Smartphone Threats and Open Source Firewalls

Harpreet Singh Malhi

Faculty of Engineering

University of Alberta

Table of Contents

1. Abstract

The smartphones are getting popular in the business environments. These are no longer devices just for the phone use. With increasing processing power, these are like small computers always connected to corporate network by a variety of different methods. Usually, the organizations have a solid defence and security policies in place for regular workstations and servers but smartphones are ignored. Microsoft ActiveSync and Blackberry Enterprise Server are two leading technologies for smartphones to access the corporate data. On the other hand, the security solutions like IPS and firewalls are considered an extra cost to IT budget. There is a need for small to medium organizations to look into the open source solutions. Vyatta provides one such platform capable of delivering IPS/firewall/web proxy/VPN in a cost effective way. The Vyatta can be installed as a VM in today's mostly virtualized network infrastructures and is a viable option for small to medium size organizations as compared to expensive and proprietary solutions offered by Cisco and Juniper.

In this project I have studied the threats posed to the corporate security by smartphones so that the C-level management can understand the risks to business environment. A high level comparison of Microsoft ActiveSync and Blackberry Enterprise Server in the terms of security is also done. The second part of project is dedicated to the study of Vyatta platform as a firewall/web proxy/VPN solution in corporate environments. Overall, this project will provide significant understanding of the security threats by smartphone use in the corporate networks and a high level implementation of Vyatta solutions. I have used Running Room Canada's network as an example to approach these from a real world scenario.

2. Acknowledgement

I would like to thank Dr. M. H. MacGregor for giving me an opportunity to work on this project. A special thanks goes to Mr. Shahnawaz Mir for providing Vyatta training resources. In the end, I am grateful to my previous employer Running Room Canada for the use of their test environment and allowing me to map the project scope to their IT requirements.

3. Introduction

The smartphones nowadays are offering as much processing power as regular PCs were a few years ago. This has resulted in the increased use of smart phones like Blackberries, iPhones and Android devices in corporate world. The mass adoption of both consumer and managed mobile devices in the enterprise has definitely helped to increase employee productivity but has also exposed the enterprise to new risks.

In the PC world, a corporate can standardize on a single platform. But in the mobile world, it must support different types of devices including various versions of RIM Blackberries, Apple iPhones and Google's Android phones. The challenge is allowing productivity improvements while mitigating the risk of numerous types of devices with each requiring a different security policy due to different risk factors. Another major issue is that smartphones as mobile devices are lot easier to get lost compared to the laptops. The cost of the information stored on the device in this case will be much higher than the cost of device itself. With laptops and PCs, it is easier to draw a line what is owned by the employee and what is owned by the organization. However, the accessibility of corporate data with smartphones on the employee owned devices is getting increasingly popular and the organization will still be liable for the data lost.

A mobile security solution to mitigate these risks need to take into consideration; what types of applications will be used, what kind of data will be stored on the device as well as what regulations an organisation is required to support like Health Insurance Portability and Accountability Act (HIPAA) as some organisations are required to be complaint with these information security laws. The following diagram shows various common threats that could lead to data leakage from smartphones:-

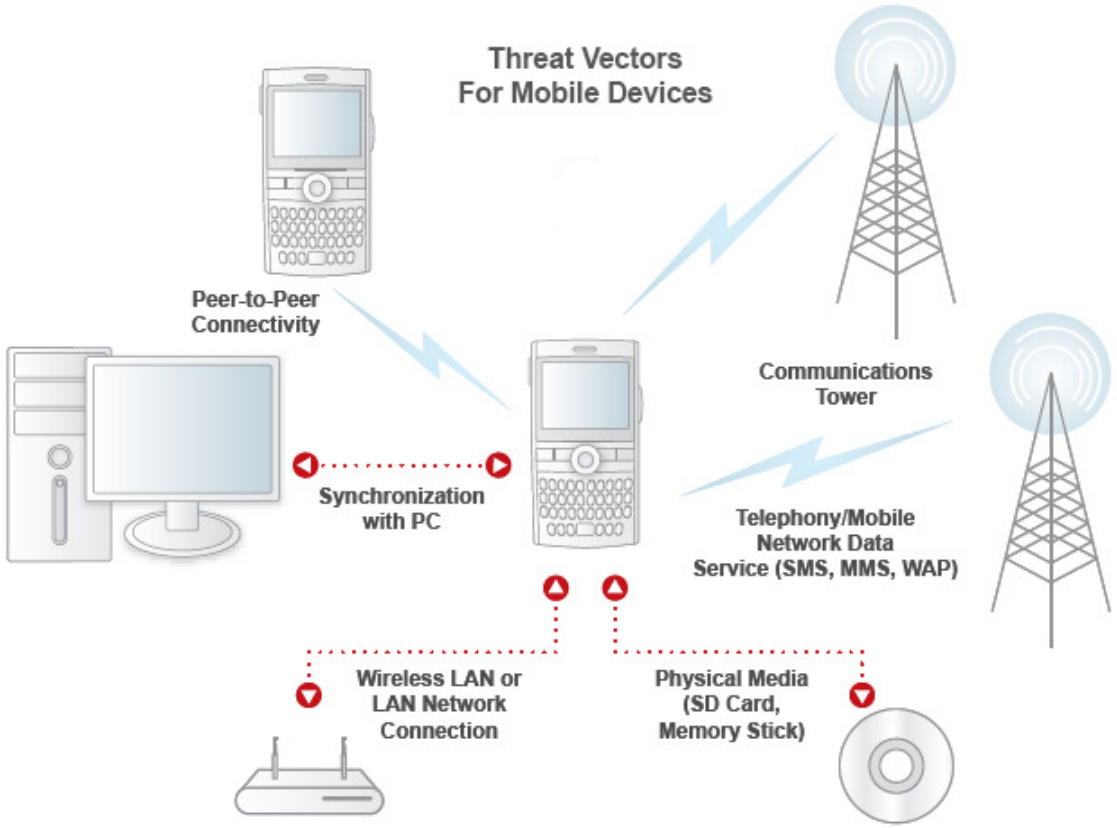

Figure 1: SmartPhone Threat Vectors (Trend Micro, 2007)

This whole picture is getting more interesting with the wave of cloud computing. Today's smartphones are connected to an entire ecosystem of supporting cloud and PC-based services. Many corporate employees synchronize their devices with at least one public cloud based service that is outside of the administrator's control and also use their home computer to back up key device settings and data. In both scenarios, the key enterprise assets may be stored in any number of insecure locations outside the direct governance of the enterprise. When properly deployed, smartphones allow users to simultaneously synchronize their devices with multiple cloud services both private and enterprise without risking data exposure between the two. In a normal mobile solution, an employee will connect his device together to both an enterprise cloud service like Microsoft Exchange, Blackberry Enterprise server for his work emails and calendar as well to a private cloud service like Gmail, MobileMe to get his private emails. Smartphones generally

present a consolidated view of data from both these sources into one seamless user interface while the device maintains an internal layer of separation which ensures data is not moved across two services accidently i.e. Microsoft Exchange Server data won't go to MobileMe and vice versa. However, it is quite easy for end users to use third party tools and services to intentionally or unintentionally exposing the enterprise data to public clouds and unmanaged computers during backups. Lets example this from three different common scenarios:-

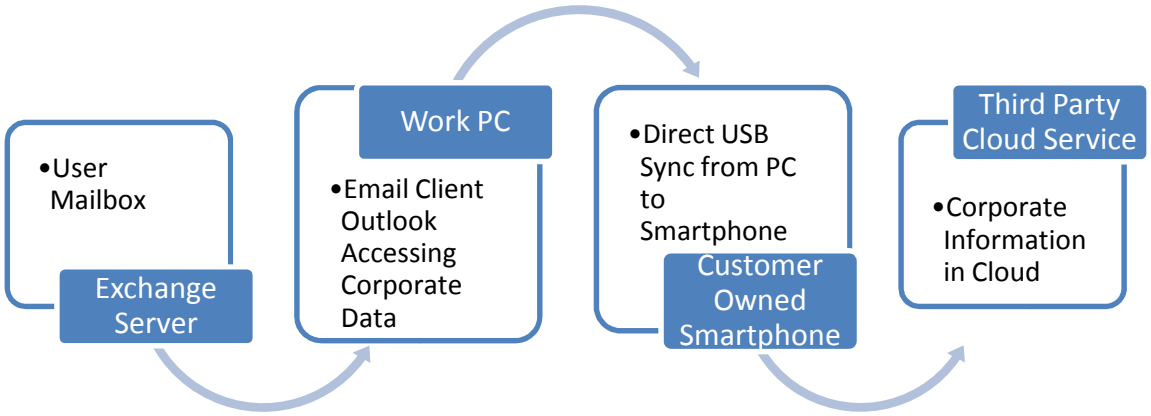

Figure 2: Scenario 1 – Unsanctioned Desktop Sync

In the first scenario, an employee brings his own phone to the work which is not managed by the corporate. He simply connects to his computer to synchronize his work calendar and contacts using iTunes or another third party software. This is much easier to do on Mac OS platforms due to the lack of enterprise restrictions like Group Policy Objects. The Macs are recently getting popular as the workstations in the corporate world especially with c-level executives. However, there is lack of centralized enterprise administration tools for Macs. Once, the user synchronizes data from his work computer to the smartphone and then, he decides to enrol in a third party

cloud service like Gmail , which will pull off all the enterprise data from the smartphone, to the public cloud which don't have enterprise level governance or protection policy.

In the second scenario, an employee uses a tool like Google Calendar Sync or iTunes on his work computer to synchronize the work data directly with a private cloud service like MobileMe or Gmail. This will automatically move the data from Microsoft Outlook or iCal to a third party cloud.

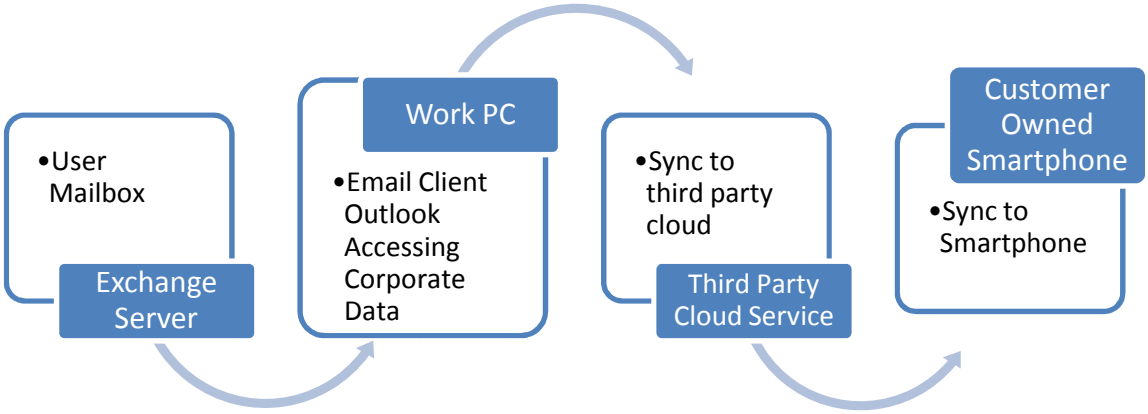

Figure 3: Scenario 2 – Unsanctioned Enterprise Desktop to Cloud Sync

Then, the user uses his own smartphone to synchronize directly with the private cloud service causing his device to download all corporate data. Thus, employee is able to get their work data on their own phone without having it to connect to Microsoft Exchange or Blackberry Enterprise server directly.

In the third scenario, the users typically copy music and multimedia files from their home computers to smartphones using a variety of software like iTunes, Blackberry Desktop Manager and some third party tools in case of Android. By default, these software make a backup of device data even if the user opt out not to synchronize the calendars or the contacts. This backup

will typically contain the entire calendar data, notes and address book along with the device settings. This may not be in the encrypted form. As an example, iTunes does not encrypt this by the default. This leads to enterprise data on the user computer without them knowing it which is again a security risk.

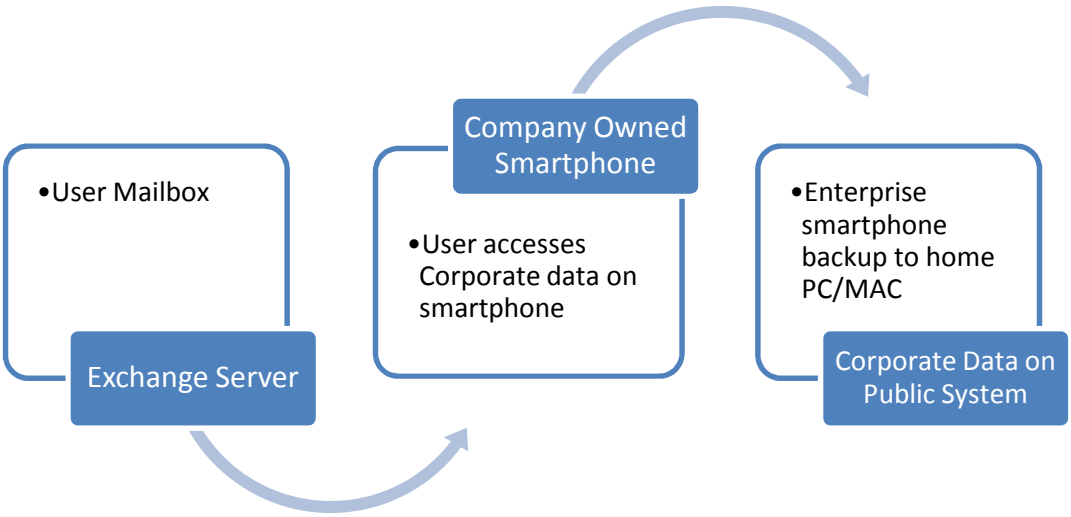

Figure 4: Scenario 3 – Unsanctioned Enterprise Device Sync with Home PC

4. Components of Mobile Security Solution

The security of any enterprise mobile solution can be looked into using following three components. These components are as following-

1) The device Security deals with securing the data on the device and device itself. It deals with the protection from the malware, the malicious applications, the social engineering attacks and the device OS architecture itself. We will look into three popular platforms Apple iOS, Google Android and Blackberry OS.

