of specific changes to the architecture.

The *Client*that is under discussion was deeply involved in a multi-streamed packaged software implementation and had not previously been faced with compliance to a set of externally imposed Security standards of this nature.As the focus from the Board of Directors had been on regular and timely delivery of content throughout the implementation, security was often considered a low priority and was often the first piece of functionality to be pulled from a product release for any given application.

Since the focus from the Board of Directors had always been the regular and timely delivery of content throughout the implementation cycle, security was often considered a low priority and would be the first piece of functionality to be pulled from a product release for any given application.

The *Client*had a relatively comprehensive security policy which was largely followed by individuals and groups within the business but it was commonplace for exemptions and exceptions to the policy to be granted by senior management. As such the prospect of an audit against an external, objective security framework(from which exemptions cannot be made readily)was of significant concern to thestakeholders.

As a heavy user of payment cards, PCI compliance would be of great concern and could have immense financial implications. The company's financial institution would effectively need to accept or refuse the risk associated with non-compliance to these standards for all payment card transactions and then negotiate their fees accordingly.

The first step in complying with PCI DSS (Data Security Standards) was understanding the requirements and their impact on the infrastructure, applications, processes and people. Although the DSS are relatively straight-forward and precise, (at least from the 1.1 version), there was work to be done in interpreting the requirements at an enterprise level. We needed to understand how the varying aspects of security were handled by each system at every layer. For example: the encryption of credit card details at rest, would need to cover application logging, database encryption, storage architecture and backup / recovery mediums for each of the company's systems.

This paper will attempt to explain each of the PCI requirements as they were interpreted at the focus organisation: the challenges they imposed; how they were to be audited against; the impact that complying would have across the business and some of the lessons learned from the experiences. The paper will underline the architectural, technological and procedural changes that were made along with recommendations and preventative measuresthat could be taken.

The paper is written from the experiences of a *Systems Integrator*, and will underline the architectural, technological and procedural changes that were made along with recommendations and preventative measuresthat could be taken.

Specifically, it will discuss in detail the hardening of legacy applications, remediation of user accounts, overhaul of data centers and the implementation of a centralisedauditing and policy managementtool. For each of the PCI requirements, lessons learnt from *theClient* (understood from the perspective of a Systems Integrator) along with previous project experiences are explored.

The Client covered here was maintaining over 1000 legacy applications (many externally facing) and had more than 25,000 employees with one of the largest Production environments in the region. While they had fairly mature documentation and procedures they had not previously been faced with an audit of their compliance to external requirements such as these.

The paper will proceed to highlight the obstacles faced while working with a managed service provider that was selling a 'silver bullet' solution and the recommendations that can be made for any company faced with similar challenges.

DISCLAIMER

The names of all parties involved in this paper have been obfuscated to protect their privacy; all events are described truthfully to the best of my ability having been recorded in personal notes after the fact. The paper is written from these notes and email records to piece together the most accurate representation possible. The author's role at the client was working as Security Architect for a Systems Integrator with no ties to any vendor or service provider and no previous relationship with any other concerned party.

Introduction

With epidemic levels of threats to Information Security the real cost of e-commerce is spiralling out of control. Previously much of the damage caused by hackers, Trojans, worms and 'script kiddies' was absorbed by the Payment Cards Industry with losses due to fraud written off as bad debts.

In an effort to reduce the exposure to these losses, the Payment Cards Industry devised a set of fundamental security guidelines that a merchant, processing customer credit cards, should follow to provide basic hardening. The idea being that when compliant to these principles the risk of theft would be significantly reduced. These guidelines were thereafter known as the PCI Data Security Standards or PCI DSS.

The organisation covered in this paper, for privacy hereafter referred to as the *Client*, was rated as a Level 1 Merchant (processing or storing more than 6 million customer records). Essentially this meant for *the Client* that they would need to become compliant within 18 months or face fines of over $25,000 per day and potentially voiding their current agreements with their bank. A 2008 Poll by Visa indicated that 42% of the Level 1 merchants did not reach their 2007 Compliance deadlines and were fined up to $25,000 per day for each day of non-compliance.

PCI DSS is a full compliance framework - you cannot be partially compliant (just as you cannot be partially pregnant), i.e. 1 infraction means total non-compliance. Ultimately the burden will lie with the financial institution to assume or refuse the risk of a non-compliant merchant. Since the timelines expected to reach full compliance would be in excess of 12 months the costs of non-compliance could quickly become material, not to mention the costs associated with potential forthe bad media coverage and reputation (for the Client, the Systems Integrator andthe Service Providers).

On the other hand, taking a positive approach, there have been instances of clients renegotiating a lower rate with their bank due to PCI DSS Compliance. That is, the financial institution will assume a lower level of risk associated with each of the client's transaction and therefore be able to offer a lower loaded rate per transaction providing financial rewards to the compliant and incentives for those on a PCI roadmap.

In evangelising Security across this project and throughout *the Client* there were significant challenges with deeply entrenchedbusiness processes that circumvent security controls and governance. Some of the legacy applications had been running without being patched or upgraded for years and users had well-established backdoors for doing their job.

At the time of writing, *the Client* was deploying the first release of a major implementation, which involved installing an enterprise CRM (Customer Relationship Management) application, billing application and middleware service bus. When it became clear that PCI DSS compliance would be a priority for *the Client*, the first task for us was education – why was all this extra work necessary and who was responsible?

Additionally *the Client*executives were pragmatic and would often weigh the financial penalties against the total cost of becoming compliant – supposing that it may be cheaper to pay the fines. They were furthermore reluctant to believethat an infraction would jeopardise their relationship with their financial institution, due to the enormous value of their business accounts. PCI and indeed any Security compliance was a hard sell.

Still *the Client* was very supportive of our efforts to harden their infrastructure and applications - though vulnerabilities in either one could surprise them (hence the title of this paper).This was particularly the case with regards to application security - predominantly handled by the SI.

PCI Requirements and their stories

This paper will discuss the situation at *the Client* and the experience with compliance in terms of the PCI Requirements as they are detailed in the DSS specification. An outline of each requirement is describedas follows including how it was interpreted and the context in which it was handled in this case. Each chapter will then go on to propose ways to audit compliance to the requirement internally and recommendations on what/not to do.

1. Building and Maintaining a Secure Internal Network

Building and maintaining a Secure Internal Network proved to be quite a challenge at *the Client*, primarily because their internal network had grown so large and, covering so much of the country that Network Administrators would refer to their LAN (Local Area Network) as a hostile network. So for instance while PCI provided some exemptions for transport layer encryption provided the data was routed *internally*, there was much debate as to whether this would hold up in an audit.

A secure network is the cornerstone of any defense-in-depth strategy and it's not a coincidence that this is the first requirement. If you don't get this right, the rest of your work can go to waste.

1.1. Requirement 1: Install and maintain a firewall configuration to protect cardholder data

INTERPRETATION: This was partly the responsibility of the Managed Service Provider (hereafter referred to as SP) who ran the data centers housingtheClient's externally facing production infrastructure. To the best of our knowledge this requirement was fulfilled well by SP and no breaches were uncovered during our tenure. Due to the relationship between the SI and SP there was little more that could be done to verify the compliance to this requirement and upon inquirythe*Client* had instructed us to wait for the auditors to determine any gap areas.

IMPACT: The requirement also fell upon Client personnel who managed their internal network and had in excess of 200,000 rules on their Cisco PIX firewalls with a lead-time ofup to6 weeks for new rule creation. To create a new Firewall rule a request had to be submitted to the Client infrastructure management teams who would

1. Review the request
2. Assess it for impact
3. Confirm no duplicates or conflicts existing
4. Update the rule set on all firewalls

Typically the request would be filed by completing the following template with the details of the new rule requirements.

Interface					Source System		Destination System		Ports
Source	Target	Direction	Service	Protocol	Source Hostname	Source IP Address	Destination Hostname / URL	Destination IP Address	Port(s) UDP/TCP
	SMTP Gateway	Outbound	SMTP	SMTP					TCP 25

IMPLEMENTATION CHALLENGE: From the Client's perspective PCI DSS 1 was difficult to meet. The infrastructure was in place but was constantly under threat from the business who would need additional ports opened to meet their needs. During a remediation exercise a vast number of duplicate and conflicting rules were identified – clearly slowing the performance of

the firewalls and all traffic that passed through them.

Recommendation

The best approach to achieving compliance in this area is to implement logical segregation so that those systems that store and process sensitive Credit Card data will be segmented from the rest of the network. This will drastically reduce any compliance effort as the hardening of the infrastructure will focus on only those systems and the subnets or physical servers on which they are housed.

Validate each and every rule in the rule set, go through a full remediation from the beginning and require an approval process / justification for every additional port that's open. Firewall rules should be constantly monitored in a large enterprise, at some point, somewhere, someone will request *IP Any Any*.

All client machines should have a personal firewall solution installed and patched with a default rule set. In the case of the SI the BlackICE personal firewall was deployed with the understanding being that minimal additional configuration would be required for their daily usage. However, due to the knowledge required to customize the rule set, more often than not this firewall was simply disabled if it prevented an application from running. A number of End to End Internet Security solutions on the market today will allow for user friendly configuration and customization, perhaps one of the most usable (importantly - with a smallfootprint) is the Zone Alarm Pro (Firewall and Anti-spyware) from Checkpoint.

There is a difficult balance between usability and customizabilityhere but where possible a blank firewall template should be applied to users by default to provide a minimum level of protection. Security templates for most major platforms are available freely online from the NSA[1]. Additionally, no users should be allowed to subvert the enterprise firewalls with Dial-up Internet or other direct connections to the Internet (including ADSL, Cable, Public Wi-Fi, 3G).

1.2. Requirement 2: Do not use vendor-supplied defaults for system passwords and otherSecurity parameters

INTERPRETATION: In an effort to meet aggressive timelines, Security standards and policy were often moved aside in order to quickly deploy applications and services with minimal obstacles. Part of the problem with this approach was the poor conventions, such as using default and vendor supplied passwords ran rampant and often carried through to production as baseline configurations were migrated. The same ran true for user accounts and system configuration providing significant challenges in retrofitting the additional controls (more on this later).

IMPACT: It would be fair to say this was the greatest area of non-compliance at the Client, an initial audit of the /etc/shadow file using *john the ripper* found that although there were over 400 passwords in use with 119 distinct salt values, around 30% of these could be cracked in under 30 seconds using a single core (2.3GHz) machine with 1GB RAM.

This was largely due to the ubiquitous default passwords including "<CLIENT_NAME>", "<APPLICATION_NAME>", "need2change" and "WELCOME1". When accounts were provisioned with default passwords, there was no policy in place forcing the user to change their password and since the default password was always the same there were literally hundreds of 'virgin' accounts lying dormant with these default passwords.

[1] National Security Agency Central Security Service – Security Configuration Guides http://www.nsa.gov/snac/downloads_all.cfm

The password audit was performed by:

1. Logging in as the root account on a Sandbox server (that had the /etc/shadow file replicated from other environments)
2. Copying over SFTP (Secure File Transfer Protocol) / SCP(Secure Copy using WinSCP) the shadow and password files to a local file system.
3. Concatenating the password and shadow files using john the ripper's *unshadow* binary
4. Running the combined password/shadow file against the john(16) cracking application from an MS-DOS (Microsoft Disk Operating System) prompt.
5. The results were then piped to a locally stored output file which was later analysedfor weak passwordsand improvements that could be made to the password policy.

Similarly in the configuration files controlling various system functions including data migration, EBR (Enterprise Backup and Recovery) and EAI (Enterprise Application Integration) there would often be found instances of default passwords that had not been changed after the product was installed. When working to tight deadlines the SI's Development Architecture team, responsible for application installation and configuration would consider their job done when the application was up and running. More often than not, the vanilla install would proceed with default passwords and security parameters (including non secure ports and protocols).

As a result the following password convention was devised and generally adhered to because once the convention was understood, Technical Architects and Service Management people could usually determine the password for most environments and applications.

The password convention was as follows:

- The first 3 characters of the password indicated the type of application that it was used for, the next the actual product name. So a CRM Business Intelligence application would begin crmbi

- The next series of characters indicated the type of user account this was for and the environment it was to be used in. So for an Administrator in an Integration Test environment the next fragment of the password would be adminsit.

- After that the password would indicate the current year followed by @ and the City name. So for example the final password in this case would be crmbiADMINSIT2006@London.

IMPLEMENTATION CHALLENGE:While the convention was difficult to memorize it allowed for relatively strong passwords by default and we would constantly push the theme of *passphrases* rather than passwords.

Prior to this convention, a series of random passwords had been used across each of the environments.Forthose accounts that could not have weak re-used passwords, the account credentials were stored on the network in a Rational repository, in a file named - passwords.xls! The file itself was password protected but needless to say, despite the fact that Microsoft Excel sheets do not implement very sophisticated access control and can be easily cracked with freeware applications, this file would havebeen a relative gold mine for any intruder.

As a next step, instead of relying on passwords - for inter-application authentication a Secure ESB (Enterprise Service Bus) or Federated I&AM (Identity & Access Management) solution may

be feasible. Either of these can be implemented with packaged software and configured to your environment to effectively pass authenticated user credentials across an interface from one application to another. If you're lucky enough to be involved in making the design decisions at the beginning of the project, be sure to investigate options such as these which require some initial investment but will pay security dividends in the long run.

Applying security to applications retroactively proved to be one of the greatest challenges of PCI Compliance. It was sufficiently difficult to change a default password in a configuration file once the application had been put into production. However, retrospectively applying transport layer encryption or implementing password hashes instead of clear text in these same configuration files was far more complex. Often this would require a significant code change within the vendor's application and would be treated as per any other additional functionality or enhancement; it would be prioritized and queued by release management. Needless to say, prior to PCI, security had not been a high priority in the first release of the application stacks.

Recommendation

Do not take an administrator, vendor or SI's assurances with regards to default passwords. The most effective tool here is strict, regular password audits. We used john the ripper for our Unix machines, doing audits of /etc/shadow every 90 days. For Windows machines we ran pwdump5 and piped the results to delimited text files (then concatenated using the *unshadow* binary) for john the ripper with a similar audit process.

Similarly spend some time talking to the users and understanding their approach to security in their daily job. Even the most complex access controls can be easily subverted by relatively simple procedures so it is worth looking into what is followed and what is not. Working on a previous project at a large telecommunications client, Internet access was tightly restricted through their proxy servers with requests for access run through multiple layers of business approval. Still it was found that 1 – 2 accounts with Internet access approved were distributed to all new joiners and then used for proxy authentication. As the proxy allowed multiple sessions, at any one time there could be upwards of 20 users logged in with the same account.

Clearly this would make auditing and non-repudiation nearly impossible. One recommendation for this environment would be to use an Active Directory or LDAP (Lightweight Directory Access Protocol) implementation for proxy authentication so that users would be less inclined to share their account credentials. Similarly one might choose to have employees use their ATM PIN (Personal Identification Number for Automated Teller Machines) for the 4 digit PIN required when connecting remotely using two-factor authentication based VPN (Virtual Private Network).

Finally when using SSH (Secure Shell) tunneling and terminal sessions for remote connectivity, be sure to audit the ~/.ssh/authorized_keyskeyfile, ideally for each user but at a minimum for the *superusers* as this can be a common backdoor into systems (where intruders append a compromised account's public key to the authorized_keys file so they can remotely connect without the password). Better yet would be to enforce password authentication at login by modifying the ssh_config file though you may find some services and applications will require automated / scripted authentication. Also be sure to turn off .rhosts support and delete any remnant .rhosts files. If this is not possible at the very least monitor these files with alarms and triggers to immediately flag any modifications, especially pattern matching like "++" and other derivatives.

2. Protecting our Cardholder Data

As a level 1merchant with large volumes of credit card data processed and stored internally, this was perhaps the most palatable of the PCI Requirements for *the Client*. Requirement 3.4 in particular – rendering the PAN (Personal Account Number) unreadable whenever stored or processed, seemed like the one requirement that made good business sense to all that came across it and was really a focal point for all PCI discussions.

Though it focuses on the PAN itself, this requirement was interpreted as a need to effectively protect all sensitive customer data when at rest or in flight. The PCI DSS themselves provided an objective benchmark for standards such as encryption. When explaining to Client personnel the need to upgrade from 56 to 128 bit encryption, PCI Compliance was usually a well understood reason and gave validity to this benchmark level.

2.1. Requirement 3: Protect stored cardholder data

INTERPRETATION: The first reading of this requirement will most likely centre your attention on databases, LDAP stores, EDWs(Enterprise Data Warehouse) and other forms of data repositories.

IMPACT:Indeed these were significant hurdles in terms of protecting stored Credit Card data, however when digging deeper into the life of a PAN on *the Client*'s internal systems we uncovered some additional locations that proved far less secure, including:

- Application Logs
- EAI / Middleware logs and temporary files
- Database REDO logs and tables
- Email repositories (yes some customers would actually send their unsolicited details)
- Logminer and Data Warehouses
- Server caches
- Audit Tools' internal stores

IMPLEMENTATION CHALLENGE:In light of the above, application databases will often include their own encryption or hashing functionality, which later proved to be a problem when devising a unified approach to key management. For example, the *Client's* CRM application shipped with a built in 128 bit encryption algorithm usingRSA's RC2 cipher applied to database fields marked as sensitive and used for proprietary transport layer encryption.

While on the other hand, the Billing and middleware applications had no such inbuilt functionality and in order to reach compliance on this requirement we went back to the vendors for product enhancements. The Billing application employed a framework known as the DBMS (Database Management System) Obfuscation toolkit which ships with Oracle 9i[2] and essentially consists of a virtual API (Application Programming Interface) of stored procedures and PL/SQL scripts that allow for a transparent layer of encryption. This was crucial to the business as we could not have applied this enhancement if it meant restructuring the database tables and views. Any sensitive data would need to be encrypted as it was stored and decrypted as it left the database to provide transparency to the business functions.

[2] It is worth noting that this problem is solved out of the box with Oracle 10g which ships with the TDE (Transparent Data Encryption - as part of the Oracle Advanced Security plugin), a proprietary framework for achieving this kind of seamless encryption, storing the keys in an 'Oracle wallet' (table based key store). This solution would have been preferable but was not supported by the vendor at the time of writing.

For the middleware application, a product enhancement was required and credit card data was encrypted using a 3DES (Data Encryption Standard) algorithm with 128 bit keys. The keys were generated, distributed, managed, aged, expired and destroyed by *the Client*(using their enterprise key management implementation) so that effectively key management was a service consumed by the middleware application (no SOA puns intended).

2.1.1. Key Management

INTERPRETATION: The Enterprise key management service made use of a hardware based key storage card (plugged into the PCI slot of an E25K frame) and would distribute keys using SFTP scripted file transfers to the respective application server's file system. Application administrators would be notified in advance by automated emails when keys were expiring and would prepare to receive their updated keys from the KDC (Key Distribution Centre) so that once copied across they could be rotated into use.

Prior to the key management functionality being built and deployed across the enterprise[3], each of the applications would generate their own keys (using native application functionality or 3rd party APIs) and either store the keys in a system table within their database or in a protected file on the file system (secured with 0700 access - Owner Read, Write and Execute[4]). These keys would then remain in use without expiration or rotation until a system administrator manually enforced a key update within the application or even manually through a small scale (export/import) re-migration activity.

IMPACT: For the most part, logging was less of a challenge for protecting PAN data and ultimately didn't require the key management functionality. Instead, hashing and obfuscation was utilized instead. Since only the last 4 digits of the card were rendered visible in the application / system logs, no encryption would be necessary.