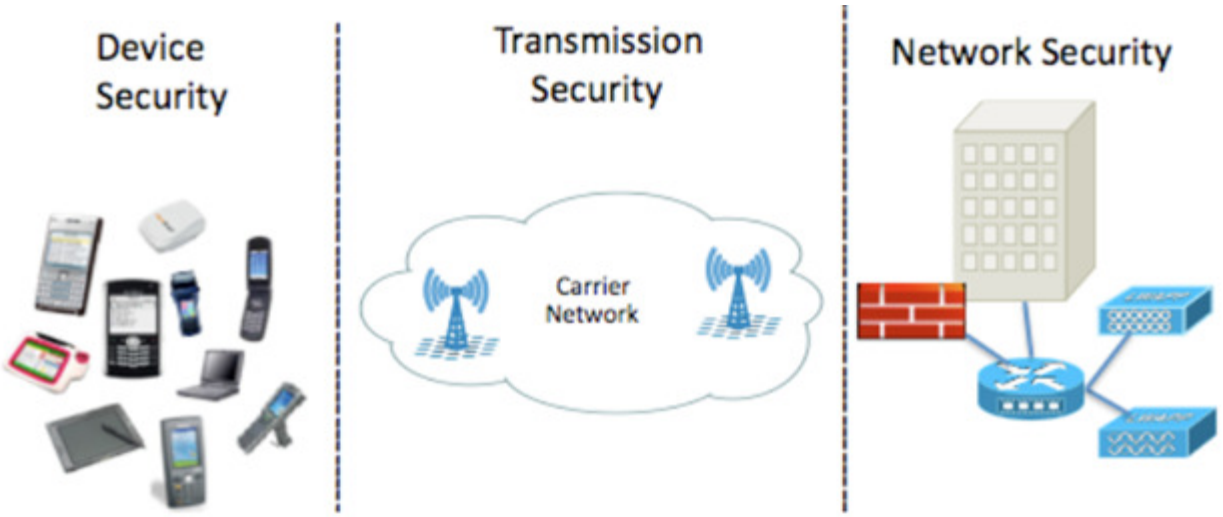

Figure 5: Three Components of Smartphone Security (Trend Micro, 2007)

2) The transmission security deals with the security of data as it is transferred over the wireless network from the device to the enterprise network. It should ensure a balance in the CIA triangle. This topic deals with various encryption protocols used like 3DES, AES and SSL. I will compare the transmission security of two enterprise mobile solution Blackberry Enterprise Solution and Microsoft ActiveSync later on.

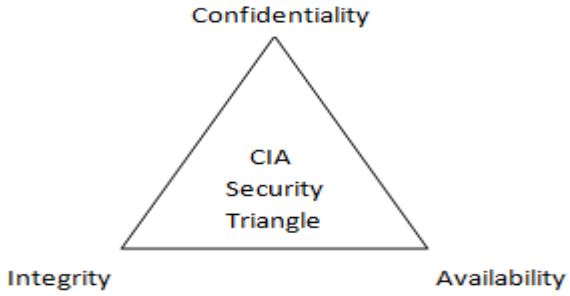

Figure 6: Information Security Triangle

3) The network security deals with the security of enterprise network required to support smartphone solutions. I will look into the network infrastructure needed to support Blackberry and Microsoft ActiveSync solutions. This will involve the authentication

methods supported for these platforms and best practices when designing corporate network to support these smartphone solutions.

5. Device Security

The mobile device OS has very little in common with the traditional desktop peers when it comes to OS architecture even though Apple iOS and Android platform are based on Mac OS X and Linux respectively. The desktop OS typically rely on corporate restrictions and third party security tools like antivirus and anti-malware protection. Due to OS's small image size and processing power limitations, the mobile OS reply on the security models designed into their core implementation. The major threats to the device security can be seen from the following diagram:-

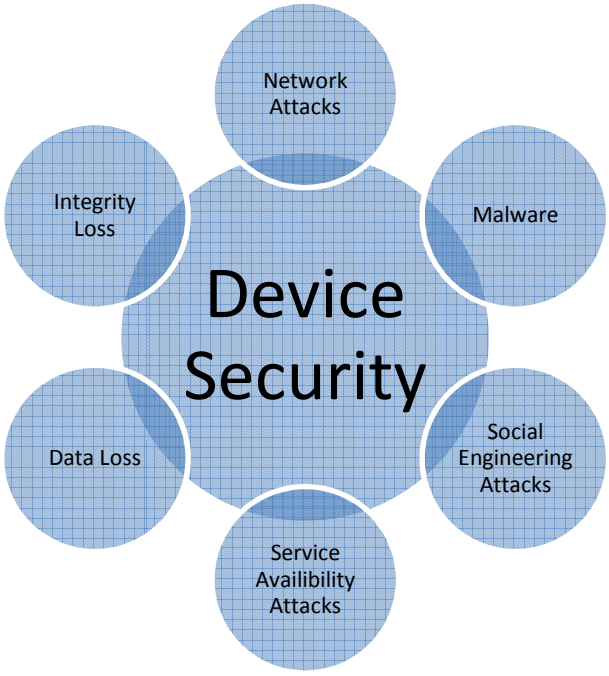

Figure 7: Device Security Threats

Let's look at these threats briefly before we look into the mitigation techniques for these.

- **Malware Attacks**: - Malware can be briefly divided into three categories: viruses, worms and Trojan horse programs. The viruses attach themselves to valid applications. The worms spread from one device to another over the network. The Trojan horse programs do not self replicate but compromise CIA triangle and use device resources for the malicious purpose. Malware attacks are more common in the desktop and server arena but the mobile devices are getting attacked starting recently. Some of the well known attacks are Storm8, iPhoneOS.lkee, Android.Rootcager and Android.Bgserv.
- **Network Attacks**: - The network based attacks on mobile devices are typically launched via malicious web pages. The mobile users surfs the compromised page which causes the browser to run the attacker's code. This is usually accomplished via sending down a specially crafted set of malicious data to the browser. The attack is successful if the mobile device is running a potentially vulnerable browser version. Once the attacker has gained the control of browser, they have access to user's surfing history and stored passwords. They can also access to other parts of device depending on the application isolation model employed in the OS.
- **Integrity Loss Attacks**: - In this type of attack, the attacker tries to corrupt or modify the data on the device without the user's permission for financial again or to disrupt enterprise operations.
- **Data Loss Attacks**: - This may happen if the device gets stolen and has sensitive information on it without any sort of encryption. Another scenario will be when a user synchronizes their device to a public cloud service like Gmail, MobileMe which leads to unintentional data loss.

- **Service Availability Attacks**: - In this type of attack, the attacker misuses computing, identity and network resources of the device for malicious purposes. The two common scenarios are to send spam emails from compromised devices and launch DOS attacks against the wireless carrier or corporate network supporting the mobile device.
- **Social Engineering Attacks**: - These types of attacks trick the end user to disclose the sensitive information. The most common way is via email phishing and luring the user to install malware by using a legitimate looking application.

6. Device Security Model

The mobile device security largely depends on the security model used. The security model used by iOS, Android and Blackberry OS can be thought of based on these five security pillars.

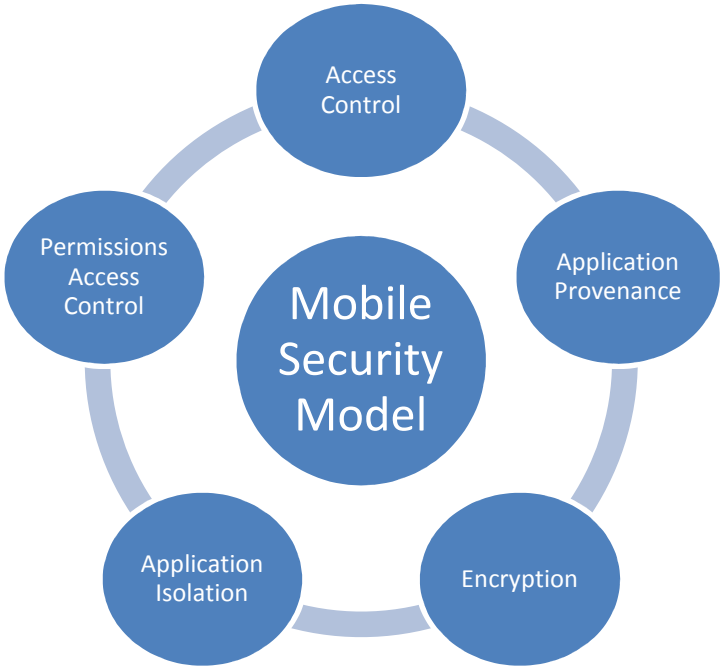

Figure 8: - Mobile Security Model

Let's look into these five pillars before we dive into details of how these are implemented in different smartphone OS.

1) **Access Control: -** It deals with protecting the device with the use of passwords, PIN code, face reorganisation and pattern drawing. The screen ideal-time lockout is another factor here. This access control is usually the first line of defence against the device security. Mathematically a four-digit PIN only needs 10,000 brute-force attempts to crack the PIN. However, many devices have time delay functionality after ten failed attempts making the 9,990 attempts take much longer. The time delay won't prevent a successful brute-force attack. However, the caused delay should reach a point where the user, who has lost the phone, is able to notify appropriate authorities.
2) **Application Provenance: -** Provenance is simply a vetting process in order to provide users some level of assurance concerning the application. It associates the authorship and privileges to an application but should not have taken as a security measure for its code. The application is stamped with the identity of its author and then, its integrity is protected using a digital certificate.
3) **Encryptio**n: - The likelihood of losing a mobile phone far exceeds the possibility of losing a laptop and the corporate data, stored on the device, can provide a goldmine for any thief. Encryption aspect of security model deals with the protection of data at rest to address the data lost or theft. It deals with the full disk encryption, file encryption and email encryption.
4) **Application Isolation**:- This is also known as application sandboxing. This provides the security and stability. It limits an application's ability to access sensitive data on the device as well as limiting application's calls into core OS. Applications should also be

isolated from each other as each one of them could have access to the data with the different security sensitivity label.

5) **Permissions Access Control**:- Each application is granted a set of permissions at its install time which limits what it can do on the device OS as well as which areas of the data and systems are within its access scope. The applications are blocked by device OS if it tries to access something outside of its assigned permissions.

7. Apple iOS Security Model

Apple's iOS operating system powers iPod, iPhone and iPad devices. It is a slimmed down version of Apple's desktop OS Mac OS X. It is written in Objective-C and large part of Coca API. Therefore, it carries the security flaws traditionally associated with C programming such as buffer overflows, integer overflows and format string attacks. Apple iOS is mainly based on four out of the five Mobile Security Model pillars: access control, application provenance, encryption and application isolation.

Apple iOS Access Control

The iOS provides traditional access control options for passwords as well as account lockout options. It also provides the reasonable security in case of loss or theft. When an iPhone or iPad is used with Microsoft ActiveSync, the administrator can control password and device lockout options using Microsoft Exchange ActiveSync policies. The ability to remotely wipe the device is also imperative. If the direct push is enabled on Apple devices using Microsoft Exchange ActiveSync, the device can be remotely wiped out using Microsoft Exchange server and this feature combined with password access control on iOS, can be a good deterrent in the mean time while the user reports the stolen device to IT department.

The screenshot below is an example of basic access control options available for iPhone and iPad when enabled for direct push using Microsoft Exchange 2010 SP1:-

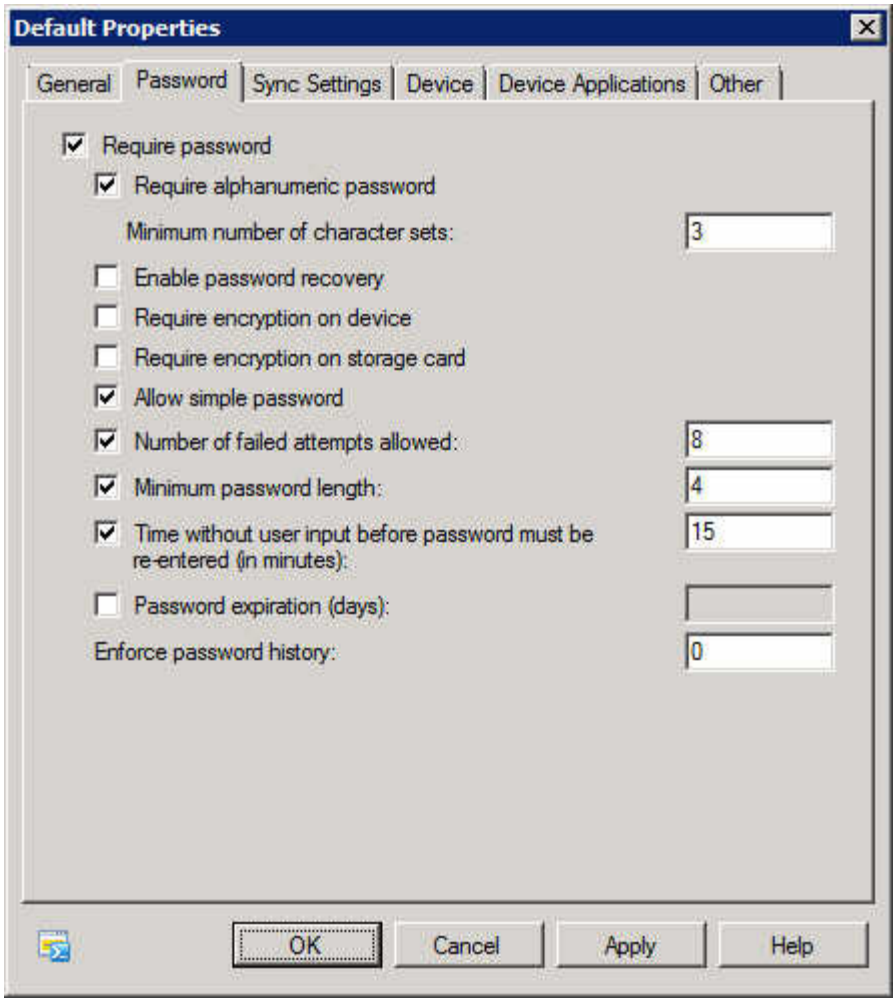

Figure 9: Microsoft Exchange 2010 SP1 ActiveSync Policies

Apple iOS Application Provenance

Apple's application provenance model is made up of iOS Developer and iOS Developer Enterprise programs. Before an application is released to end users, developers must go through a registration process with Apple by paying an annual fee. They are issued a digital certificate by Apple which is used to digitally sign each application. This signing process embeds the

developer's identity into the application which guarantees that the developer is approved by Apple and the application code has not been tempered with after its creation. There are two ways to distribute the application for iOS:-

1) The first way is to post the application on Apple's App Store. In order to do this, the developer first must submit the application for the certification process with Apple. Once, it is verified that it is posted to App Store by Apple.
2) The second way is when a corporation deploys privately-developed applications to their internal users via iOS Developer Enterprise program. To participate in this program, the applicant corporation needs to be certified by Apple for a clean track record. Once this certification process is done, the corporation can develop and sign the in-house applications and deploy these to Apple handheld devices. These applications can only be installed on devices with a digital certificate called "provisioning profile" installed. If this certificate is ever removed or expires, all in-house applications will stop working.

Apple's provenance model is quite effective. It has a strong deterrent affect but it's not fool proof. Firstly, it does not work against the jailbroken devices. The jailbroken devices can run applications from any source leading to potential security threats. Using Cydia and Installer are two most popular ways to install unauthorized applications. Once jailbroken, the handheld can run a SSH daemon to copy the unsigned applications to the device and these run even outside of application sandboxing used by Apple. Secondly, an attacker could use a stolen identity to register for an account to post malicious applications on App Store. Apple does have ability to issue a global certificate revocation for a corporation's provisioning profile that disables all in-house applications by a corporation immediately. However, when it comes to applications from

App store, they are yet to possess a mechanism to remove malicious installed applications from the devices.

Apple iOS Encryption

Apple devices running iOS version 4 or later uses two types of encryption. It uses AES-256 encryption using hardware acceleration to encrypt the flash memory of the device. Secondly, it uses a software based encryption to provide a second layer of encryption for emails and attachments. This implementation has a problem; iOS needs to keep the decryption keep around even when the device is locked to give background applications access to the device storage. This leads to a problem as the majority of data on the device can be decrypted without the user needing to provide mater passcode. Therefore, an attacker with the physical access to device with a jailbreak can access most of the storage. However, since emails and attachments are protected by a second layer of software encryption derived from the user passcode, this data is secure even if the attacker has physical access to the device.