IMPLEMENTATION CHALLENGE:Email repositories were one of the places we hadn't expected to find credit card information. Around 2 – 3 customers each week would send through their credit card details in unencrypted emails in an effort to pay existing debts without using IVR (Interactive Voice Response) / online bill payment. These emails would be attached to the customer's file by the CSR (Customer Sales Representative). Fortunately the CRM application would store these attachments in a proprietary binary format on the file system with 0700 access. Additionally the credit card details would be linked to a customer account only by the Table ROW_ID so an intruder would need ROOT and SYSTEM access to effectively compromise an identity. Since the credit card data could not be uncovered during penetration testing:

eg: fgrep <KEYWORD><TARGETDIR>/*/*/* or grep <KEYWORD>(executed from the directory containing the email attachments) where "KEYWORD" was a variety of terms that could be used to refer to credit card details ranging from the Column ID (CC_NUM) to common terms like "Visa" and even known numerical combinations like the common local prefix for American Express[5].

It was considered to be sufficiently protected by the native architecture, providing effective

---

[3] The Enterprise key management functionality was designed and deployed to meet this PCI DSS Requirement. Without this compliance driver the lack of consistent approach had been a low priority.

4 The leading 0 in 0700 ensured Set User ID and Set Group ID functionality could not be used to subvert the file permissions by using *'run as'* owner or group owner.

[5] This is a 3 digit number that is specific to the geographic region and has been removed from the paper to protect the client's privacy.

security by obscurity. Still a checkpoint was taken for when the client was audited and this point would need to be agreed on by the auditor and financial institution.
With regards to viewing the PAN on screen, the CRM and Billing applications already had incorporated reasonably tight Role Based Access Control and were able to implement appropriate levels of authorization to ensure only approved users could view this information. If a new view into the data was required it would need to be approved by the change control board including a Client security and privacy officer.

Recommendation

Key management was something that came as a surprise to us, though encryption was employed frequently across the enterprise there was no centralizedapproach to key management. In fact, prior to the PCI requirements, there were no guidelines or policy regarding the generation, strength, distribution, ageing, revocation, expiration or destruction of keys. Ironically the certificate management process was fairly robust but the same process had not been applied for encryption. This should be the first step before any encryption effort is embarked upon.

In this case no volume level encryption was used in any of the environments or client machines largely due to the performance implications. From our perspective there was little sense in applying strong encryption to operating system binaries and non-sensitive data. Instead the focus was on areas storing credit card information or the keys themselves.

Encryption Keys should be stored in secure locations, generated by the KDC and accessible only by the relevant Application account. The keys should then be encrypted with asymmetric PKI (Public Key Infrastructure) encryption and alarmed with file integrity monitoring tools like Tripwire. Key management procedures should apply ageing along with automated expiration and subsequent rotation of keys.

2.2. Requirement 4: Encrypt transmission of cardholder data across open, public networks

INTERPRETATION: As mentioned earlier, *the Client*'s LAN could easily be considered an open public network. Brief site surveys revealed multiple rogue wireless access points, many with weak encryption (40 bit WEP[6] keys) and no physical access control. Often these were set up by employees who were unaware of existing wireless access points or were unable to configure the relatively complex wireless security imposed on officially sanctioned APs (Access Points) including Certificate based 802.1x, WPA2-Enterprise using AES-128 bit encryption and PEAP/TLS over MS-CHAPv2 (Wi-Fi Protected Access using Protected Extensive Authentication Protocol with Transport Layer Security and Microsoft Challenge Handshake Authentication Protocol)

Site surveys were performed using an IBM ThinkPad with a non-amplified built in 802.11a/b/g Atheros Wireless Card and running Netstumbler for Windows XP SP2. The results were dumped to a file for later analysis. No APswere connected to and no sniffing, packet injection, flooding or AP dissociation attacks were performed on any AP or AD-HOC network.

IMPACT: There were also a number of ad-hoc networks from un-patched SI / Client machines

[6] Wired Equivalency Privacy – a Wireless Encryption algorithm vulnerable to a range of attacks.

and potential MiTM (Man in the Middle) style APswith suspicious SSIDs[7]such as "*Free Public Wi-Fi*"[8].Still for the purposes of PCI compliance, *the Client*'s LAN was not to be deemed a public network and unencrypted communication of sensitive data was allowed internally across wired infrastructure only.

This is not to say that this requirement won't be an issue at the next site that is surveyed. At a previous government department they had adopted the approach of locking down the LAN so that workstations could not connect out to the Internet and become infected or have sensitive information compromised. However when developers complained of not being able to do their jobs without the use of Google, newsgroups and other collaboration tools the department agreed to make wireless available so that they could use their laptops for internet access.

In this case the security provisions put in place included a 128 bit WEP key (though stronger than 40 bit, still largely vulnerable to Weak IV and collision attacks) and MAC (Media Access Control) Address filtering. The MAC ACLs(Access Control Lists) proved difficult to maintain as system administrators were frequently inundated with requests for access when development teams ramped up. Additionally this form of access control was vulnerable to MAC spoofing and on one occasion a developer was unable to connect as they're MAC Address was already in use.

IMPLEMENTATION CHALLENGE:When considering this requirement an auditor should also take into account the environment that they are in. While this government department was in a relatively isolated area (surrounded primarily by roads, parklands and shopping centers – as opposed to Starbucks, libraries and internet cafes) some sites may be less fortunate. During a security review of a medical research facility, I was told by the IT Administrator that they were a low risk site because there were only client machines within 100m of the AP. When I pointed out that a major national university with budding computer science departments was less than 500m away he was surprised to have this considered as a threat. At this point it is worth mentioning that recent Wireless Distance demonstrations[9] at BlackHat 2005 showed successful connection (maintained for over 3 hours) to a wireless access point from 125miles away, using an unamplified 802.11b AP at 11Mbps!

For the most part, infrastructure at *the Client* was hosted by SP and their internal networks were relatively locked down with detailed security policy enforced and audited. Whenever possible cardholder data was encrypted during transmission and if SSL (Secure Sockets Layer) communication between web servers was an option it would be enforced regardless of the network in use. Generally SSLv2 was part of the SOE (Standard Operating Environment) and would employ at least 128-bit keys for the SSL tunnels and certificates. Similarly for batch transfers or administrative tasks, all remote connections to servers containing sensitive data were over SSH encrypted tunnels, via SSH terminal sessions or SFTP / SCP transfers using SSH-2 based on the SHA-1 (Secure Hash) algorithm.

Remote access to non-production infrastructure was permitted over public networks using IPSec (Internet Protocol Security)VPN's that enforced multi-factor (Pin and RSA SecureID Token) authentication and TLS encryption.

[7] Service Set Identifier – the name of the Wireless Network or Access Point
[8] Ironically this seems to be a hoax AP / Virus that is propagated by the vulnerability in XP SP1
See Techblog by Dwight Silverman http://blogs.chron.com/techblog/archives/2006/09/free_public_wif.html
[9] Recorded on Wifiworldrecord.com with details at http://www.wifiworldrecord.com/2005writeup.html

Recommendation

If you don't deploy an easy to use wireless solution, your users will! The solution implemented at *the Client* was highly secure but relatively under-utilized as users were unable to configure their workstations to use the WPA2/802.1x protected access points. Additionally the hidden SSIDsand lack of communication regarding wireless access led many users to believe that Wi-Fi was not supported at *the Client*.

Also be sure to define what is known in Data Loss Prevention language as the sensitive data universe. Ultimately this is a classification of every item of data owned by the client which provides taxonomy for what is secret / sensitive data and what is public. The most frequent trap here is that a client will define the majority of their data (from CEO's Social Security Number through to their product list) as being Top Secret, be sure to validate any such classifications with a rigorous methodology.

3. Maintain a Vulnerability Management Program

3.1. Requirement 5: Use and regularly update anti-virus software

INTERPRETATION: Though *the Client* did their best to restrict the software introduced to their network (including withholding Administrator access from user PCs), infection was a constant threat with an enterprise of this size. This is well understood by the Payment Card Industry and this subset of requirements aims to address the management of vulnerabilities and related intrusion.

IMPACT: From an enterprise perspective vulnerability management was reasonably proactive, *the Client* had active implementations of SNORT running on Red Hat boxes that were monitored and patched by the *Client's* system administrators. These administrators were found to be subscribers to a range of Security and vulnerability mailing lists such as Bugtraq and made it their jobs to stay up to date on the latest exploits for their platforms and infrastructure.

Additionally hardware based IPSs (Intrusion Prevention Systems) from Cisco were deployed sitting in line with the network scanning all incoming packets for malicious traffic. These IPSswould generate in excess of 1 million events per day and would need to cope with throughput of and above 5 Gbit/s (at the time of writing, a new IPS solution was being investigated as the Cisco devices were capable of a maximum 2 Gbit/s throughput). The events would then be forwarded to the national response centre where they would be analysedby a team of approximately 10 people who would apply rules to prioritise, correlate andaddress each event according to their severity.

Additionally all Internet bound traffic and in fact any external DNS (Domain Name System) request would be routed through one of *the Client*'s 4 main proxy servers. These servers would perform deep packet inspection, blocking unapproved traffic such as those with MSN Messenger tags in the header or P2P (Peer to Peer) session initiation packets.

IMPLEMENTATION CHALLENGE:There was little physical access control over the devices that could be connected to their LAN (no endpoint authentication such as 802.1x, ironically used for wireless access but not tethered). Users could bring in machines from home as could the SI or SP employees. Client, SI and SP administered machines all shipped with a version of Symantec Anti-virus which forced auto-updates whenever connected; however there was little governance over these machines in terms of how they were used outside the office or whether they were re-imaged privately.

It is worth noting however that this, like many other network restrictions, was easily bypassed by users with any one of a plethora of technologies including:

- Redirecting the traffic over another port (such as IRC[10] packets over Port 80)
- Running a web proxy like squidso all traffic traveled over HTTP/S[11]
- Tunneling their traffic over SSH (making it impossible for the proxy servers to read the encrypted channel) using Putty or OpenSSH clients with port forwarding
- Configuring local SOCKS servers on 1080 (which remained open for legacy application compatibility)

[10] Internet Relay Chat

[11] Hypertext Transfer Protocol / Secure using Port 80 / 443

- And finally (in the case of MSN) just using an alternative client like Trillian that wrote different information to the packet headers!

The proxy servers would interrogate all clients scanning for open ports by pinging the router's broadcast address (X.Y.Z.255). If these ping requests were ignored, as suggested by some personal firewalls, the proxy would drop all outbound requests from the client until they responded – so the user effectively lost Internet access unless they participated in the port scan.

Recommendation

Do some research into whether your IPS product is actually doing Intrusion Prevention. Products like the Hardware IPSsfrom Sourcefire will sit inline with your network (by not obtaining their own IP Address from DHCP they can't be a target themselves from Denial of Service or 'DoS attacks'), do real time traffic analysis to determine what traffic is actually being passed and identify the endpoints on the network to better aid in understanding the network environment and ultimately deploying HIDS (Host Based Intrusion Detection System). Moreover an IPS of this nature should be able to perform network segmentation and authorization, so certain MAC Addresses will be allowed/denied access to a given network segment.

The reason this recommendation asks for you to look into the IPS product in use, is that this is a term that has been misused since its inception. When the idea of Intrusion Prevention and the term IPS was coined, IDS (Intrusion Detection System) vendors were quick to label themselves as doing Intrusion Prevention. Similarly Anti-virus, firewall and even patch management vendors would profess to be stopping intrusion and hence market 'IPS' solutions. A solid IPS should perform detailed network analysis with up to 10Gbit/s throughput (depending on the price tag) but more importantly be able to prioritiseand categorize events into a manageable series of alerts for an administrator to deal with.

Finally, regain control of your user's machines by not only removing Administrator access if it's not needed but auditing for the presence of enterprise firewall, anti-virus and anti-spyware / malware / phishing tools. One way to achieve this is using hardware based SSL VPN's such as those produced by Juniper Networks, which will accept connections from the client machine if they're fully patched and refuse connections if not. Assuming users were then required to use VPN in their daily job (either to work from home or perform a sensitive business function like submitting time and expense reports) there would be little option other than to maintain the enterprise sanctioned image.

3.2. Requirement 6: Develop and maintain secure systems and applications

INTERPRETATION: While sounding vague and ambiguous this requirement addresses an important aspect of any custom or packaged development project, specifically the need for secure coding standards. Much of the coding for this project was performed offshore in one of the SI's delivery centers. These centers were highly regulated with controls, governance and procedural rigor with a requirement that all operational units are compliant to the highest standard in this area – CMMI (Capability Maturity Model Integration) Level 5.

IMPACT: Still, though all development went through rigorous change control procedures including nightly baselining and release packaging there was little emphasis on secure coding and vulnerability awareness. As part of this effort to move towards a more compliant solution,

random code reviews were undertaken and vulnerabilities were identified to the development team leads. These included looking for vulnerability to SQL injection, buffer overflow and cross site scripting attacks and primarily centered on input validation for externally facing systems.

IMPLEMENTATION CHALLENGE:While physical distance can be a limitation,ultimately if standards are enforced offshore development is in principle not different from onshore development. In this case the offshore team connected via IPSec VPN over a private leased linewhich was further secured by restricting access to a particular VLAN. This extranet style access can be seen as simply an extension of the local development model.

3.2.1. Code Reviews

INTERPRETATION: Essentially the code review process involved contacting the respective application development leads for links to the 'checked-in' source code in the Document repository. There was a manual process of looking through the code repositories to find any externally facing web applications and interfaces or alternatively any additional libraries that may be invoked by these applications.

IMPACT: Once a pool of code was established to focus on, a random sample was taken from each team's work and allocated to one of the two Code Reviewers (both Security Architects with development experience on the project, working for the SI).

The input processing routines would be assessed first for length validation and stack protection to secure against buffer overflows. Next the data type validation would be checked for alphanumeric input restrictions in order to prevent SQL injection[12] attacks.

Finally any hyperlinks or referencing within the web applications along with the input validation routines would be verified to ensure only absolute links were pointed to and that any relative paths weren't susceptible to XSS (Cross Site Scripting) exploits.

If the input validation routines relied on helper classes or services, additional methods or invoked any libraries during run time then this code would similarly be reviewed for the aforementioned exploits and input validation.

IMPLEMENTATION CHALLENGE:As discussed, this approach of reviewing code for vulnerabilities was augmented by meticulous change control procedures and configuration management software. There is always a danger that patching a piece of code can introduce new vulnerabilities or that the patched module will break existing functionality and simply be rolled back. As such the first step in undertaking this requirement is to ensure the version control and configuration management processes are upheld by the developers.

3.2.2. Patching and Updates

INTERPRETATION: Patch management was a constant struggle with the application and environment owners.

[12] This was largely an additional layer of defence – as mentioned previously the Oracle Database that was used would perform statement isolation routines so that only one SQL statement could be executed at any one time (preventing SQL Injection attacks)

IMPACT:While the vendors were reasonably forthcoming with new patches and releases, application teams were reluctant to jeopardize their stable baselines and would often prefer 'the Ostrich approach'.

IMPLEMENTATION CHALLENGE Though there was a large scale Rational implementation in place for all version control, issue management and baselining there was no focus on a centralisedsolution or even approach to patch management. Each application patch or update would be weighed on it's merit and often packaged into a later release due to the threat of unknown impacts to stable baselines and the associated regression testing.

Recommendation

Secure coding guidelines and patch management are two tenants of security that are particularly hard to retrofit. In both cases the education of users and business owners is key to their success. For secure code, developers must be aware of the importance of validating their input and protecting their stack. For patching, environment owners should be measured on flexibility of their infrastructure in addition to the availability – if the only metric for CIO performance is uptime…. patching won't stand a chance.

With regards to version control and release management, there is often too much focus on the technology used. Whether it's a million dollar Rational implementation, legacy CVS systems or an open source install of Subversion; users will determine your data quality. Sophisticated change control can be subverted by simple process and whatever infrastructure is in place may go to waste. Once again, education is key – spend some time working with your users to establish procedures that work for them.

4. Implementing Strong Access Control Measures

4.1. Requirement 7: Restrict access to cardholder data by business need-to-know

INTERPRETATION: This requirement is easily defined, rarely enforced and always very subjective.

IMPACT:While the security principles of least privilege and need to know were always espoused at the Client many a situation arose where the need-to-know classification was dubious.

IMPLEMENTATIONCHALLENGE:Another layer of complexity lay in whether the need-to-know driver was enough to trump sound security principles. In one instance previously cited a firewall rule was created to allow direct access to a Production database via TOAD (Tool for Oracle Application Development[13]). The premise for this security violation was that the business people needed this information to do their job and this would be the simplest way to obtain it.

Recommendation

Sometimes *need-to-know*isn't enough. Least privilege should be the driver for establishing and implementing security policy andsecurity personnel should always be looking for the most secure way to meet the business needs. In the case discussed previously, the TOAD access was replaced by a custom GUI written to utilize a 3rd party ODBC (Open Database Connectivity) driver that employed SSLv2 to provide an encrypted and end-point authenticated tunnel into the database so the hole in the firewall could be closed.

The basis for all firewall rules (personal and enterprise) should be *"DENY ALL unless explicitly allowed"*. Open-ended rules and pattern matching should not be used if it can be avoided. A system administrator should consolidate their rule sets so that they can block all traffic disallowed by their non-functional requirements – no more and no less, with the minimum number of rules possible. As mentioned earlier, having 200,000 firewall rules will not only slow your network throughput but also raise the likelihood of a conflict.

It is important to ensure however that the other extreme is not used either. A previous client of ours (and hosted service provider) proudly told us that in their SELinuxIPTables definition they had less than 10 rules. When we reviewed the list, one of the chains effectively stated IP <MYROUTER> ANY and another IP ANY <MYROUTER>. Simplicity and elegance should be strived for but not at the expense of your defenses.

[13] 3rd party client for Oracle databases that uses an unencrypted ODBC link over port 1521

4.2. Requirement 8: Assign a unique ID to each person with computer access

INTERPRETATION: Though this seems like a fairly straightforward requirement the implementation of this principle proved to be more complex. As mentioned previously at another client, user accounts were openly and publicly shared in order to obtain the necessary privileges (such as Internet access). So while each person with computer access may have a unique ID, it may not necessarily be in use.

IMPACT: At *the Client,* UserIDs were provisioned upon receipt of an approved access request form and would be granted the privileges that had been similarly approved. However, accounts were usually created with default passwords and so when left inactive could often act as dormant backdoors into the system. Moreover while the password policy seemed inherently complex it was vulnerable to dictionary attacks and usually allowed relatively weak passwords.

For instance the policy was implemented as follows

Password must contain a minimum of 8 characters

Password must contain a mixture of upper case and lower case

Password must not be the same as 8 previous passwords

Password must contain 1 non-alphabetic character

So while this would seem fairly standard for password policy, the following weak passwords were permitted (and identified during audit within 60 seconds).

Password1

Welcome1

<CLIENTNAME>1

Banana77

Gunner88

For remote access the client would allow users to connect via IPSec VPN with TLS encryption from any network including Public wireless such as available in Starbucks. The VPN security policy enforced multi-factor authentication through the use of an RSA SecureID token and a 4 digit PIN, with connections required to be made using the Nortel Contivity client (though this was more likely a limitation of the switchesrather than an additional layer of access control). Additionally the client would not permit split tunneling, so a single authenticated user session would be restricted to a single MAC Address.

Remote connections would be terminated after 15 minutes of idle activity but there was no lockout threshold (due to the number of users reattempting connections with mis-configured clients). Conversely LAN sessions would not timeout after a level of inactivity but had a 3 attempt lockout limit, where 3 incorrect attempts to logon would force a 15 minute lockout from that account.