The encryption protects against the loss of data if the device is stolen. Since the hardware based encryption is used to encryption the whole flash memory so if the device is lost, a kill signal can be sent by IT Dept using a centralized Mobile Device Management solution like Microsoft Exchange Management Console to throw away the hardware encryption key. The same can be setup via Active Sync device policies if someone enters a wrong passcode certain number of times.

Apple iOS Application Isolation

Apple devices offer the application isolation by a technique called sandboxing. It is used to create application partitions to prevent a malicious application from modifying the underlying

system or reading the data meant for the other applications. Each application is installed in its own directory identified by a GUID. The applications allow limited reading access to certain areas but do not allow access to read/write to other application's data in directories /private/var/mobile/Applications. The applications are also not allowed to access iOS operating system kernel directly and cannot install privileged drivers. If the device is running low on the memory, the applications cannot also designate themselves as the system critical to avoid termination by iOS. The applications are also isolated from the device's SMS, phone and email subsystems and this isolation prohibits initiating emails, calls and SMS without user participation. The below is the application isolation model used by iOS:-

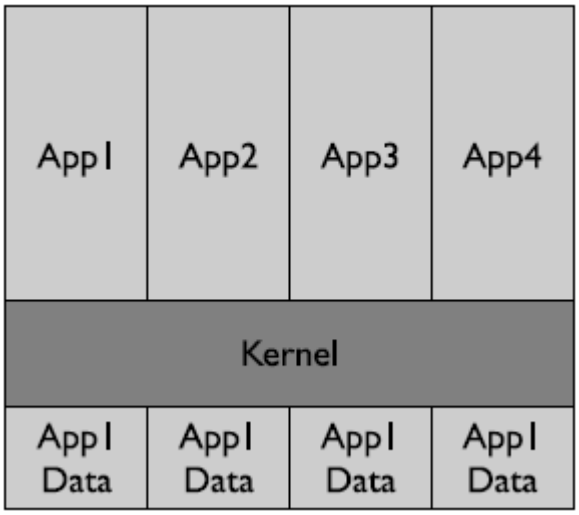

Figure 10: Apple iOS Application Isolation Model (Dwivedi Clark & Thiel, 2010)

iOS applications are allowed to freely access following system level resources without any explicit granting of permissions by the user:-

- Communicate to any computer over the wireless Internet.
- Access the device's address book including mailing addresses and notes associated with each contact.

- Access the device's calendar entries.
- Access the device's unique identifier (a proprietary ID issued to each device by Apple).
- Access the device's phone number.
- Access the recent safari search history.
- Access the device's microphone and video camera.
- Access the device's music/video files and its photo gallery.
- Access the Wi-Fi connection logs.
- Access items in the device's auto-completion history.

Let's see how effective this Application Isolation model has been in iOS. If an application is compromised, the attacker won't be able to attack other applications or the iOS kernel. The same is true if a malware affects one particular application. This offers a reasonable protection against browser based network attacks as even if the browser is compromised, the attack would not be able to spread to other parts of iOS due to this isolation policy. However, there are some risks here as well; the system wide resources such as contact list and photos can be accessed by the browser process due to iOS isolation policy. The information available or saved in the browser such as passwords, bookmarks and browsing history is also at risk as iOS allows communication to any host on the internet. This way information can escape to third parties within the restrictions of application isolation model.

The iOS's isolation model can prevent a subset of resource abuse attacks. On the negative side, iOS applications are given unrestricted access to the Internet, so technically they could be used to launch email-based spam campaigns, search engine optimization campaigns where an attacker tricks a search engine into raising the ranking/visibility of a particular website and some types of denial of service attacks against websites or the carrier's network.

On the positive side, given that iOS's isolation system prevents the automated transmission of SMS messages or automated initiation of phone calls, this eliminates the possibility of SMS based DoS attacks, telephony-based DoS attacks and SMS-based SPAM attacks on non-jailbroken devices.

The application isolation approach of iOS also gives some protection against the data loss and integrity loss attacks. Beyond the library of media files, the calendar and contact database which is accessible by each application, there is not a central repository of data. But, the users storing important information such as password for other systems, credit card numbers and security codes in forms of notes and calendar entries still pose some risk as this information is accessible to any application which can easily be transferred to a host on the internet. The applications are not allowed to modify or delete the user's media and photo libraries but they can modify or delete the calendar and contacts. However, these can be easily synced with corporate services using Microsoft Exchange if this information gets removed. This way integrity of data on the device is somewhat protected.

Apple iOS Permissions Access Control

The iOS has little permissions based access control for its applications. There are only four system resources that applications may access which first require explicit permissions from the user. The all other system resources is either allowed or blocked explicitly by iOS's built-in Application Isolation policy model. The four system resources are :-

- To access location data from the device's global positioning system.
- To receive remote notification alerts from the Internet.
- To initiate an outgoing phone call.

- To send an outgoing SMS or email message.

This approach does protect against the network and integrity loss attacks such as unwanted applications from tracking the user's location. It also ensures that devices resources are not abused for email or SMS based DOS attacks.

Overall, Apple iOS security model has been quite successful in preventing a variety of different attacks but it does not address social engineering attacks.

8. Android Platform Security Model

Android is another popular platform created by Google and the Open Handset Alliance. It is based on a Linux kernel and is typically programmed with Java language. The software developers write their applications in Java programming language and convert the applications to run on the proprietary Dalvik platform for Android OS. Once the conversion process is complete, the application can run on any Android device. Each Android application runs within its own virtual machine which is isolated in its own Linux process. Unless the device is jailbroken, this model ensures that no process can access resources of any another process. This sandboxed technology is capable of containing the potentially malicious program, However, Android does not rely on it alone, and instead security is enforced directly by the Linux-based Android operating system.

Android security model is based three security pillars: Access Control, Application Isolation and Permissions Access Control. Google releases the source code of the entire Android project that enables the scrutiny from the broader security community. It differs from Blackberry OS and Apple iOS which are closed source platforms. Let's examine these security pillars in detail and how these protect against common attacks.

Android Access Control

Android version 2.x or later provides the password configuration options which includes defining the strength of passcode, lockout time span and failed login attempts before the device wipes its data. With Android 3.x later, the administrators can compel their users to update their password using the password expiration feature. However, there is no enterprise scale reliable centralized password control mechanism available as in the case of Blackberry and Apple iOS in the forms of Blackberry Enterprise Server and Microsoft ActiveSync. Android password policy system is sufficient to protect the devices against the casual attacks. However, some versions of Android do not encrypt data stored on the removable SD memory card. Hence, an attacker with physical access to an Android device could simply eject the SD memory card and obtain a subset of the device's data in a matter of seconds, bypassing any and all password controls enabled on the device. However, this limitation has been overcome by encrypting SD memory card in Android versions 3.x or later.

Android Application Isolation

The applications are given a unique user identifier (UID); this is similar to Unix UID seen on desktops and servers. It is a small number like 1011 that is unique on a given system and used by the kernel to control access to files, devices and other resources. Applications will always run as their given UID on a particular device, just like users always have their same UID on a particular server but different UIDs on unrelated systems. The UID of an application is used to protect its data and developers need to be explicit about sharing the data with other applications. This isolation system not only isolates the applications from other applications on the system but also prevents applications from accessing or modifying operating system kernel. It ensures that a malicious application cannot gain administrator-level control over the device. The default

isolation policy prohibits the access to virtually every subsystem of the device with the following exceptions:-

- The applications may obtain the list of applications installed on the device and examine each application's programming logic but not its private data.
- The Applications may read the contents of the user's SD flash card which typically holds the user's music, video files, installed programs and documents or saved attachments regardless of which application created a particular piece of data.
- Apps may launch other applications on the system such as the Web browser and the maps application.

Android uses the same security model as Apple iOS. Any data, process or permission associated with the application remains glued to the identity and also reducing the amount of sharing across the core OS. For example, the data, files, and folders assigned to a certain application identity would not have access to any data, file, and folders assigned to another application's identity.

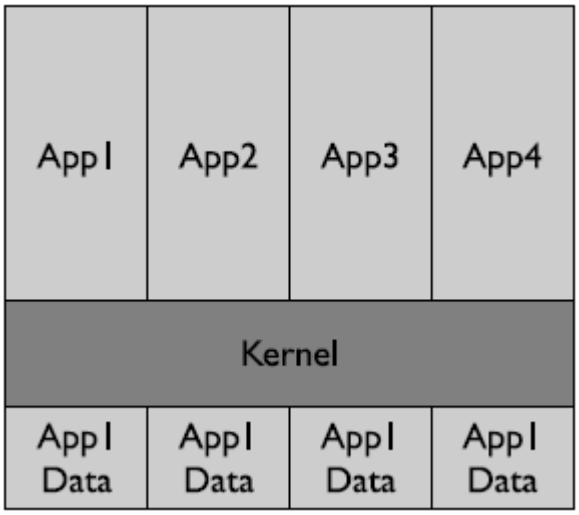

Figure 11: Apple iOS Application Isolation Model (Dwivedi Clark & Thiel, 2010)

Since Android isolates each application from every other application on the system and from the operating system itself, this means that if an attacker compromises a legitimate application, they will not be able to attack other applications and Android operating system itself. Let's take the example of web browser application. Since the attackers know that web browsers often have security flaws that can easily be exploited by a properly crafted malicious web page. Imagine a scenario in which an attacker posted a malicious web page capable of attacking a known flaw of Android's web browser. If an unsuspecting user surfed to this web page, the attack could inject itself into the Android browser and begin running. Android's isolation policy would ensure that the attack could not spread beyond the browser to other applications on the system or to the operating system kernel itself. However, such an attack could access any parts of the system that the web browser application had been granted permission to access. For example, if the web browser had permission to save or modify data on the user's SD storage card, then the attacker could take advantage of this permission to corrupt data on the SD storage card. Therefore, an attacker effectively gains the same control over the device as the application they manage to attack, with varying implications depending on the set of permissions requested by the compromised application.

Moreover, the malicious code within the attacked process can also steal any data that flows through the process itself. In the case of a web browser, the attacker could easily obtain login names, passwords, credit card numbers, account numbers, browsing history and bookmarks. Since mobile users often access the internal enterprise applications via their mobile web browser, this could lead to leakage of highly sensitive enterprise data even if VPN or SSL encryption is employed. Such a malicious agent in the browser could also initiate malicious transactions on behalf of the user without their consent. Android applications can do very little without explicitly

requesting the permission from the user to do so. If an application wants to communicate over the internet, it must explicitly request the permission from the user to do so; otherwise the default isolation policy blocks it from initiating the direct network communications.

Android Permissions Access Control

Each Android application contains an embedded list of permissions that it needs in order to function properly. This list of permissions is presented to the user in non-technical language at the time an application is installed on the device and the user can then decide whether or not to allow the application to be installed based on their tolerance for risk. If the user chooses to proceed with the installation, the application is granted permission to access all of the requested subsystems. On the other hand, if the user chooses to abort the installation, then the application is completely blocked from running. Android offers no middle ground which allows little permission while rejecting others. The third-party applications can request the permissions to use the following high-level subsystems:-

- Applications can obtain the device's phone number, the device ID (IMEI) number and its SIM card's serial number. These codes can be used by criminals to commit cellular phone fraud.
- Applications can establish network connections with other networked devices over Wi-Fi or using the cellular signal.
- Applications can access emails and attachments in the device's inbox, outbox, and SMS systems.

- Applications can also initiate the transmission of outgoing emails and SMS messages without user prompting and receive incoming emails and SMS messages.
- Applications may access multimedia and pictures hosted by the device's photo application.
- Applications can read, modify and delete new entries to the system calendar and address book.
- Applications can request to save, modify or delete existing data on external plug-and-play SD memory cards.
- Applications may obtain the device's location.
- Applications may access the device's logs and obtain the list of currently running applications.

Let's see how effective Android permissions access control has been. Android's permission access control system seems to be extremely robust, enabling software vendors to limit an application to the minimal set of device resources required for the operation. The problem with this approach is that ultimately, it relies upon the user to make all policy decisions and decide whether an application requested combination of permissions is safe or not. Unfortunately, the users are not technically equipped to make these security decisions in the vast majority of cases. By requesting the proper permissions, a malicious application could launch resource abuse attacks, data loss attacks as well as integrity attacks.

Android Application Provenance

The ultimate goal of digitally signing an application is to ensure that the application logic is not tampered with and to determine the identity of application's author. Google's approach undermines both of these goals. Like Apple, Android operating system will only install and run applications that have been properly signed with a digital certificate. However, software developers need not to apply to Google to obtain a code-signing certificate. Instead, the application developers can generate their own signing certificates, as often as they like, without any oversight. In fact, the software developer can place any company name and contact information in their certificate which they like. The result is that a malware author can generate anonymous digital certificates as often as he likes and none of these certificates or malware signed with them can be traced back to the author.

In order for the developers to sell their applications on Google's official Android Marketplace, the developers must pay a fee via credit card. This enables Google to associate the payee with the digital certificate used to digitally sign the developer's applications and should act as a mild deterrent against the malware authors posting malware on the Android Marketplace. However, given that developers have the ability to distribute their applications from virtually any website on the Internet, not just the Android marketplace. The malware programs can also be distributed without any vetting process by Google. The attacker can also obtain a legitimate application, add some malicious logic to the application and then, re-sign the updated version with an anonymous certificate and post it onto the Internet. While the newly signed application will lose its original digital signature, Android will certify and install the newly signed malicious application with its anonymous digital signature. Thus, Android's model does not realistically prevent tampering.

While Apple enforces a single vetting and distribution channel for all iOS apps, Google chose a far more open model for Android applications. First, Google does not appear to perform a rigorous security analysis of applications posted onto its Android Marketplace. This means that the malware authors can distribute their applications through this distribution channel with less likelihood of being discovered. While Apple's certification approach can certainly fail to detect some classes of attacks, it at least acts as a deterrent to malware authors. In contrast, Google's lack of validation offers less of a deterrent. Second, Android application developers can distribute their applications from virtually any website on the Internet. By default, Android devices may only download applications from Google. Thus, the users may override this setting with a few taps of their touch screen and further download applications from virtually anywhere. In a nutshell, the provenance approach adopted by Google for Android devices is less rigorous and hence less secure.

Android Encryption

Android devices running OS version below 3.x rely upon the isolation model, instead of encryption to protect data such as passwords, user names, application-specific data, etc. This means that if an attacker is able to jailbreak a device or otherwise obtain administrator level access to a device either by exploiting the vulnerability or by obtaining the physical access to a device, they can access virtually every byte of data on the device including most of the passwords, Microsoft Exchange/private email account credentials. However, Google introduced the device encryption features starting with Android version 3.0 and later.

The third-party Android applications may optionally encrypt their data using standards-based encryption algorithms. However, the application developers must explicitly add this logic to their program. Otherwise, all data created by the applications is saved in an unencrypted form.