Since *the Client* had implemented a 'Single Sign On'-style Identity Management System, locked accounts could be reset using preconfigured secret questions and answers or by a users 2-up manager (their supervisor's supervisor). This account reset and unlock facility proved to save

the company a fortune in lost time and password-reset infrastructure. A study from Gartner[14] pointed to Password Reset as the number one cost for technical support in the enterprise with an estimated cost ofover $30 per reset, based on the lost productivity of all involved and the cost of maintaining additional support staff and infrastructure for this purpose.

At the previously mentioned government department, help desk and support services would take as many 3,000 password reset calls in a week!

IMPLEMENTATION CHALLENGE:The use of clear text communication was prohibited at the *Client* for external authentication but internally; unencrypted protocols like FTP were initially encouraged by Client administrators for use within trusted zones. The justification being that it would put less demand on the infrastructure and result in better performance. However, since protocols like FTP and Telnet would authenticate in the clear, passwords were sent unencrypted for application accounts (eg: batch servers doing internal data migration) and this requirement would be violated. Furthermore as a Security architect trying to educate users and harden applications across the enterprise, we had worked hard to eradicate clear text communication and treat such protocols as vulnerabilities. To start opening up Ports 21, 23 and even 25 seemed to run contrary to the intentions of the compliance effort.

Recommendation

There is a definite linear (and perhaps exponential) relationship between the number of passwords that a user is required to use in their daily job and the security that is requiredfor generating and protecting these passwords. If a password is too complex or must be changed too frequently and may be one of many passwords a user has to remember, the likelihood of it being written downor the account being locked out will start to increase.

Though the implementation can be expensive, Identity Management solutions (at least for provisioning/deprovisioning of users) can be a great value proposition. Often products like Sun Identity and Access Manager will go a long way towards compliance with this requirement. These tools (once installed and configured for your environment) can effectively authenticate users, authorize/deny access to a resource and allow them to administer their own account through secret questions. If there are multiple applications requiring authentication within the business, these products can be configured to pass the identity of the user from one authentication service to the next and even federate that identity to 3rd parties (where permitted).

On a previous project in Europe, an infrastructure company required users to login daily to up to 11 systems all with distinct passwords and different expiration. As you can imagine the service provider doing the account resets for this client had a very secure revenue stream. The use of an enterprise identity in this case could really cut costs and aid the users in their job.

A successful implementation of this style ofSSO was run in the author's own firm, with their enterprise user account and credentials federated among the various websites and online data stores to which they'd be required to authenticate. If the user only needs to login once, chances are they'll remember their password and can afford to make it a good one.

Instead of focusing on password complexity rules try and emphasize passphrases notpasswords. A minimum length of more than 12 characters will provide a much larger key space when brute forcing passwords than a series of high complexity rules. For instance with 1 Upper case character, 1 number and 1 symbol with a length of 8 characters (assuming password originality and not based on a dictionary word) this would provide a key space of

[14] Gartner Study ID G00123531: http://mediaproducts.gartner.com/reprints/passlogix/150863.html

(26 + 26 + 10 + 10) ^ 8 which is equal to 722204136308736 (7.22 * 10^14)

Whereas an all lowercase passphrase of 12 characters would provide a key space of

26^12 approximately 9.54 * 10^16 or 132 times larger than the previous example. Now which password would be easier for an average user to remember

P@ssw0rd (common deviations of password are far too often used)

OR

thisismypassword (16 characters providing an even larger key space)

Additionally using passphrases will eradicate the threat of attack using a pre-computed key space as found in Rainbow tables with tools like Rainbowcrack or Winrtgen.

4.3. Requirement 9: Restrict physical access to cardholder data

INTERPRETATION: This requirement covers physical security and access control.

IMPACT:Unfortunately in the case of *the Client* much of the physical maintenance and access control was treated as a black box to us.

IMPLEMENTATION CHALLENGE:Working for the SI at the client site, we had little visibility of the SP's physical controls and even less involvement in the decision making process. In this case we were forced to shift the burden of compliance back onto SP and the client's own Security officers to enforce regulations around entry to facilities, physical identification, visitor logs, access to and destruction of storage media used for backups.

Recommendation

Only the Paranoid Survive by Andrew S. Grove should be mandatory reading if you work actively in the Security field. On a previous project we had used the same SP for our EBR solution and had on many occasion requested confirmation of the successful completion of our hot SAN (Storage Area Network) backups from the SP'ssysadmins. In this case we had a Development environment hosted on a Sun E25K frame with Solaris zones - semi-virtualized servers, each Operating System running in a virtual container mounted at /Zones with direct access to the hardware.

2 days before the deadline given to verify the backups on the SAN disk, there was an incident of corruption when writing to the MSS (Multiprotocol Switched Services) disks over MPXIO (Multiplexed Input Output – a feature within Solaris that allows a single interface to write out to multiple devices as with dual hot backups) and all data on the disks was lost. The reason I describe the architecture in detail is to point out that due to the virtualized environment, not only was uncommitted development code lost but also the installed/configured application suite and any Operating System configurations!

As luck would have it, this was in fact the first run by the SP of the hot backup technology (on this particular project) and effectively, over night, the Development environment was gone. My recommendation for all future projects is to maintain strict control over all storage infrastructure and physically verify successful backup completion.

One should also beware of physical threats to any storage media or infrastructure. At a recent security review of a Research Institution, the data centre appeared on paper to be very securely

and tightly controlled. They had locked down implementations of SELinux and IP Tables configured according to the NSA Secure templates[15]with a strictly enforced patch regime that included subscription to update services and sandbox environments for testing patches before they were applied. However, when doing a survey of the site I was astonished to find that some of the server rack space was in fact occupied by jars of various liquids and other biological matter. When questioned on this, the server administrator had informed me that they had run short of space in the laboratory fridges and this was the only place they could store the cooled specimens!

Some of the most significant data loss, particularly of sensitive data has in the past come from failure to protect the physical media, including:

JP Morgan Backup Tape Lost (SANS NewsBites - Volume: IX, Issue: 36)

J.P. Morgan Chase probing data breach shown in YouTube video

Another often-neglected area is print media. While many organisations now have clean desk policies, adherence to this policy can waver due to a lack of regular enforcement. Similarly there will often be secure trash bins available but then unsecure trash adjacent to them thereby reintroducing the risk of secure documents being left out in the open. One of the biggest offenders in this area is the print room, where users will print a sensitive document and forget about it or unknowingly print it twice only to have the 2nd copy sitting idly and unprotected due to lengthy print queues or delays in the caching. Controlling access to sensitive data in this format can sometimes be just as difficult or time consuming as implementing encryption solutions.

[15] See NSA Central Security Service: Security Enhanced Linux http://www.nsa.gov/selinux

5. Regularly Monitor and Test Networks

5.1. Requirement 10: Track and monitor all access to network resources and cardholder data

INTERPRETATION: As discussed later in the paper, this is a requirement that is often covered off with a packaged software solution andat *the Client* it was no exception.

IMPACT AND IMPLEMENTATION CHALLENGE:A lot of the detail from our experiences at the client here will be explored later in the chapter entitled “The Silver Bullet” though in order to protect the privacy of the SP and the software owners of that particular product, the names have been obfuscated.

Recommendation

While there are a lot of good products out there that will assist in capturing effective and accurate logs it is important to understand the different levels of logging. This may range from high level alarm-only access logging which may flag events like attempts to modify syslog.conf right through to *process accounting* style logging which will interrogate system daemons and provide detailed information on the resources utilized by even the most harmless of processes. As with the sensitive data universe, make sure the right log levels are set for the right level of sensitivity of the environment and infrastructure.

It is commonly accepted that different users will have their own set of rights and privileges as well as resources they can / cannot access, so why then don’t we implement individual or group logging levels? A problem faced at *the Client* was that policies were implemented across all User IDs so that system accounts were tracked to the same degree as non-technical users. In this case the result was that the logging levels were turned down across the board but a better solution would be to define groups and the respective level of logging required.

Similarly it should go without saying that processes like *inetd* and *xinetd*(*‘superservers’* or process launchers) would need to be watched more closely then say *xeyes* (an X11 add-on that follows the mouse cursor around with a pair of eyes on screen). The same is also true for directories, where the system administrator configuring the audit logging application should specify log rules based on the binaries and configuration files that are accessed rather than the directories that contain them. At the *Client*, specifying directories like /var, /sbin and /etc was a shortcut that was used to circumvent appropriate policy design though ultimately this wasn’t viable due to the volumes of logging generated. Suffice to say that most environments are better off having *‘level 4’* log detail on processes with superuser privileges than *‘level 3’* log detail on all processes.

Much of these design decisions will feed into the audit policy, which depending on the product will be key to how much visibility you have before, during and after a security incident. The payoffs areoften felt after a breach has happened when detailed audit and logging can identify both the intruder and their method of exploitation using digital forensics tools.

Log Server Infrastructure

Another direct impact of a policy implementation is the infrastructure that will be required to support the audit and logging. This may involve storage (local and external such as Tape, NAS or SAN backup), processing power, memory and network capacity planning. With detailed logging over the network to SAN or NAS (Network Addressed Storage) there may be material impacts to the network interfaces and local network segments used to transport the log data.

When building a log server, availability is crucial – if an attacker can perform a DoS attack on

your logging infrastructure then generally they'll be uninhibited in their attacks (safely knowing you'll have no record of their movements). Conversely an attacker should not be able to use the logging infrastructure against you, for example in a DoS attack by filling up the logs or congesting the network with excess log traffic. In order to maximize availability there are 3 principles that should be taken into account:

1. Redundancy
2. Redundancy
3. Redundancy

It may seem overkill but dual NIC (Network Interface Card) failure is more common than we'd like to believe and SAN storage is not sublime in any way. At a previous client there was an incidence of massive data loss without any kind of system or hardware failure. An intern that had been posted to the data centre decided to clean up around the place by trimming off the excess 2 cm of tape protruding from an array of magnetic tape drives. The 'excess' tape was actually the header information for those volumes and the tapes were rendered unusable!

That said logging should be performed over the network (to an offsite location) and locally with a regular *diff* comparisonperformed againstthe two logs. It may seem simple but when those logs don't match there's a high likelihood of an incident having occurred. Depending on business requirements the logs should then be retained in two locations for X number of years (often 7 years in Australia / New Zealand but can be longer for medical and other records).

In order to maintain the value of the logs any records, especially automated, should be digitally signed and certified by a respected CA (Certificate Authority) - potentially the company or service providers. The authority of the signing party and the verifiability of the signature will underlie the level of non-repudiation. In the event of a breach, particularly one involving an internal user or legal proceedings the ability to repudiate a user or transaction is critical and the integrity of the logs will be relied upon.

Alarming and Monitoring

More important for protecting your data than logging itself is alarming and monitoring. Generally after-the-fact logging will be useful in court and for eDiscovery / forensics but system administrators will benefit more from knowing when an attack is happening – while it is actually happening. Setting appropriate alarms and triggers on protected resources can alert a sysadmin to an incident while it is still in progress and they will be empowered to take more effective action than an audit tool could ever do. If need be, sysadmins should have the ability to press a panic button of sorts that allows for emergency shutdown of various components when under extreme duress. When defining the relevant resources to be alarmed, work with the business to understand what is sensitive but also work with the technical architecture and infrastructure management teams to determine what could be exploited. These will be the resources for which any use should be monitored and where appropriate, alarmed on.

Lastly it may seem trivial but since logs are predicated on timing and timestamps, synchronization of all system clocks is key to having accurate logging and non-repudiated evidence. All servers should be running synchronized versions of NTP (Network Time Protocol), which must be patched for vulnerabilities (as even a system clock can be exploited).

5.2. Requirement 11: Regularly test security systems and processes

INTERPRETATION: This is frequently interpreted as penetration testing which is usually well understood and easily embraced by executive level management but care must be taken to fully understand this requirement in the context of a specific environment. At the *Client* for instance, the SP would run automated vulnerability scans against the web servers looking for directory traversal, buffer overflow, cross site scripting and other vulnerabilities.

IMPACT: Being deeply involved in 3 separate packaged software implementations the *Client* was no stranger to testing. At some points there were more test environments and test phases than there are PCI requirements, from: Application Test, through to System Integration Test, Stress and Volume Test and finally Operational Readiness Test. When it came to Security testing however the *Client* was a lot more reticent.

Site surveys are often the best way to identify users subverting security policy with modems, non-standard hardware and wireless access points. A physical walkthrough will give the most accurate insight into the effect IT Controls and governance are having at a company as well as an understanding of their vulnerabilities. In the case of the *Client* we would regularly test authentication schemes (including password audits and cracking) along with their wireless infrastructure. These tests were the most common because:

1. They were easy to perform, requiring minimal sign off – being non-intrusive and

2. They were very effective at highlighting in an easily palatable way the backdoors into the business.

Site surveys were performed by the Client's Security and Privacy officers and usually involved a physical walkthrough of the floor on a midweek morning and then again after hours during the week. Often this would be accompanied with a wireless survey and would involve looking over user workspaces (selected at random) for violations of policy, specifically intentional violations such as wireless routers or modems brought on to the project site unofficially.

IMPLEMENTATION CHALLENGES: The effort required to test security technology and processes can be wrapped into other forms of technical testing like Operation Readiness Testing (ORT) or Technical Verification Testing (TVT). Usually this will be a hard sell as

A) so much effort has already been expended on testing application integration and functionality and

B) the significantly political overtones of exposing flaws in a product, implementation or business process.

Recommendation

Scanning for vulnerabilities is one of the more well established Security tests that can be done across an environment and in terms of identifying areas for improvement it can usually yield some of the best results. Open source tools like NMap and Nessus can simplify this process and automate the results to make enterprise level network scanning a regular activity.

One great recommendation from SANS is to run NMapagainst your system from the outside and then run netstat –a from the inside and diff the results. If there are more ports exposed internally than externally then you may have a polymorphic Trojan that is leaving a backdoor

open for it's owner. If there are more ports externally than internally then you may have a rootkit that is hiding available ports from you using it's own netstat binary. Either way, the results should match and a difference will tell you something is wrong.

Though not open source, file integrity monitoring tools like Tripwire can be very useful if there are stable OS (Operating System) baselines in use. When there are a number of files on the system that should remain static, be they configuration files, binaries or user data, Tripwire style alarming is a solid precaution. The file integrity monitoring tool should be notifying the sysadmins (via email, sms or other real time alerts) whenever the MD5/Blowfish hash on one such file has changed and can be set to replace the file automatically with a locally stored copy. In the case of Web servers this can be particularly useful by reloading crucial HTML pages from non-writable media such as CD-ROM in the event of a defacement.

When it comes to Penetration Testing there is a lot of stigma involved and it is worth spending some time with management to assure them that nothing will be irrevocably damaged and no individuals will be exposed. Additionally help them to realize this will be done in a controlled environment by professionals with the same levels of governance applied to another cycle of testing. We found it very difficult to shake the impression of penetration testers being a group of hackers "running rampant on the systems".

Conversely clients with bold claims about uptime or 'unbreakability' should be wary of the fact that Penetration testing is often one of the last activities in hardening an environment. It is important that they cover off the majority of their Security requirements first, before embarking on an effort to break them – it is no coincidence that this standard is second to last.

While deploying Honeypots and devising penetration tests for an environment are some of the more exciting aspects of a sysadmin's job, they are far from the least urgent. Be sure to verify a sound business need for engaging in these somewhat risky activities.

6. Maintain an Information Security Policy

6.1. Requirement 12: Maintain a policy that addresses information security

INTERPRETATION: It's been said time and again that the information security policy is the cornerstone of any secure business environment. There are only certain circumstances under which this will remain true however. At many a client surveyed, the security policy is dusted off whenever there's been a breach or a business requirement to break with sound principles.

IMPACT: On a number of occasions at the *Client*, the information security policies were used to look for ways to allow/deny a request that just didn't *smell* right. For example, while technically there was nothing to say that test environments shouldn't connect directly to production, architects and managers alike knew this was a bad idea.

IMPLEMENTATION CHALLENGES:Drafting and circulating a Security policy is unlikely to be the primary challenge here, insteadawareness of and adherence to policy are key factors for success. Security policy should be understandable by each and every employee to which it applies in order for it to be used as a template for acceptable / unacceptable user actions.

Recommendation

In order for the policy to be an effective pillar for security, it should be all encompassing but palatable. This is a document that should cover any such situation unequivocally and still reach a broad audience. The policy should lay down in simple terms - the guidelines under which the company will operate. If these guidelines are only used retrospectively when they've been broken, or interpreted in the context of a specific need then they're hardly a document worth laying your foundations on.

While half the battle is making the policy concise and relevant, the other half is having it disseminated. Not only should the policy be easily accessible and widely distributed but users must be able to apply it as needed to their daily jobs. An information security policy should cover items like P2P file sharing, social networking like Facebook and Meebo, Web 2.0 sites like eBay and YouTube, Bluetooth devices and Mobile internet access. Even more importantlyusers need to know these items are covered (and accordingly endorsed or prohibited). Ignorance cannot be an acceptable excuse for security violations.

Company procedures need to reflect not only the right way to perform a task but also the common practice way. There should not be an official and an unofficial way of doing something, so any Security policy will need to be pragmatic and take ease-of-use into account. Spend some time with the users and if the policy doesn't suit their business processes, either the process or the policy need to change. Additionally any time spent providing training and awareness for users will pay dividends in compliance -education will often be a significant obstacle. Training can range from informal, optional 'Brown-Bag' sessions over lunch time through to mandatory training schedules and will usually depend on the level of commitment from senior management. If the executives are supportive of the policy and stakeholders supportive of it's implementation then mandatory security training (performed online) should not be a hard sell.

When dealing with any 3rd parties or partners, system integrators, consultants or business process outsources, ensure that they too are compliant to your policies and standards. It's no use expending effort on PCI Compliance if your customer's credit card data is going to be stored in an unencrypted spreadsheet on a consultants laptop.

Finally when it comes to security procedures that need to be followed, design and test a detailed Incident Response plan ahead of time. In the event of a security incident there are likely to be enormous technical, temporal and legal pressures to do the right thing … and quickly. The only way to be prepared for this type of occurrence is rehearsal. Practice and refine the plan until it can be easily followed under pressure and make sure it is stored in hard copy with a 'jump bag' or emergency store of necessary supplies including:

External USB Drives, CDs / DVDs, Pens / paper, batteries, torches, LAN cables, mobile phone and other essential equipment specific to your environment. Most importantly have a clearly delineated chain of command, highlighting who is responsible for leading a response effort and who needs to be engaged at which point.

7. A very tempting Loophole: Compensating Controls

Education is a criticalfactor for success in any enterprise wide security endeavour and particularly one involving compliance. At the Client we were constantly faced with users trying to circumvent security policy and IT Governance. Some of the more senior users were even great advocates for tighter controls and regulatory compliance….. just not for themselves.

Whenever there was a shortcut that could be taken (saving time, money or resources) it was likely to be well supported, leaving it up to the 'purist' architects to argue the merits of sound architectural solutions. PCI DSS was no exception and as is true elsewhere, a little bit of knowledge was a dangerous thing.

For the uninitiated the PCI DSS Standard includes a loophole for being required to implement security measures, recorded as follows in the final appendix of the standard.

"For companies unable to render cardholder data unreadable (for example, by encryption) due to technical constraints or business limitations, compensating controls may be considered."

Essentially for the client this meant there was a rather tempting loophole that allowed them to shortcut the encryption route and keep their sensitive financial data stored in clear text for ease of use. However it is worth noting that these *Compensating Controls* must be accepted as sufficient by the PCI auditors and then approved as a solution by the company's financial institution, they should not simply be a loophole for any hardening that is 'too hard' to do.