Although Android security model is a major improvement over the models used by the traditional desktop and server-based operating systems yet it has two major drawbacks. Firstly, its provenance system enables attackers to anonymously create and distribute the malware and malicious applications. Secondly, its permission system, while extremely powerful, ultimately relies upon the user to make the important security decisions. Unfortunately, most users are not technically capable of making such decisions and this can lead to social engineering attacks.

9. Blackberry Device Security Model

The BlackBerry device and supporting platform are developed by Research In Motion (RIM), a Canadian software and hardware company based in Waterloo, Ontario. One of the BlackBerry's main selling points is the provision of an integrated wireless messaging system which offers the push email access over the cellular wireless networks throughout the world. Another major factor in the BlackBerry's popularity is its comprehensive and systematic approach to the security. BlackBerry devices are versatile and can be used for a range of functions including telephony, SMS, email, and web browsing amongst other things. The BlackBerry OS is primarily based on Java and supports J2ME Mobile Information Device Profile (MIDP), Wireless Application Protocol (WAP) and Connected Limited Device Configuration (CLDC) profiles natively. A RIM proprietary Java API for using device-specific features is required to take a complete advantage of the BlackBerry platform. Most BlackBerry specific Java applications are CLDC based and use RIM’s proprietary APIs. RIM calls these applications RIMlets.

The BlackBerry OS is a modern OS with features such as multitasking, inter-process communication (IPC) and threads. All OS and device features are accessed using RIM and J2ME APIs. The security is enforced using a combination of signatures, Java verification and class restrictions. The Java Virtual Machine does not support Java native invocation which prevents the attackers from controlling the device in ways that RIM did not intend. To increase the speed, RIM created a custom Java Virtual Machine that supports the standard JME instruction set and several RIM JVM-specific instructions. All other applications, such as messaging and the browser, are written using Java. The security system is intended to control the access to the data and does not prevent the applications from consuming an unfair share of memory or CPU time. The OS does not enforce limitations on the number of objects an application can create and the developers are responsible for minimizing the amount of memory and system resources that their applications could use. When the JVM is no longer able to allocate storage space for objects, Java garbage collection runs to remove unused objects from the memory.

Blackberry Access Control

Blackberry devices provide pretty good options for access control security options including password configuration options as well as account lockout options. These options can be controlled from Blackberry Enterprise Server via Blackberry policies. Blackberry devices also provide adequate protection if the device is lost as a remote wipe can be initiated by the administrator from Blackberry Enterprise Server.

Blackberry Application Sandboxing

Blackberry application sandboxing is based on the traditional model which uses Normal and Privileged assignments. In Normal and Privileged assignments, the certain applications have

access to everything on the device and Normal applications have restricted access on the device. This model would prevent Normal applications from accessing the parts of the file system that are set aside for Privileged applications. However, all Normal applications would have access to the same set of files and folders on the device.

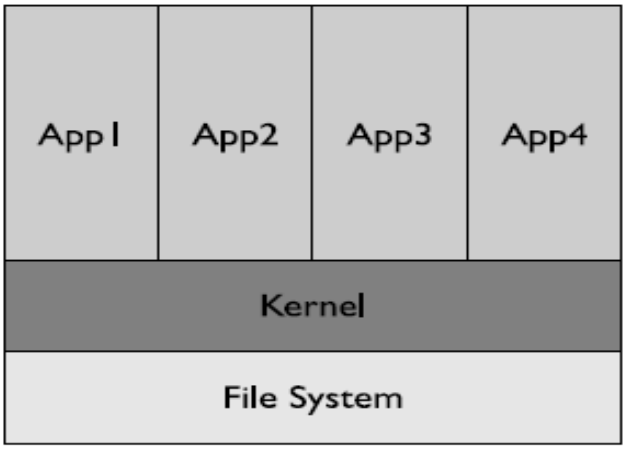

Figure 12: Blackberry OS App Isolation Model (Dwivedi Clark & Thiel, D., 2010)

Privileged applications have full access to the entire device and its data, processes, APIs, files. Normal applications have access to only parts of the file system, but all the Normal applications have access to the same subset of the operating systems. Although one Normal application can access the same part of the file system as another Normal application, it cannot directly read or write to the other application's process memory. The application signing process decides if an application is allowed to run in Normal mode or Privilege mode. The BlackBerry OS's file system is laid out somewhat like a traditional Unix file system with the exception that there is no root directory. The "File:" URLs are used for referring to individual files, and URLs must contain the physical storage location. The BlackBerry implements simple file-access restrictions and not all files are readable or writable by all applications. For example, unsigned applications can write files under the file:///store/home/user directory but not under the operating system

location file:///store/samples. Blackberry application sandboxing is good but weak in comparison with modern application isolation model used by iOS and Android.

Blackberry Permissions Access Control

Permissions are determined on per application basis and assigned based on the application's signature or a policy specified by the user. Most APIs are not considered sensitive and can be accessed by unsigned applications. The sensitive API set includes APIs for accessing personal information manager (PIM) data, phone features, operating system configuration and the network. Applications that use these APIs may have to be signed, depending on whether the sensitive API is an MIDP or CLDC API or a RIM proprietary API. (Dwivedi, H., Clark, C., & Thiel, D., 2010)

RIM Controlled APIs are divided into different API sets. Each set has a unique signing authority. The three most common signing authorities are:

- RIM Cryptographic Runtime includes the majority of RIM's Cryptographic APIs.
- RIM Runtime API provides access to sensitive platform functionality such as the Application Permission Manager.
- RIM BlackBerry Apps API provides control of built-in BlackBerry applications. For example, the BlackBerry browser.

The applications that use more than one controlled API set are signed with multiple signatures. For example, an application that uses the BlackBerry browser and the Application Permission Manager will have two signatures. The signing infrastructure is extensible and the third-party developers can add their own signing authorities to control access to their APIs by using the BlackBerry Signing Authority Tool. When developers purchase signing keys from RIM, they

receive authorization for signing from the three common signing authorities RCR, RRT and RBB. The two BlackBerry OS features also require signatures: applications that automatically launch on BlackBerry start up and BlackBerry system modules. If these features were allowed, unsafe code could run as start up and either monitor the user's actions or commit an effective spoofing attack to steal the important information. Before an application is allowed to run, it must pass a Java verification stage to ensure that the application uses well-formed Java instructions and does not use dangerous Java features such as Java native invocation or reflection. Disallowing these features ensures that the application does not load dangerous code or bypass class access restrictions to call or create prohibited methods.

Only the BlackBerry JVM and lowest-level firmware are written in the native code (C/C++, ASM) which eliminates a large portion of the BlackBerry's attack surface that may be vulnerable to buffer overflows and other memory corruption issues. This is proven by the fact that there are less reported BlackBerry memory corruption vulnerabilities. The combination of verified Java code, code integrity and per-API access control is powerful for security. These mechanisms enable sandboxed applications and enable users to control how those applications use data on the device. But the code-signing certificates are cheap and do not require an extensive authentication process. The requirement of signatures increases the accountability and enables the code integrity. However, a signature does not guarantee whether the signed code is well written or non-malicious. The only foolproof mechanism to avoid malware is to avoid installing third-party applications altogether.

Blackberry Application Provenance

BlackBerry signatures use a public key cryptography and RIM-managed online signing service. To sign code, the developers must have a public/private key pair and have registered that key pair with RIM. Getting signing keys is the first step in creating RIM signed applications. An individual application may not need signatures from every signing authority. For example, if the application does not use cryptography, then a RIM Cryptographic Runtime (RCR) signature is not required. The Signature Tool inspects the application and determines which keys are required. Behind the scenes, the private key is used to create a digital signature of the application which is then sent via HTTP POST to the appropriate signing authorities. If they accept the signature, they will sign the response and return a signature of the signature. The Signature Tool adds this to the COD file; a COD file is a RIM-style code module. By being online, RIM is able to monitor the signing process and control the number of times an individual signing key may be used. RIM can also respond to compromise and refuse signatures if a key is known to be compromised. Every time a signature is requested, an e-mail will be sent to the signature key's owner summarizing what was signed and who signed it. This email also contains the number of signatures remaining. Standard developer keys may be used a little over two billion times.

While code signing provides a potential hurdle for malicious code writers, signatures can still be obtained with relative ease and anonymity. Code-signing keys can be obtained anonymously via the use of prepaid credit-cards and false details. Pre-paid credit cards can be bought and charged locally with cash without the requirement of presenting identification proofs. This makes it potentially impossible to determine the creator of a signed malicious application, and as a result track the malicious user.

RIM has the ability to revoke signing keys. This disables signing keys and prevents their use to sign any further code. However code that has already been signed by such keys cannot be revoked, although it can still be blocked by IT Policy Application Control on Blackberry Enterprise Server deployments. This is in contrast with a Certificate Revocation List system. The signing keys are encrypted on the host by default, and the user must enter a password in order to decrypt the keys and initiate the code signing process. Offline brute force cracking of this key is not possible, because the only way to know if the key has been decrypted correctly is to initiate code signing with RIM across the network and to wait and see if it has been successful. The code signing process is monitored by RIM for anomalies such as a significant number of failed signing attempts, so attempts to crack the password online would be noticed. RIM's application provenance model is at par with Apple's approach.

Blackberry Encryption

The popularity of the BlackBerry in government and enterprises makes on-device encryption a necessity and the BlackBerry's secure storage options are extremely advanced. To encrypt sensitive messaging and contact data stored on the device, BlackBerry uses content-protection feature. The content protection encrypts data when it is written to flash memory using a key generated from the unlock password. Because there would be no way to generate the key without a password, the user is required to specify an unlock password. Users can optionally encrypt the address book, which has the interesting side effect of causing caller ID to not show the name of incoming callers when the device is locked.

Three keys are used by Blackberry content protection to protect data. There's an ephemeral AES key for unlocking keys, a 256-bit AES key for persistently stored data, and an Elliptical Curve

Cryptography (ECC) public/private key pair used for encrypting data when the device is locked. The length of the ECC key can be changed in security options and can be up to 571 bits long. The ephemeral AES key is generated from the device lock password and is therefore only as strong as the password itself. The ECC public key is kept in memory while the device is locked and encrypts all incoming data. The public key has to be used because the AES storage key is wiped from memory as soon as the device is locked. By only keeping a public key in memory, the BlackBerry protects against attackers who are able to read the device's memory directly. When the user unlocks the device with their password, the ephemeral key is used to decrypt the AES storage key and the ECC private key. The ECC private key is then used to decrypt all of the data that arrived while the device was locked; before being written to persistent storage this clear text is encrypted with the AES storage key. So in a nutshell data is encrypted with a key that comes from the unlock password so this requires a good password. Keys must be held in accessible memory for some period of time if they are going to be used to perform all of these encryption operations. The BlackBerry can be configured to scrub sensitive data from memory when the device is locked, holstered, or idle.

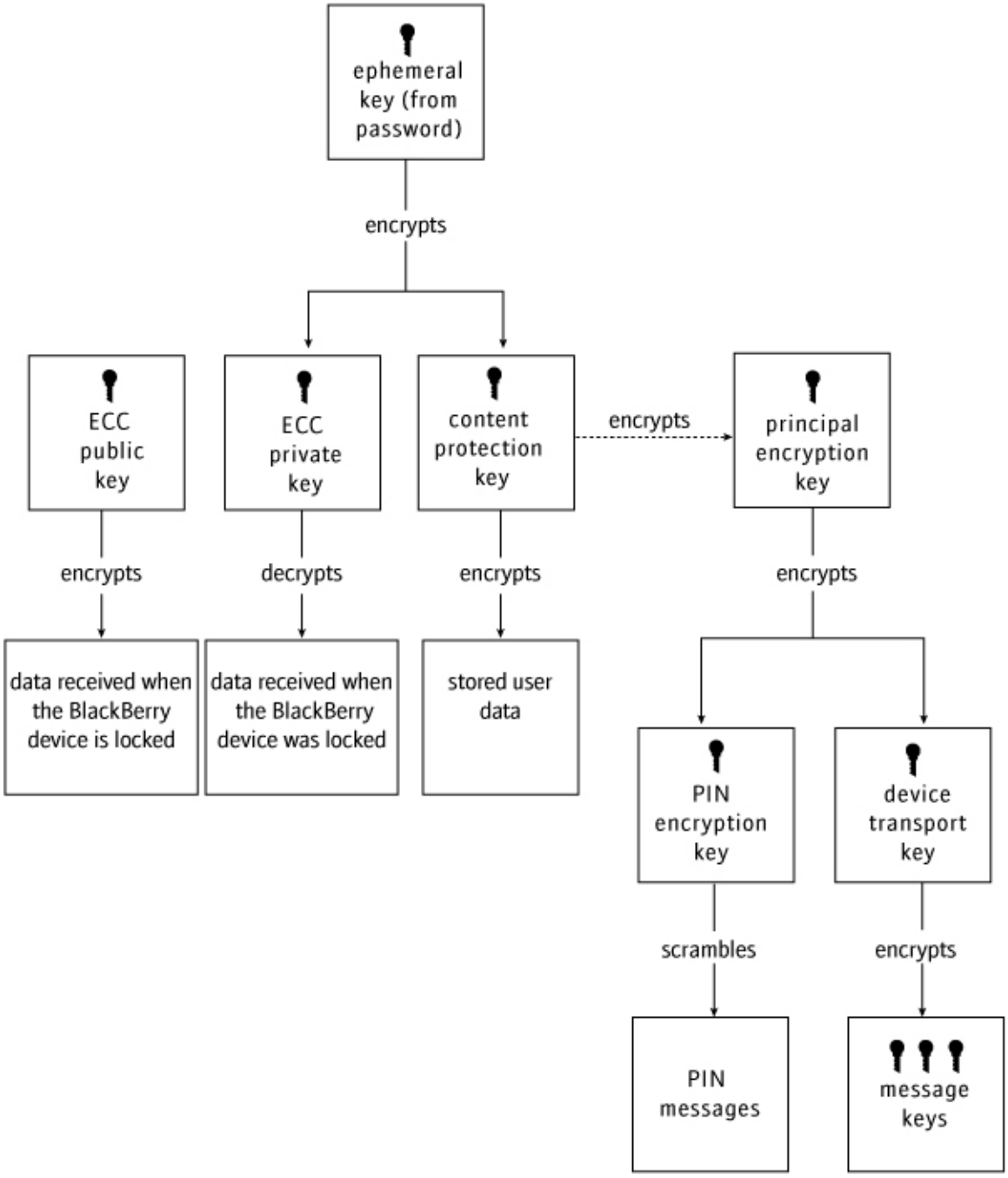

Figure 14: Blackberry Content Protection Keys (Research In Motion Limited, 2011)

Many BlackBerry devices include the memory card slots for the storage expansion. The smaller these cards physically get, the easier they are to lose and the more protection is needed. BlackBerry can encrypt documents and media files stored on removable media using the same content protection mechanism as is used on the primary device. There are three modes for protecting external media:-

- **Device Mode**: - The BlackBerry uses a cryptographic random number generator to generate the external memory encryption key. If the card goes missing, but the device stays in the owner's possession, then anyone who finds the memory card will be unable to read it because the key is still on the device.
- **Security Password Mode: -** The user's device password is used to generate an encryption key for the device. This is the weakest form of protection because users choose poor passwords and attackers who get the SD card can perform offline attacks against the encryption key.
- **Security Password + Device Mode**: - A combination of the device password and a randomly generated per-device key is used to encrypt the memory card. The combination of the two key-generation methods prevents the attacks possible against each one alone.

In a nutshell, Blackberry provides very good encryption compared to Apple iOS and Android. Overall BlackBerry is an advanced platform with many security features for users and application developers. When compared against the other mobile platforms, BlackBerry is the clear leader in on-device security and manageability. Applications are easily isolated from each other, and users or administrators are able to control how applications interact using BES IT policies

10. Device Security Solutions

The smartphones are different than PCs when it comes to available security solutions. This is due to low resources and processing power available on these devices.The following solutions can be used to strengthen the device security:-

Mobile Antivirus/Anti-malware Protection

There are already a number of first-generation antivirus scanners for the Android platform. However, given iOS's strict isolation model, it is difficult to implement an iOS-based antivirus scanner without relaxation of the isolation system by Apple.

The mobile antivirus solutions can address threats in the malware category as well as a subset of malware-based attacks in the resource abuse, data loss, and data integrity categories. These solutions scan for mobile threats and block them from installing on the device. To be most effective, anti-malware and antivirus solutions should receive regular updates of new patterns for known malware with minimum user or administrative intervention. Mobile data should be scanned in real time. This includes data on mobile devices and on external memory cards when inserted. If needed, the administrator should be able to schedule a manual scan on one or more devices.

Enforce Active Firewall and Intrusion Detection Systems

In the same way that PC security solutions typically include a firewall to restrict access, mobile devices also need firewall protection. The firewalls have stateful inspection capabilities. When activated, firewalls limit the type and origins of traffic. The firewalls are necessary to block port scans that attackers may use to determine vulnerabilities when the device is connected to a public network. They also are the first line of defence against any exploitation of unpatched security holes in an operating system or client applications. Blackberry devices have firewall capabilities that can be turned on from BES using IT Policies. However iPhone and Android devices currently lack such centralized management of firewall rules.

Improve User Awareness and Training

Improving security protection involves considering people and processes along with technology, and includes comprehensive training to communicate the threats posed to networks through the use of mobile devices. Just as the PC users recognize spam and phishing, mobile users also need to recognize the potential threats on their devices. Users should be educated to follow corporate policies for safe use of mobile devices such as do not download applications from untrusted sources, do not keep Bluetooth functionality enabled when not in use, do backup all data on the device, do keep the software on the device up to date and follow tips for safe synchronization.

Centralized Management for Mobile Devices

The centralized management reduces complexity and cost of managing multiple devices. Mobile devices are by definition routinely outside the organization, so any solution that does not enable remote management and policy enforcement for these devices may be inadequate. These tools enable the administrator to remotely administer managed Blackberry, iOS and Android devices. The typical security policies might include setting the password strength, configuring the device's VPN settings, specifying the screen lock duration, or disabling specific device functions to prohibit potentially risky behaviours. In addition, the administrator may perform security operations like wiping lost or stolen devices, or using the device's onboard geo-location service to locate a device. These tools ensure that all mobile devices contain the same version of the same software and removes applications that are unauthorized. Software distribution is also a critical element in the mobile data protection strategy. The Blackberry Enterprise Server solution is ahead of its competitors in this area.

Secure Browser

Most known attacks against the smartphones are carried out using vulnerabilities in mobile web browsers. Several companies have introduced their own secure Web browser apps for Blackberry, iOS and Android platforms. These applications are meant to be used instead of the built-in Web browsers provided on the iOS and Android platforms. Each time the user visits a URL in the secure browser, the browser checks the URL against a blacklist or reputation database and then blocks any malicious pages. The only problem with this secure browser approach is that the user cannot use the familiar, factory-installed Web browser shipped with the device. Instead, they must use the third-party secure Web browser to do all web surfing.

Enterprise Sandbox

Sandbox solutions aim to provide a secure sandbox environment where employees can access enterprise resources such as email, calendar, contacts, corporate websites, and sensitive documents. All data stored in the sandbox, and data transmitted to and from services accessible via the sandbox, is encrypted. To use the sandbox, the user must first log in and the sandbox must then check with a corporate server to ensure that the user is still authorized to access both local data on the device as well as enterprise services. Given that the sandbox is provisioned by the corporate administrator, it can easily be deprovisioned if the device is lost or stolen. This approach has the effect of dividing the device's contents into two zones: a secure zone for the enterprise data, and an insecure zone for the employee's personal and private data. The benefit of such a solution is it enables the consumer to use their own device, yet it still lets them safely access enterprise data. The drawback of such a device is that the user can't use the regular mail, calendar, or contact apps built into the device to access enterprise resources, forcing them to adapt to a different set of sandboxed, potentially less usable equivalent apps. RIM has recently

introduced BlackBerry Balance technology allowing users to use their personal mobile devices for work without compromising the enterprise's need for security. This is built on enterprise sandboxing technology.

11. Comparison of Different Smartphones' Features

Capability	**Apple iOS 4.x**	**Android 2.3**	**Blackberry OS 6.x**
Complex Passwords	Yes	Yes	Yes
Support VPNs	Yes	Yes	Yes
On-Device Encryption	Yes	No, Introduced in Android 3.x or later	Yes
Enforce Password Policies	Yes using AS,TP and iPhone profiles	Yes, Require PIN only using AS	Yes using BES
Disable Camera	Yes using AS,TP and iPhone profiles	No	Yes using BES
Restrict App Store	Yes using AS,TP and iPhone profiles	No	Yes using BES
Remote Lockout	Yes using AS,TP and iPhone profiles	Yes using AS and TP	Yes using BES
Remote Wipe	Yes using AS,TP and iPhone profiles	Yes using AS and TP	Yes using BES
Restrict Wi-Fi	Yes using AS,TP and iPhone profiles	No	Yes using BES
Selective Wipe of Data and Business Applications	Yes using TP	No	Yes using BES for Blackberry OS 6.x or later
Enforce and Mange policies	Yes using AS and TP	Yes using AS and TP	Yes using BES
Manage Over the Air	Yes	Yes using AS and TP	Yes using BES
Second Sector Authentication	No	No	Yes with certain models
Number of Policies	Around 30	Around 30	Around 500

Active Sync – AS, Blackberry Enterprise Server – BES, Third Party Software – TP

12. Blackberry Transport Security

The BlackBerry architecture is a fully integrated solution from one vendor that includes the device and its OS, the BlackBerry Enterprise Server (BES) and RIM owned Network Operations Center (NOC). The system is designed so that data remains encrypted at all points between the BlackBerry device and the BES. In addition to this, Blackberries also support both S/MIME and PGP, sender to recipient security solutions, which ensures that the message cannot be read or modified anywhere along the way. Mutual authentication and transport encryption between the device and the BES provides confidentiality, integrity and authentication without requiring a separate VPN.

Some companies have expressed concern that the NOC provides a single point of failure and they are uncomfortable with certain data transiting through a foreign country. However, all messages sent through to the NOC are encrypted using Triple DES or AES 256 encryption and all messages are encrypted with keys, which are stored only in the BES and the device. Neither RIM nor the operators have access to the customer keys and therefore cannot see the content of any of the messages. RIM also claims the NOC reduces costs by amortizing the cost of multiple redundant connections to the carrier across all BES servers.

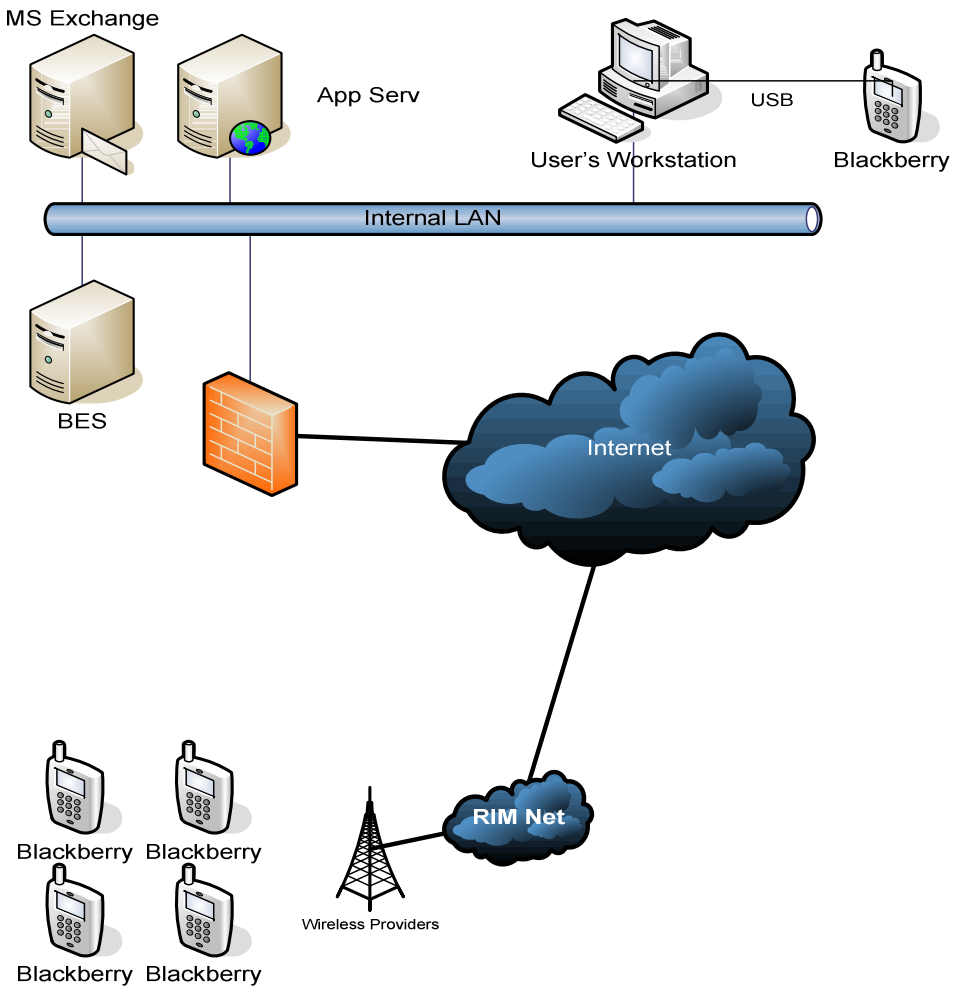

Figure 14: Blackberry Infrastrucutre

The BlackBerry Enterprise Server is placed behind the corporate firewall and includes all the modules necessary to access internal corporate data. The RIM Network Operation Center (RIM NOC) is a key part of the BlackBerry BES solution. This center operates from Canada and is the point where the BlackBerry device and the BES can find each other and communicate. The NOC takes care of handling individual BlackBerry connections and also queues up data that is destined for a BlackBerry when it is out of coverage or turned off. For a corporation, the only connection being used for all communication is the one that is established between BES and NOC. The traffic and uptime to the devices is handled by the NOC. Every BES has a unique address called

a Server Relay Protocol (SRP) ID which allows the BES to log and identify itself to the NOC. This same SRP ID is included in the BlackBerry Service Book. When the BlackBerry is turned on, it registers with the RIM NOC using its PIN number, which refers to the SRP ID. Since the RIM NOC is aware of both the BlackBerry and the BES, it allows them to communicate with one another via the RIM NOC. During the activation of the BlackBerry device to the respective BES, which is done at the corporation, encryption keys are interchanged and then used for all communication.

The architecture is quite simple. Your BES administrator adds you to a BES, then your BlackBerry associates with that BES by performing an activation process. This activation process used to be done via the USB cable connected to your PC, but it can also be done wirelessly. The activation process essentially sets up the BlackBerry to communicate with the BES it is assigned to by first establishing an encryption key which is then used to encrypt all data using 3DES or AES that is sent and received from the BlackBerry in the same way as a Virtual Private Network. Then, the BES sends a few Service Books to the BlackBerry which tells it who to communicate with when it needs to do certain things. For example, who to talk when sends e-mail, who to talk to when browsing the web, etc. After that data is synchronized down to the BlackBerry. This could be the last 5 days of email, the last 90 days of calendar entries, the entire address book, etc. In the mean time, the BES starts watching the BlackBerry user's corporate mailbox for changes. When it sees them, they are instantly sent to the BlackBerry. Depending on the email system being used (Novell GroupWise, Lotus Domino, or Microsoft Exchange) the mechanism used to figure out what has changed is different. In an Exchange environment for example, the BES makes a request to the Exchange server and asks that it be told whenever a new email arrives. The Exchange server duly obeys and when a new email arrives it notifies the

BES, which in turn grabs a copy of that new email and sends it to the BlackBerry. It all happens within seconds.

So, the RIM NOC is the point where BlackBerry and BES can find each other and communicate. This means that the BES itself doesn't need to worry about doing that extra work. In fact the BlackBerry architecture itself allows any company to add an infinite number of BlackBerry users without the need to ramp up remote connectivity capacity, since the only connection being used for all communication is the one that is established between BES and NOC. The NOC also removes the need to run a 100% uptime remote connectivity environment since RIM takes care of this at the NOC.

Before the device sends a message, it compresses and encrypts the message using the device transport key. When the BlackBerry Enterprise Server receives a message from the device, the BlackBerry Dispatcher decrypts the message using the device transport key, and then decompresses the message.

The BlackBerry Enterprise Solution uses AES or Triple DES as the symmetric key cryptographic algorithm for encrypting data. By default, the BlackBerry Enterprise Server uses the strongest algorithm that both the BlackBerry Enterprise Server and the BlackBerry device support for BlackBerry transport layer encryption. If you configure the BlackBerry Enterprise Server to support AES and Triple DES, by default, the BlackBerry Enterprise Solution generates device transport keys using AES encryption. The BlackBerry Enterprise Solution uses AES in CBC mode to generate the message keys and device transport keys. The keys consist of 256 bits of data. The BlackBerry Enterprise Solution uses a two-key Triple DES encryption algorithm to generate message keys and device transport keys. In the three iterations of the DES algorithm,

the first 56-bit key in outer CBC mode encrypts the data, the second 56-bit key decrypts the data, and the first key encrypts the data again.