Still according to the PCI DSS, therecommended compensating controls are as follows (with a short explanation of how they would be met at the *Client*):

1. Provide additional segmentation/abstraction (for example, at the network-layer)

This was easily accomplished through the multi-tiered network design in which the database servers storing the credit card data were logically and physically zoned off in a different network tier, providing segmentation and abstraction from direct access

2. Provide ability to restrict access to cardholder data or databasesbased on the following criteria:

- IP address/Mac address
- Application/service
- User accounts/groups
- Data type (packet filtering)

IP Address filtering was done as only the pre-registered privately assigned IP Addresses of the application servers were allowed connections through the firewall at the database tier in order to access the database server. This design was implemented by the SP in their hosted environments and afforded strong access control, to their credit.

Application based access control was in some places implemented by the applications, as connections would only be accepted from approved services. However in the case of the middleware ESB, a number of the services were exposed internally and could be connected to or consumed by any other service without authentication or authorization.

Clearly (as discussed previously) there were tight controls around the user accounts (official application and system accounts only) and groups that could connect to the database servers.

Data type was a slightly greyer area in that since the hosted environment was somewhat 'black box' to the client; there was little visibility of the filtering being performed. It would be fair to say though that the SP were using a tiered firewall design with in line IPSs that would perform a degree of packet based filtering on all data coming down the wire.

3. Restrict logical access to the database -
Control logical access to the database independent of Active Directory or LDAP

This requirement was interpreted in a number of different ways according to the environment to which it was being applied. Network Administrators would claim that access to the Database was controlled independently of LDAP by the network segmentation rules. OS Admins would profess the built in Unix Access Controls were logically controlling access and firewall / IPS vendors would similarly tout their products as fulfilling this requirement. With this level of ambiguity it was very easy to claim that the 'logical access to the database' was in one way or another restricted.

4. Prevent/detect common application or database attacks (for example, SQL injection).

While there were still significant gaps in the hardening of application code with respect to preventing exploits, these attacks were protected against using hardware and software technologies such as:

Intrusion Prevention Systems – Cisco &McAffee IPS boxes would monitor network traffic and identify malformed SQL when sent across the line.

Firewalls – Cisco PIX Firewalls could identify incoming SQL injection attacks before they reached the web server.

Native Database Protection – Oracle 9i and above includes built in SQL Injection triggers that will trap and discard any unexpected SQL statements sent via the Web Server[16]. Specifically these triggers will perform statement isolationso that only single SQL statements can be executed at any one time. For instance if the expected input is

USERID: <username>

PASSWORD: <password>

and the input validation routine is:

SELECT USERID FROM USERS_TABLE WHERE USERID=$USER AND PASSWORD=$PASSWORD;

Then in a typical SQL Injection attack the hacker would put together the following input

USERID: John

PASSWORD: ""; DROP SCHEMA; COMMIT;

[16] This is configured using dbms_assert in 9i (part of the PL/SQL DBMS_OBFUSCATION toolkit) or in later releases using Oracle Application Express: Session State Protection

In this case if no SQL injection protection is provided the input validation routine retrieves no records but then also executes the second and third statements with the same privileges causing massive data loss and a DoS.

However with patched versions of Oracle 9i and above the SQL Injection triggers will isolate any SQL statements executed by the input validation routine so that no matter what, only the first statement is executed against the database.

Whether they were aware of the PCI DSS Appendix or not, throughout this entire exercise a multitude of users would assert the argument supported by Compensating Controls. Statements like the following were found on any email trail regarding a PCI Encryption issue:

"Can't we just secure the logs"?

"The data doesn't need to be encrypted, it's locked away in the database"

"A hacker couldn't access those servers anyway"

"Do we really need another layer of security"

and most dangerously … *"this is not required according to the standard, we can comply by just implementing security elsewhere".*

8. The Silver Bullet – from SP

One of the biggest obstacles to PCI Compliance was education and misinformation – hence our initial focus on education. In their effort to address the PCI requirements once and for all *the Client* had started looking for a silver bullet solution for all their compliance needs and the vendors and service providers were waiting in the wings.

In this case SP had been managing all *the Client*'s infrastructure including storage, backup, maintenance and operation of all server / network components. Given they had previously been a full service provider for *the Client*, it made sense that they would offer their wares as a solution to 'make PCI go away'.

The product that was sold was in itself quite powerful offering policy enforcement and audit logging at each layer of the stack. It could provide agent based functionality from detailed operating system logging right through to authorize/deny decisions for access requests to individual resources.

However, an initial review with *the Client* and SMEs from the SI however determined that the MS Vista style allow/deny policing would be too intrusive to be inserted in between the existing applications and the operating system. Instead *the Client* decided on an implementation of this product (hereafter referred to as SYSPOL), which would only include the OS logging functionality.

Once this approach was decided on, we conducted a PCI DSS Fit / Gap assessment to determine just how many requirements could be met with this tool in its current form. After sitting down with the various stakeholders to review the PCI standard we developed a requirements traceability matrix that linked each level 3 requirement (eg: Req. 5.1.1)to it's owner (the responsible party: either SI, SP or Client) and assessed the Client's current position with a colour coded status of

Green: Compliant to this requirement

Orange: Not currently compliant to this requirement but with plans to address any outstanding items in place OR compliant to this requirement in some areas with remaining areas identified and targeted.

Red: Not compliant to this requirement with no plans to address compliance.

One of the aims of this requirement gathering exercise had been to identify how much of PCI would be addressed by the SYSPOL solution and whether it would be cost effective. It is worth mentioning that in order to implement the SYSPOL product additional cycles of SIT (Systems Integration Testing) and SVT (Stress and Volume Testing) would be required on *the Client*'s CRM and Billing applications to measure any adverse affects from the SYSPOL agents. The implications of this would mean thousands of man-days in testing effort. This would be augmented with the cost of the SYSPOL licences, policy design and additional infrastructure required to support the central server processing the data collected by the agents. When combined with the policy design workshops and review sessions, the SYSPOL implementation would be an exceedingly expensive exercise.

The extensivereview session identified only 2 out of over 200 requirements that would be addressed by the SYSPOL solution. One would think this would be enough to torpedo the project but the SP had sold this solution as a "silver bullet" to handle the 'PCI issue'. At one point they even titled their proposal documentation as

"SYSPOL – The PCI Compliance Solution"

As new people joined the project and came across this situation they would often struggle to understand how so much effort was being expended on compliance to these 2 requirements

“10.1 Establish a process for linking all access to system components (especially access done with administrative privileges such as root) to each individual user”

“10.3 Record at least the following audit trail entries for all system components for each event”

Meanwhile in other areas of the business, initial efforts were already under way to address issues like credit card data encryption, secure coding practices and mandatory access control but these largely competed for attention (and associated funding) while SYSPOL was highly visible and seemed to be everyone’s top priority, largely due to the size of the implementation task.

Additionally there were significant technical limitations of the SYSPOL implementation. For one thing – SP’s infrastructure could only support a maximum of 200 agents when the number of servers that needed monitoring was more than double this. Then the SYSPOL agents were configured to identify the issuing of illegitimate SQL statements by users but because the product’s shims were Operating System only, the most detail that was ever logged was that someone had connected to the database or run an Oracle binary.

For example a typical log entry would theoretically list:

John43: sudo su oracle

John43 AUTHENTICATED SUCCESSFULLY as Oracle

John43: cd /ora/bin

John43: sqlplus DBNAME /nolog

John43: exit

Now from these logs there would be no way to tell whether John43 was doing his job and looking for a system event by connecting to the database as Oracle and issued the following statement

SELECT * FROM EVENTS_TABLE WHERE ID=”MYRECORDID”

Or whether the John43 account had been compromised and the following statement was issued

UPDATE USERS WHERE ID LIKE “*ADMIN*” SET PASSWORD=”MYPASS”

So effectively a large amount of the log data was after-the-fact logging where the useful information was missed. Additionally since *the Client* had agreed to be responsible for reporting on and monitoring this data (SP would only log the events and forward the logs), this would be tantamount to asking someone to watch your back and having them come to you after an attack saying *“your back was stabbed”*.

There are a number of products around that can address this problem in different ways: Oracle 10g implements Fine Grained Auditing that will capture this information for specific users based on predefined policies. Alternatively 3rd party solutions like SQL Guard from Guardiumcan sit on a database server’s network switch watching the span port to monitor the SQL statements that are actually sent to the database.

The sheer volume of data that was captured was yet another limitation. Since the decision to implement the SYSPOL tool was made with the proviso that it would be stood up ASAP, there was little consultation with the application streams or platform architects to identify the binaries and configuration files that would need to be monitored. As such, SP deployed to the Sandbox environment, agents with a Default 'watch-all' policy that would effectively monitor access by all user accounts (including application accounts) to some of the following directories.

/

/var

/opt

/bin

/etc

As you may have guessed, with our Sandbox environment (only a drop in the ocean compared to the behemoth that was Production) the SYSPOL tool was generating over 10Mb / minute of logging which effectively rolled up to around 3 – 4 TB / week for Production. Since these records would need to be kept for up to 7 years it was determined that we would need several Petabytes of SAN storage just to maintain the logs or (according to "How much data is that?"[17] - approximatelyequivalent to the volume of all US Academic research libraries!) Needless to say these policies were fine tuned, though not through workshops and design sessions but instead by eliminating some of the directories listed above.

With the reduced policies the logging volumes were down to more manageable levels but SP had still not agreed to do any alarming and monitoring. Given that this was already after the fact logging and could end up missing crucial steps, one would assume there would have to at least be specific triggers on the logs themselves. Instead, the *Client* would end up having large amounts of spurious logs forwarded to them for analysis with no form of event prioritization or even identification.