When a sender sends an email message to a Blackberry device user, The BlackBerry Enterprise Server performs the following actions:

- compresses the email message
- encrypts the email message using the message key
- encrypts the message key using the device transport key of the device
- sends the encrypted email message and encrypted message key to the device

The BlackBerry device user clicks on the email message on the device to open it. The device performs the following actions:

- decrypts the message key using the device transport key
- decrypts the email message using the message key
- decompresses the email message
- displays the email message to the user

When a sender sends an email message from a BlackBerry device to a recipient, the device performs the following actions:

- compresses the email message
- encrypts the compressed email message using the message key
- encrypts the message key using the device transport key of the device
- sends the encrypted message key and encrypted email message to the BlackBerry Enterprise Server

The BlackBerry Enterprise Server performs the following actions:

- decrypts the message key using the device transport key
- decrypts the email message using the message key
- decompresses the email message
- forwards the email message to the recipient

The activation of a Blackberry device over the wireless network follows the following process:-

- A user opens the activation application on the BlackBerry device, and types the appropriate email address and activation password.
- The device sends an activation request to the BlackBerry Infrastructure using standard BlackBerry protocols. The BlackBerry Infrastructure uses SMTP to send an activation message to the user's email account. The activation message contains routing information for the device and public keys.
- The BlackBerry Enterprise Server sends an activation response to the device. The activation response contains routing information for the BlackBerry Enterprise Server and the long-term public keys of the BlackBerry Enterprise Server.
- The BlackBerry Enterprise Server and device use the initial key establishment protocol to generate a device transport key and verify it. If the BlackBerry Enterprise Server and device mutually verify the device transport key, the activation process proceeds. The BlackBerry Enterprise Server and device use the device transport key to encrypt further communication between each other without sending the device transport key over the wireless network.
- The BlackBerry Enterprise Server then sends the appropriate service books to the device so that the user can send messages from and receive messages on the device

13. Attacks against Blackberry Infrastructure

Blackberry Enterprise Server creates outbound, persistent connection to RIM network. Blackberry device then virtually placed on internal network wherever BES exists. It's always-on always connected independent of Wireless carrier. Generally only the security of data on handheld is considered not its impact on rest of network. Blackberries are computers with constant connection to corporate LAN. This can be exploited using a technique called BBProxy which allows exploitation of vulnerable service behind corporate firewall. This is how it can be achieved:-

- Create an outbound socket connection from Blackberry device to attacker controlled host on the internet.
- From attacker controlled host, we then initiate a subsequent socket connection to a second host including internal hosts.
- Blackberry then proxies all data between hosts.
- Now we can directly communicate with any port on an internal host from an external host through Blackberry handheld

BBProxy requires the control of device which can be achieved with an interactive application in the form of a Blackberry Trojan. This type of attack is practically hard to pull off. One such Trojan in past was TicTacToe game.

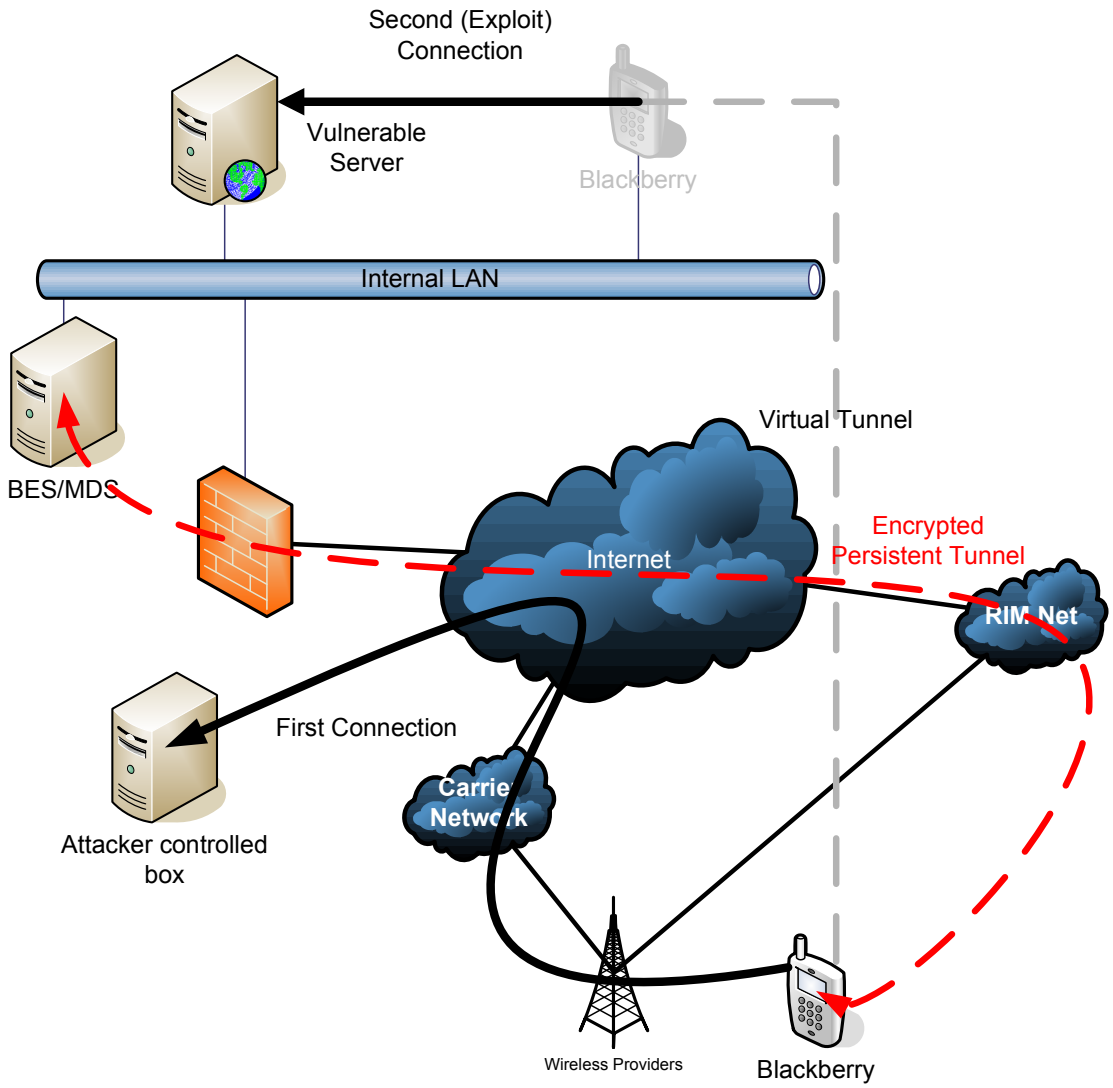

Figure 15: BBProxy Attack

14. Securing Blackberry Infrastructure

The BlackBerry Router is the component of the BlackBerry Enterprise Server that connects to the outside network. The DMZ is a neutral sub-network between the trusted corporate LAN, and the un-trusted external wireless network and public Internet. The BlackBerry Router can safely be implemented in the DMZ because all traffic that passes through the BlackBerry Router is encrypted, and all connections to the BlackBerry Router are authenticated. No encryption keys are stored in or transferred through the BlackBerry Router. This BlackBerry Router might enable

further security options because the BlackBerry Router does not have encryption keys and therefore does not compromise the security of the BlackBerry Infrastructure if the BlackBerry Router itself is compromised. It does not prevent BBProxy attacks directly but it does enhance the security of BES solution by reducing the attack surface.

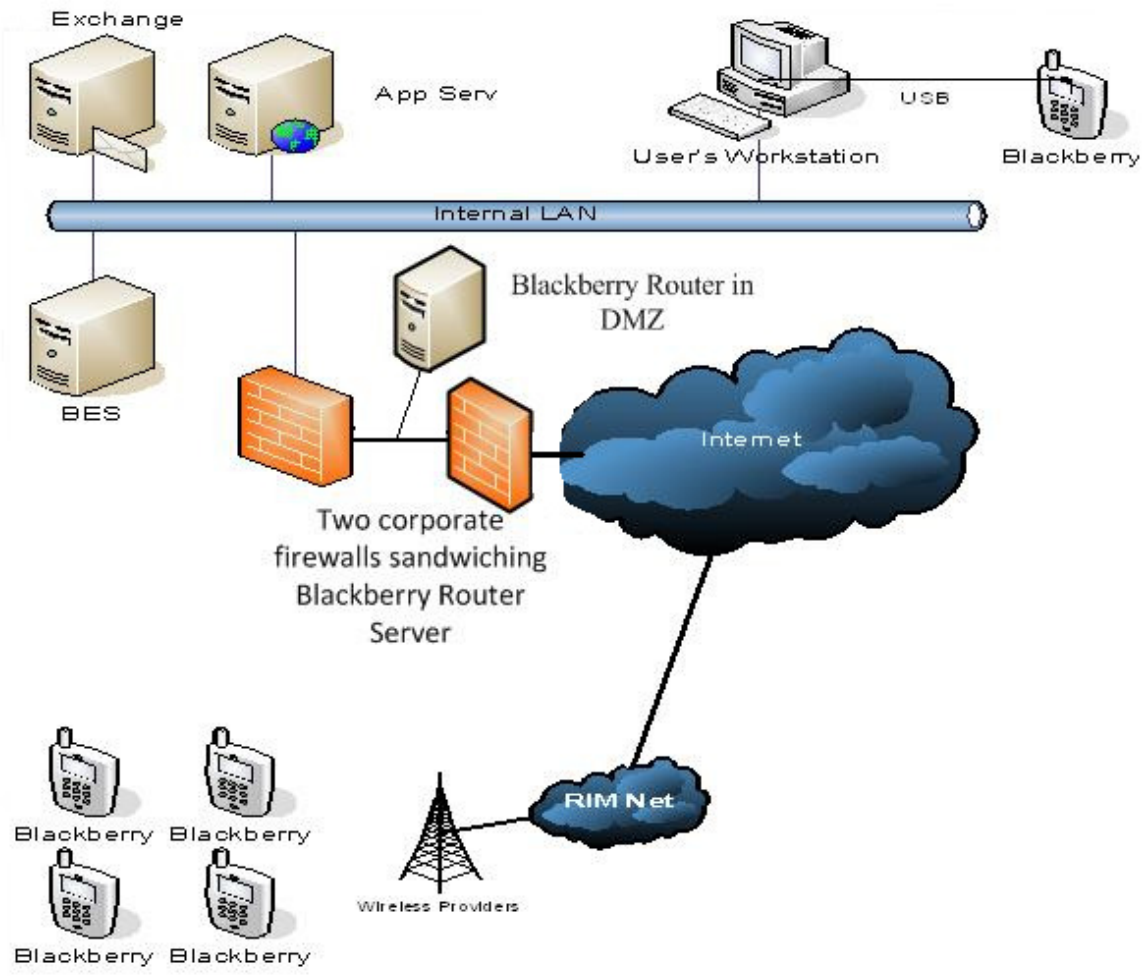

Figure 16: Blackberry Enterprise Server Best Practice

BES needs access to only Exchange server and Domain controller assuming the users won't be using Blackberry desktop manager on their corporate workstations. Access control lists should be used to restrict to tightly control this traffic. This will mitigate BBProxy attacks even if your Blackberries are compromised.

15. ActiveSync Transport Security

Exchange ActiveSync (EAS) is a proprietary data synchronization protocol created by Microsoft for wireless synchronization of mobile devices with Exchange Server. Exchange ActiveSync is optimized to work together with high-latency and low-bandwidth networks typical to mobile devices environments. The protocol, based on HTTP and XML, lets smartphones gain centralized access to their e-mail, calendar, contacts, and tasks and to have access to this information also while they are working off-line.

Exchange ActiveSync supports many mobile operating systems out of the box:

- Windows CE, PocketPC
- Windows Smartphone
- Windows Mobile 5,6
- iPhone OS X
- Symbian S60, S90 powered Nokia phones (latest firmware)
- Palm OS 4
- Palm WebOS
- Google Android (selected models)

Security is resonant with Blackberry but it is equally good in Exchange Active Sync. The mechanism used in Exchange Active Sync is SSL V2 on port 443, which is competitively good. To incorporate ActiveSync in a company, all you need is Microsoft Exchange Server with ActiveSync virtual directory published with external URL and https enabled. Apart from this one need to purchase certificate to enable https from third party vendor or install your Certificate Authority.

ActiveSync is based on Secure Sockets Layer (SSL) which is the de facto standard for providing data integrity and confidentiality for web transactions over the Internet. SSL encrypts pieces of application layer data over TCP connections providing confidentiality. It can also be used to test for the identity of the server, the client or both. SSL uses an efficient cryptographic algorithm for encrypting data and a computational intensive protocol for authentication and key exchange. The key exchange protocol employs asymmetric cryptography a methodology that requires the existence of a worldwide Public Key Infrastructure (PKI). PKI defines a procedure for binding digital certificates with respective websites by means of a chain of Certificate Authorities (CA). The binding is established through a registration and issuance process that ensures non-repudiation. The following figure explains the establishment of SSL session between a client and server:-

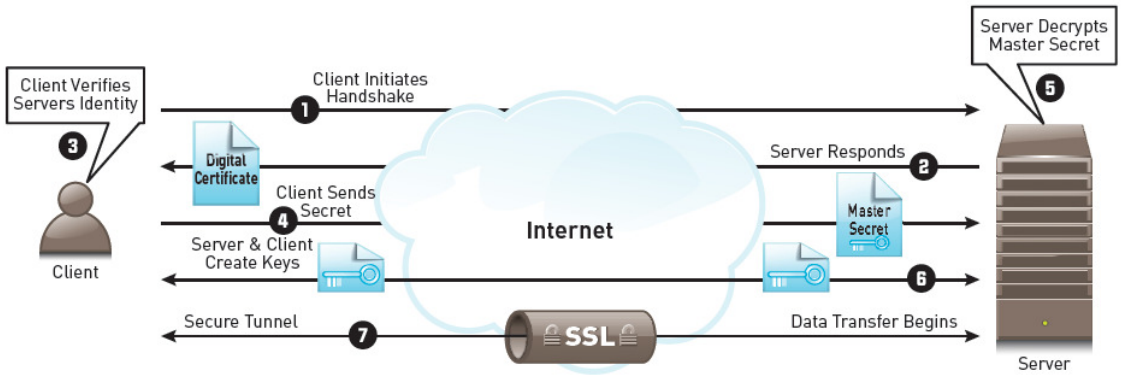

Figure 17: SSL Session Establishment (Blue Coat Systems, 2008)

16. ActiveSync Authentication Types

There are two types of authentications available with Exchange ActiveSync:-

The most common type of authentication when running ActiveSync is Basic Authentication. This basically means that when the client makes a request, it will supply a username and password. For security, SSL encryption is used so your username and password is not sent over the Internet in clear text. This SSL certificate needs to be issued by a CA (Certificate Authority).The reason for this is because the CA has to be trusted by the device; otherwise you have to add the Root CA certificate manually.

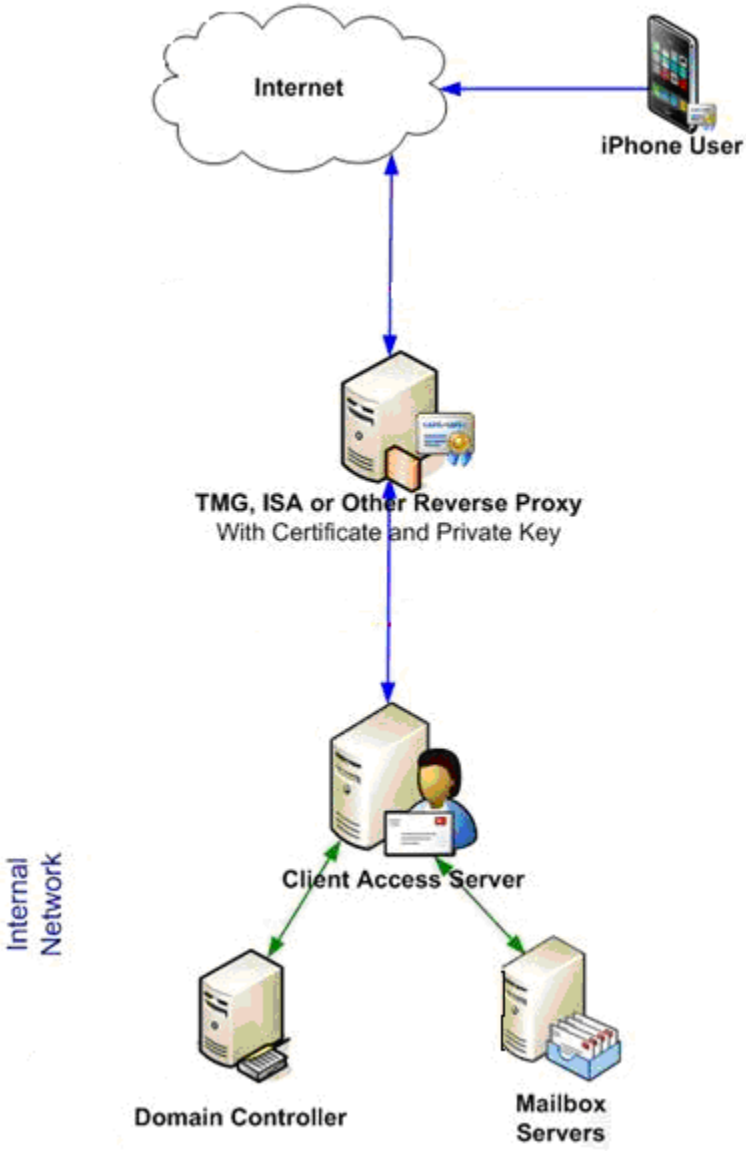

Figure 18: Exchange Active Sync Topology

The certificate-based authentication is the second option. With this, instead of configuring a username and password, the device uses a client certificate for the actual user, not the device. Basically, on the configure page where you usually supply username and password, it will just say "you're using a certificate to authenticate instead". Certificate-based authentication is not easy to manage if you have a lot of phones. Since you will need to add your Root CA certificate and issue client certificates for the user to each mobile phone.

The following process flow is taken when certificate based authentication is used for Activesync using TMG solution:-

- User attempts to access mailbox using OWA or a mobile device over a cellular network.
- Connection between client and external TMG interface is encrypted using SSL certificate.
- TMG has been configured to publish OWA and Exchange ActiveSync URL and prompts the user for authentication.
- The user or device presents an X509 certificate as proof of identity.
- TMG has been configured to use Kerberos Constrained Delegation (KCD) and connects to Key Distribution Center service on the Domain Controller and requests a Kerberos Ticket on behalf of the connecting user. This is where Protocol Transitioning takes place as initial user authentication method was X.509 certificate and TMG Firewall service has been configured to be allowed to request a Kerberos ticket on behalf of the user.
- The Key Distribution Center service passes back the user Kerberos Ticket to Network Service identity under which TMG Firewall is running.
- TMG has been configured to be able to delegate Kerberos tickets to an Exchange CAS server. The firewall rule on the TMG server knows the internal URL of the Exchange CAS server and passes the requested Kerberos Ticket for the authenticated user to the Exchange CAS server.

You can open up port 443 on your corporate firewall and forward it to your Exchange Client Access server but the most secure solution is to publish Activesync in your DMZ using Microsoft TMG server or another third party supported software in a reverse proxy setup. The TMG server also supports both Basic authentication and certificate-based authentication.

A single TMG listener cannot provide both certificate based authentication at the same time as Basic, Forms Based Authentication, or NTLM authentication. A listener can provide Forms Based Authentication with fallback to basic and NTLM can be used on the same listener, but certificate based authentication when used as the primary form of authentication cannot be combined with any other form of authentication. This means it is not possible to share one external namespace/listener between clients such as Exchange ActiveSync, when it is using both certificate based authentication and Basic Authentication. In a scenario such as this, two listeners, and therefore two namespaces and IP addresses must be used.

17. Attacks against ActiveSync

The adversaries with access to the mobile phone network and legitimately signed subordinate certificate authorities can pretty easily intercept all of the ActiveSync mail sent to your iOS and Android devices. The attacker needs simply to redirect traffic to known ActiveSync servers to a server of their own, where they dynamically generate a valid certificate signed by their sub-CA and proxy the ActiveSync connection by carrying out Man in the Middle Man attacks. State-sponsored attackers are the largest threat to the security of an ActiveSync connection, especially in places where the local phone network is controlled by the government.

This is especially a problem when basic authentication method is used for ActiveSync. Even if an attacker is using a rouge certificate in Man in the Middle Man attack if the users are not paying attentions to the certificate warning message and end up entering their credentials. The attacker in this case got the user credentials using social engineering attack and these can be used

against corporate network to launch further attacks.

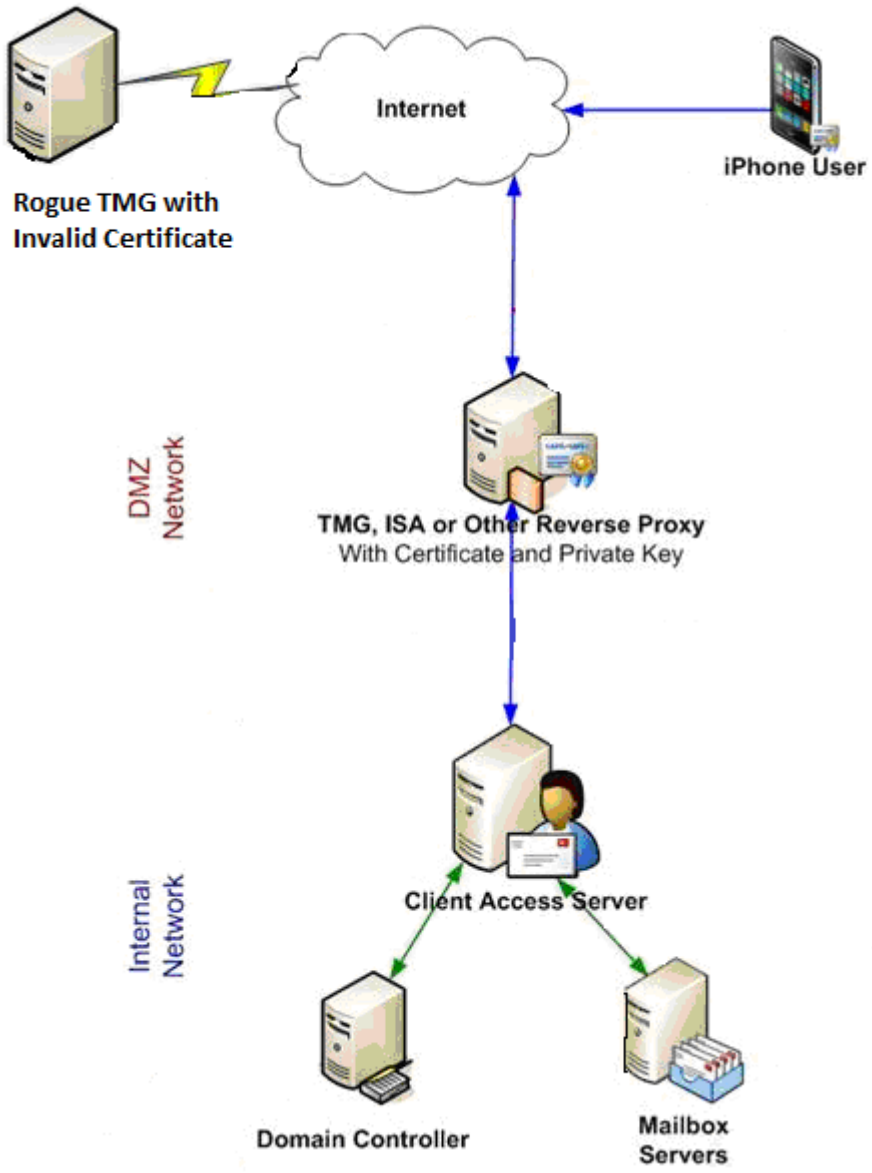

Figure 18: Rogue TMG Attack against ActiveSync

Figure 19: Social Engineering Attack using Invalid Certificate

The ActiveSync service is usually published using a reverse proxy server like Microsoft TMG server. This is a single point of failure and attack surface on the underlying OS is pretty large. Most small to medium size companies join this server to their LAN using multi homed setup and in some cases its even joined to Active Directory domain for the ease of management and for using additional features. Now if this server is comprised, this could result in your entire LAN environment to be compromised.

ActiveSync is based on SSL/TLS and the growing usages of SSL attackers are increasingly targeting the SSL layer.

Over the past years, attackers have repeatedly compromised various CA organizations. These include, DigiNotar, GlobalSign, StartSSL, Comodo and Digicert Malaysia. These attacks were a

direct consequence of the commoditization of certificates, where smaller, less competent organizations have started to obtain a bigger share in the Certificate Authority market. As it stands now, any CA can issue a digital certificate for any application without any required consent from application owner. A hacker, who gains control on any CA, can then use it to issue fraudulent certificates and impersonate any website. Additionally, there are concerns that some root CAs whose trust is hardcoded into browser software are inherently dubious.

Application certificates are no longer limited to being stored by the application. This is the consequence of the monolithic nature of SSL. While SSL prevents access to traffic by attackers it has no built-in mechanisms that restrict access to it by collaborative 3rd parties. For example, proxies, load balancers need to access the certificate's private key in order to access application data. In these cases, it would be preferable that the intermediate proxies would be able to look at message headers, or to be able to read traffic without changing it. However, this granularity is not supported by SSL. As a result, the digital certificate is now stored in many locations – some residing outside of the site's physical environment and out of the application's owner control. These open up additional attack points which provide higher success rates for attackers. This applies to Microsoft TMG server as well. It's required to export the SSL certificate from the Exchange client access server and import it on TMG server in DMZ for ActiveSync to be published. This results in private key being on the server in DMZ and if there are any underlying vulnerabilities in Windows OS, this could pose a significant threat.

The heavy computational burden incurred by the SSL-handshake process leaves SSL protected resources prime candidates for effective Denial of Service (DoS) attacks. Microsoft TMG server requires incoming port 443 to be open so you are looking at similar attack surface as with a typical web server.

18. Securing ActiveSync Architecture

Use certificate based authentication for ActiveSync. The certificate should always come from a reputable CA.

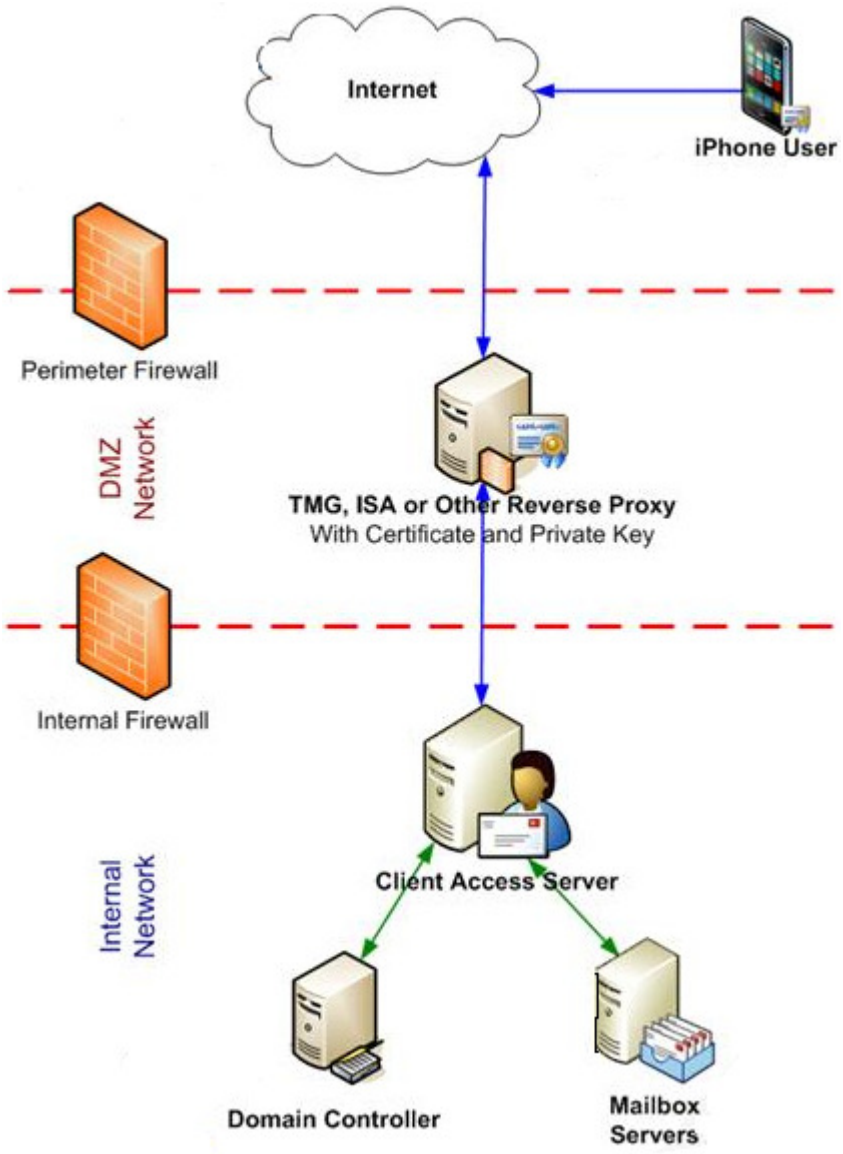

Figure 20: ActiveSync Secure Topology

Place reverse proxy server like TMG in DMZ using a sandwiched firewall/IPS archtecuture. Vyatta software firewall can be used for this purpose. Use access control list between TMG server and internal servers as TMG server only needs access to Exchange and DC servers only.

19. Comparison of ActiveSync and BES

- **Application Settings:-**

Capability	**ActiveSync using iPhone**	**Blackberry Enterprise Solution**
Remote Wipe	Remote wipe for iPhone is available through: Exchange2007/2010Management Console, Outlook Web Access ,Exchange ActiveSync Mobile Administration Web Tool Wipes take approx 1 hour per 8GB of data on the device. Remote wipes of an encrypted device (iPhone 3GS or later) will first remove the 256-AES encryption key, making the encrypted data immediately inaccessible.	Remote wipe is a core feature of the device and can be triggered from the BlackBerry Enterprise Server (BES).Wipe times and implementation are comparable to the iPhone.
Encryption	iPhone has hardware encryption which is also enabled by enabling the ActiveSync option.AES256 employed by default. Pre-3GS devices do not provide encryption. Rumored encryption bypass vulnerabilities, some of which are true, all require the iPhone to be already jail-broken.	BlackBerry devices provide encryption and policy from the BlackBerry Enterprise Server (BES). The implementation is trusted and validated by many Government organizations.
Provisioning	Requires local configuration	Remote and local provisioning
Remote Restore	No	Yes
Password	Passcode rules (passcode strength and length, minimum number of complex characters, passcode age, auto-lock timer, passcode history, grace period for device lock, and maximum number of failed attempts).Out of the box, the iPhone uses a 4 digit passcode.	Rich policy management of password complexity comparable to a desktop environment.