An additional scope limitation was that with the reduced policies only certain binaries were monitored so for instance tailing syslog or using *cat*or*more* on /etc/* was not allowed by SYSPOL but there were no rules against using *head, less*or*awk.*

Recommendation

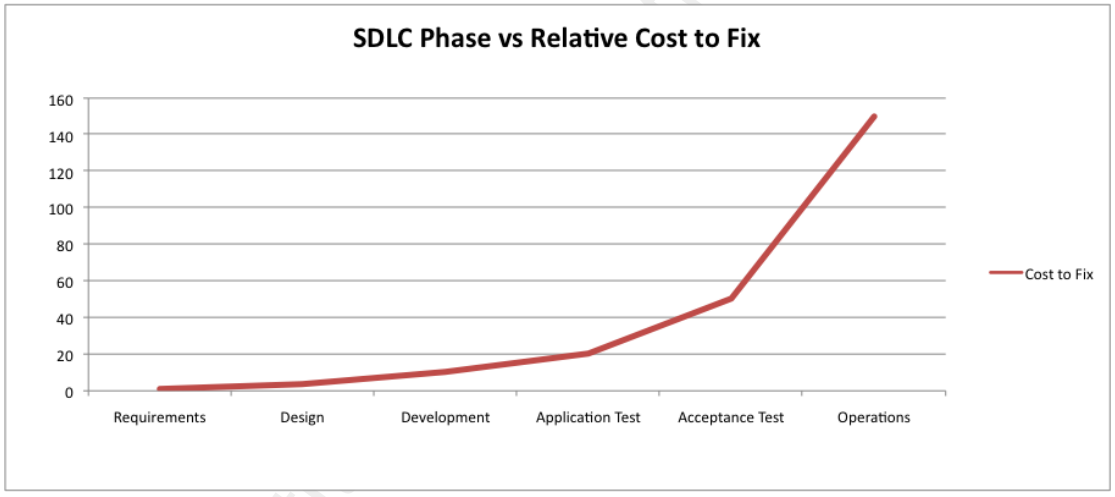
Since the inception of PCI DSS a number of vendors and service providers have been queuing up at large clients like this to sell their product as the *Silver Bullet* for compliance, the one-stop-shop that would make the requirements "go away". As is true elsewhere, there is no quick fix for PCI compliance. The PCI Requirements are a set of rules that have been designed to harden your enterprise throughout for which there is no shortcut. My recommendation would be to treat any vendor's product as addressing a single requirement of PCI only and while software like SYSPOL may span multiple Data Security Standards; it is only one layer in your strategy of Defense in Depth.

[17] *How much data is that? By James S. Huggins* http://www.jamesshuggins.com/h/tek1/how_big.htm

9. The Cost of Retro-fitting Security

It has been found that an error found in an implementation will magnify exponentially in effort required to correct it[18], as that error is allowed to flow through to each subsequent phase of the SDLC. So for example an error identified during the development phase requiring 2 man days of effort to rectify will then cost around 20 days to be fixed during testing and 200 days should it be allowed to go into production.

It is important to note that the review of the Client's architecture and push towards compliance came about after the first release of their enterprise applications had gone into production. This implementation was no exception to the mantra above and applying fundamental security principles proved a very costly exercise, particularly as the Security architects would be ironically constricted by the governance and controls they were implementing: i.e. a batch architecture configuration file could not simply be modified in production to remove a default password, instead the system administrators would need to go through the appropriate channels to apply the fix as per any other configuration change or code drop.



Perhaps the most important take away from this paper is that PCI compliance and indeed the majority of securing and hardening activities will be best achieved by being incorporated as early as possible. In the diagram above, the increasing cost of fixes to applications or infrastructure is shown, highlighting the need to understand requirements such as PCI from the very beginning.

If you can incorporate security and compliance best practice into your designs on any project, the investment will pay dividends when that project is deployed and the fixes required to production are minimal.

[18] *Managing Information Technology* by Martin Wainright, © Prentice Hall 2000 ISBN: 013064636

10. The Audit

At the time of writing the *Client* was still over 6 months away from their first PCI Audit. They had originally been scoped for a first round review to identify the gaps however the auditors (from a major Accounting Firm) had determined that the newly built and deployed CRM / Billing applications would not meet the minimum volumetric requirements. That is, the PCI Compliance audit was to be performed on the Legacy systems only, as the CRM / Billing applications only held pilot data at that time.

The decision to do this was understandable from the Client's perspective as it would mean the burden of complying would be largely delayed until the auditors returned for the second round review of the legacy systems (which would at that time be approaching retirement). However the implications of this approach would be significant. Shortly after the auditors would finish reviewing the legacy systems, the full scope of migration activities would begin and the full production data set (all remaining customers beyond the pilot) would be moved across to the new environment.

Aside from having only the legacy systems as compliant, a major problem with this approach was that any additional encryption functionality or application hardening would be required to be applied to the data set while in production. For instance if existing data in the new Billing application would need to be encrypted / obfuscated, the routines would have to either run over the existing systems while they were live or take a significant outage of the new applications to perform a migration activity where sensitive data would be exported out and imported back in. For anyone that's spent time with IT Executives, one thing they are never supportive of isoutages. If your company's CIO is measured based on the uptime that they provide to the business, it would be prudent to ensure that releases and migration activities are correctly packaged so that any tasks requiring an outage are performed at the same time.

Still, the effort to move towards compliant infrastructure and applications continued nonetheless and received various waves of support and resistance from each department.

Recommendation

The results of a PCI audit are often the most significant motivator for a lot of management as the ramifications of a negative result including any associated reputational damage will often be highly visible. Use this driver to your advantage as much as possible without appearing to be propagating fear. No one will listen to the 'Chicken Little' style Security Architect or System Administrator. One needs to be reasonable about the challenges that are faced and the implications of not meeting them but more importantly management need to be aware of these challenges and understand they will be held accountable for their decisions.

Finally

A recurring theme in this paper is that there are no shortcuts to compliance. There are no silver bullets, and the PCI requirements will not 'go away'. Each business needs to own their compliance effort and once they understand the challenge ahead, a key sponsor or stakeholder must be identified.

Something that inhibited our progress was the piecemeal nature of compliance efforts, where small groups would work on hardening individual applications, environments or areas of infrastructure often not tied to a particular requirement.

The good news is that compliance can be well targeted with a single, focused effort to with multiple streams working in parallel on the various requirements. As part of this, stakeholder(s) with ultimate responsibility and considerable authority needs to be appointed so that once convinced, the business owner can sign off on a particular approach.

The PCI DSS is not the Holy Grail in compliance and controls nor is it the last step in hardening your infrastructure. What it represents, as with a number of other forms of governanceIn fact, it is probably only the first in a series of international standards that eventually will cover also areas such as Governance, Privacy and more.

What PCI DSS has given us though – whether you are in the Payments industry or not - is a framework for best practices in IT Security. This framework should be aspired to, applied and in the long term adhered to but ultimately is only a series of layers in your strategy of Defense in Depth.

Glossary

ACRONYM	MEANING
3DES	Triple DES
ACL	Access Control List
ADSL	Asynchronous Digital Subscribers Line
AP	Access Point
API	Application Programming Interface
CA	Certificate Authority
CEO	Chief Executive Officer
CIO	Chief Information Officer
CMMI	Capability Maturity Model Integration
CRM	Customer Relationship Management
CSR	Customer Service Representative
DBMS	Database Management System
DES	Data Encryption Standard
DHCP	Dynamic Host Control Protocol
DNS	Domain Name System
DoS	Denial of Service
DSS	Data Security Standard
E25K	Sun Hardware Platform with pluggable frames for vertical scaling.
EAI	Enterprise Application Integration
EBR	Enterprise Backup and Recovery
EDW	Enterprise Data Warehouse
ESB	Enterprise Service Bus
FTP	File Transfer Protocol
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
I&AM	Identity and Access Management
IDS	Intrusion Detection System
IP	Internet Protocol
IPS	Intrusion Prevention System
IPSec	IP Security
IRC	Internet Relay Chat
IVR	Interactive Voice Response
KDC	Key Distribution Centre
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
MAC	Media Access Control
MiTM	Man In the Middle
MPXIO	Multiplexed Input Output
MS-CHAP	Microsoft Challenge Handshake Authentication Protocol
MS-DOS	Microsoft Disk Operating System
MSN	Microsoft Network
SP	Service Provider
MSS	Multiprotocol Switched Services
NIC	Network Interface Card

NAS	Network Addressed Storage
NSA	National Security Agency
NTP	Network Time Protocol
ODBC	Open Database Connectivity
OOTB	Out of the Box
ORT	Operational Readiness Testing
OS	Operating System
P2P	Peer 2 Peer
PAN	Personal Account Number
PCI	Payment Cards Industry
PEAP	Protected Extensible Authentication Protocol
PKI	Public Key Infrastructure
PL/SQL	Procedural Language / Structured Query Language
RO	Read Only
RSA	Rivest Shamir Adelman: popular implementation of PKI
SAN	Storage Area Network
SANS	SysAdmin Audit Network and Security
SCP	Secure Copy (SFTP fallback protocol)
SFTP	Secure File Transfer Protocol
SHA-1	Secure Hash Algorithm
SI	Systems Integrator
SIT	Systems Integration Test
SME	Subject Matter Expert
SOA	Service Oriented Architecture
SOCKS	Abbreviation of SOCKETS – a client server protocol for using firewall services
SOE	Standard Operating Environment
SQL	Structured Query Language
SSH	Secure Shell
SSID	Service Set Identifier
SSL	Secure Sockets Layer
SVT	Stress and Volume Test
TDE	Transparent Data Encryption
TLS	Transport Layer Security
TOAD	Tool for Oracle Application Developers
TVT	Technical Verification Testing
UDP	**Universal Datagram Protocol**
VPN	Virtual Private Network
WEP	Wired Equivalency Privacy
Wi-Fi	Wireless Fidelity
WPA	Wi-Fi Protected Access
X11	X Windows System (Windows GUI for Unix shell)
XSS	Cross Site Scripting