- **Policy Management:-**

Capability	**ActiveSync using iPhone**	**Blackberry Enterprise Solution**
Enforcement	Can deploy profile to an iPhone, requires password authentication to override.	Can set strong controls for the device with a per-option policy on what can and cannot be edited users. Flexible but rather complicated.
Patching	Patches cannot be pushed over the network automatically. They are deployed by iTunes when the user connects and opts to update. iTunes can not be centrally managed, so an admin can not dictate deployment of the patch. Reliant on user awareness.	Patches can be deployed remotely from the BlackBerry Enterprise Server (BES). Full patch audit status and management can be executed remotely without user intervention required. Where a reboot is required policy can drive whether the user is given a choice to delay to a later time.
Auditing	No,management functionality does not yet exist to be audited.	Yes, based on policy. Logged locally and centrally.
Updates	Updates to profiles are not automatically deployed and require the user to decide to apply the policy.	Updates can be pushed from the BES remotely and applied in the background without user interaction.
User Rights Management	By default a set of restrictions are provided: explicit content, Safari, YouYube, iTunes, allowing a user to install apps, allowing use of camera, screen captures.	Granular policies can be set, though they are somewhat intimidating.
Application Control	Profiles can be set to restrict deployed applications. This occurs at bootstrap, the administrator cannot deploy additional applications remotely. In-house applications can be developed for the device, which, in turn, are checked and signed by Apple.	BES provides the ability to package and deploy specific applications remotely. Restrictions can be placed on applications that the user is allowed to install to a corporate white list or for administrator deployment only. Extremely granular.

- **Transport Security:-**

Capability	**ActiveSync using iPhone**	**Blackberry Enterprise Solution**
VPN	Yes, password- and token based authentication available. Requires user interaction.	Provides encrypted tunnel back to BlackBerry Enterprise Server (BES) for data transfer also supports explicit VPN.
Email	Yes, supports SSL across a range of protocols. Depends on server configuration (Exchange by default). Also allows the use of Authentication Certificates.	Yes, by default through encrypted tunnel or directly to mail host.
Certificate	The iPhone includes the ability to use a SCEP server to control the issuing and revolcation of certificates to the device. It is also possible to create a profile containing certificates separate to using the SCEP server.	Can manage certificates and deploy remotely.
Proxy	Proxy can be set up for the Carriers Access Point. All traffic can also be forced through the VPN, which can be configured to go through a proxy.	Can set proxy policy at the BES level (routing all traffic back to the central server and routing through a single point). Can also set per-connection policies.

20. Vyatta Introduction

Vyatta came together in 2005 to develop an open-source alternative to traditional routers. By 2006 Vyatta released its Open Flexible Router that included a software-based router/firewall/VPN product. The system is a specialized Debian-based Linux distribution with networking applications such as Quagga, OpenVPN, and many others. A standardized management console, similar to Juniper JUNOS or Cisco IOS, in addition to a web-based GUI and traditional Linux system commands, provides configuration of the system and applications. In recent versions of Vyatta, web-based management interface is supplied only in the subscription edition, however, all functionality is available through KVM, serial console or SSH/telnet protocols. The software runs on standard x86-64 servers. Today Vyatta offers three editions, Vyatta Core, Vyatta Subscription Edition, and Vyatta Plus.

- **Vyatta Core**

Vyatta Core is the open-source version of Vyatta. It is free to download and use however there is no commercial support. Vyatta does host a community with documentation and a support forum where members and Vyatta employees assist with questions and discussions. Many of the features Vyatta offers come from the community. The Vyatta Core's main offerings are IPv4 and IPv6 routing, stateful firewall, IPSec and SSL VPN, and intrusion prevention. Vyatta Core only supports Ethernet interfaces. WAN interfaces support such as DSL, T1, or T3 require a Vyatta Subscription Edition license.

- **Vyatta Subscription Edition**

Vyatta Subscription Edition offers everything Vyatta Core offers plus some advanced features such as: an API for centralized management, WAN device support, and TACACS+. The Vyatta

Subscription Edition is commercially supported by Vyatta and is eligible for professional services consulting. Vyatta Subscription Edition also provides access to a customer only knowledge base. However Vyatta will not support the Subscription Edition when Vyatta Core is in use within the same organization. This means that if an organization requires enterprise support then all Vyatta installations must be Subscription Edition.

- **Vyatta Plus**

Vyatta Plus is an add-on service for Vyatta Subscription Edition. Currently the only Vyatta Plus service available is VyattaGuard web filtering. VyattaGuard is an enhancement to the web filtering offered in Vyatta Core. VyattaGuard is to further reduce web-based threats and manage bandwidth consumption.

Running Room is currently using Linux iptables firewall as its parameter firewall. There These are getting outdated and there is need for a software based low cost firewall with following features –

- Software should be open source
- Capable of IPS, web filtering, firewall and VPN support all in one box
- Clustering and failover capabilities
- Web based management
- Configuration synchronization ability
- Deployable in VMware as a virtual machine

Vyatta seemed to have met most of these requirements so it was decided to do further testing.

21. Running Room Test Environment for Vyatta

The following test environment was used to test Vyatta for its feasibility study in Running Room network. We are looking to deploy it as Firewall/IPS for Microsoft TMG solution for ActiveSync, web filtering and backup VPN server.

- VMware ESX 4.1 servers – An isolated vSwitch was used for testing Vyatta
- Vyatta Core 6.3 – Vyatta was deployed as virtual machines in test network
- Microsoft Exchange 2010
- Microsoft Forefront Threat Management Gateway (TMG) 2010

22. Vyatta Intrusion Prevention System

An Intrusion Prevention System (IPS) monitors network and system activities for malicious or unwanted behavior and can react, in real time, to block or prevent those activities. Network-based IPS operates in-line to monitor all network traffic for malicious code or attacks. When an attack is detected, the system can drop the malicious packets while allowing all other traffic to pass. The Vyatta system uses the Snort engine for intrusion detection and prevention. Snort can perform protocol analysis and content searching/matching. It can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes and OS fingerprinting attempts. An open version of the IPS for the Vyatta Core was used for testing.

The following steps were taken to configure IPS on Vyatta in the test environment:-

- After obtaining oinkcode from snort.com, download snort rules. This is to be done on both Vyatta firewalls sandwiching Microsoft TMZ server.

  runroom@DMZ1:~$ update ips rules oinkcode < >

  runroom@DMZ2:~$ update ips rules oinkcode < >

- Setup Vyatta to retrieve rules updates automatically at 5AM in the morning.

runroom@DMZ1# set content-inspection ips auto-update oinkcode < >

runroom@DMZ1# set content-inspection ips auto-update update-hour 5

runroom@DMZ2# set content-inspection ips auto-update oinkcode < >

runroom@DMZ2# set content-inspection ips auto-update update-hour 5

- Define the internal network.

runroom@DMZ1#set content-inspection ips modify-rules internal-network <Running Room DMZ and internal network for Vyatta firewall facing Internet>

runroom@DMZ2#set content-inspection ips modify-rules internal-network <Running Room Internal network for Vyatta firewall connecting to LAN>

- Setup to inspect traffic flowing from one zone to another. The test setup needs three Zones defined; Internet, DMZ and Private

Internet_Running – Public zone facing public internet

DMZ_Running – DMZ zone sandwiched between internet and private LAN at Running Room, this will have Microsoft TMG server

Private_Running- Private zone with servers like domain controller and exchange client access server

runroom@DMZ1# set zone-policy zone DMZ from Internet content-inspection enable

runroom@DMZ2# set zone-policy zone Private from DMZ content-inspection enable

- Specify all traffic within a given flow for inspection

runroom@DMZ1# set content-inspection traffic-filter preset all

runroom@DMZ2# set content-inspection traffic-filter preset all

- Apply a balance of enabled rules

runroom@DMZ1# set content-inspection ips policy balanced

runroom@DMZ2# set content-inspection ips policy balanced

23. Vyatta Firewall

Firewall functionality analyzes and filters IP packets between network interfaces. The most common application of this is to protect traffic between an internal network and the Internet. Vyatta's advanced firewall capabilities include stateful failover, zone-based firewalling, and time-based firewalling.

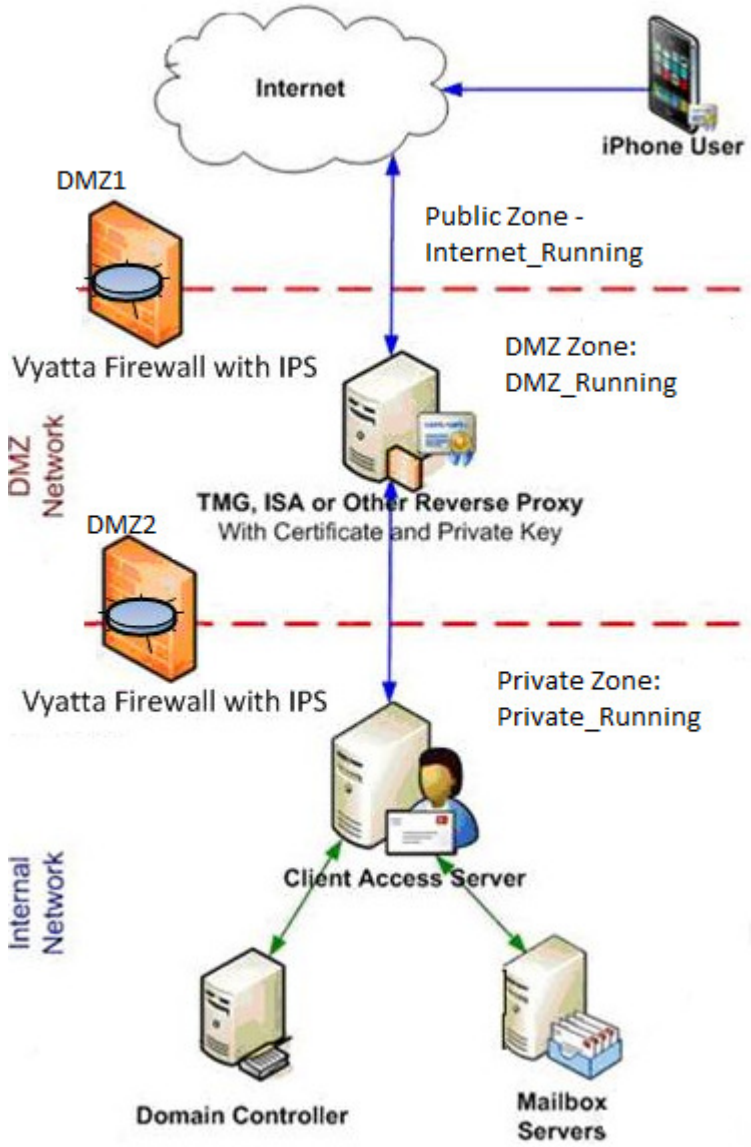

Figure 21: Running Room ActiveSync Topolgy using Vyatta Firewall and IPS

ON DMZ1 Firewall:-

- Create of zone policies

  runroom@DMZ1# set zone-policy zone public description “Internet_Running”

  runroom@DMZ1# set zone-policy zone dmz description “DMZ_Running”

- Create rule set for traffic to Internet_Running Zone

  runroom@DMZ1# set firewall name to_Internet_Running description

  "filter traffic entering public zone"

  runroom@DMZ1# set firewall name to_Internet_Running rule 1 action

  accept

  runroom@DMZ1#set firewall name to_Internet_Running rule 1 destination address <IP Addresss of TMG in DMZ>

  runroom@DMZ1# set firewall name to_Internet_Running rule 1 destination

  port 443

  runroom@DMZ1# set firewall name to_Internet_Running rule 1 protocol

  tcp

- Apply the rule set to Internet_Running Zone.

  runroom@DMZ1# set zone-policy zone dmz from public firewall name

  to_Internet_Running

ON DMZ2 Firewall:-

- Create zone policies.

  runroom@DMZ2# set zone-policy zone dmz description “DMZ_Running”

  runroom@DMZ2# set zone-policy zone private description “Private_Running”

- Creation of rule sets for traffic to Private_Running from DMZ_Running Zone.

  runroom@DMZ2# set firewall name Private_Running_from_DMZ_Running description

  "filter traffic entering private zone from dmz zone"

runroom@DMZ2#set firewall name Private_Running_from_DMZ_Running rule 1 action accept

runroom@DMZ2#set firewall name Private_Running_from_DMZ_Running rule 1 state established enable

runroom@DMZ2#set firewall name Private_Running_from_DMZ_Running rule 1 state related enable

runroom@DMZ2# set firewall name Private_Running_from_DMZ_Running rule 10 action accept

runroom@DMZ2#set firewall name Private_Running_from_DMZ_Running rule 10 destination address <IP Addresss of Domain Controller in Server Farm>

runroom@DMZ2#set firewall name Private_Running_from_DMZ_Running rule 10 destination address <IP Addresss of Exchange 2010 CAS in Server Farm>

runroom@DMZ2# set firewall name Private_Running_from_DMZ_Running rule 10 destination port 443,88

runroom@DMZ2# set firewall name Private_Running_from_DMZ_Running rule 10 protocol tcp

runroom@DMZ2# set firewall name Private_Running_from_DMZ_Running rule 20 action accept

runroom@DMZ2#set firewall name Private_Running_from_DMZ_Running rule 20 destination address <IP Addresss of Domain Controller in Server Farm>

runroom@DMZ2#set firewall name Private_Running_from_DMZ_Running rule 20 destination address <IP Addresss of Exchange 2010 CAS in Server Farm>

runroom@DMZ2# set firewall name Private_Running_from_DMZ_Running rule 20 destination port 464

- Create of rule sets for traffic to DMZ_Running Zone from Private_Running for mangement purposes

runroom@DMZ2# set firewall name DMZ_Running_from_Private_Running description

"filter traffic entering dmz zone from private zone

runroom@DMZ2# set firewall name DMZ_Running_from_Private_Running rule 1 action accept

runroom@DMZ2# set firewall name DMZ_Running_from_Private_Running rule 1 destination

port 443

runroom@DMZ2# set firewall name DMZ_Running_from_Private_Running rule 1 protocol tcp

- Apply the rule set to DMZ_Running Zone

  runroom@DMZ2# set zone-policy zone private from dmz firewall name Private_Running_from_DMZ_Running

- Apply the rule set to DMZ_Running Zone for mangement purposes

  runroom@DMZ2# set zone-policy zone private from dmz firewall name DMZ_Running_from_Private_Running

24. Vyatta Web Filtering

The Vyatta system can be configured to act as a web proxy server providing both web caching and web filtering functionality. Web filtering is an important tool for managing web access to reduce exposure to web-based threats, to limit legal liabilities by blocking objectionable content, to increase productivity, and to manage bandwidth usage. The Vyatta system provides basic web filtering services as part of the Vyatta Core. VyattaGuard enhanced web filtering is available as a Vyatta Plus service. Transparent Mode was used for testing which requires no special configuration in the browser at user workstation.

Running Room had layer 2 fiber connection between Edmonton office and Calgary datacenter. Edmonton office has no internet exit point, all internet traffic flows via Calgary data center. A hardware based 3COM device was being used in Calgary datacenter. The IP subnet was same in Edmonton office and Calgary data center, this solution was working fine with web proxy gateway in Calgary. The client computers requiring web filtering were assigned 3COM firewall IP address as their default gateway.

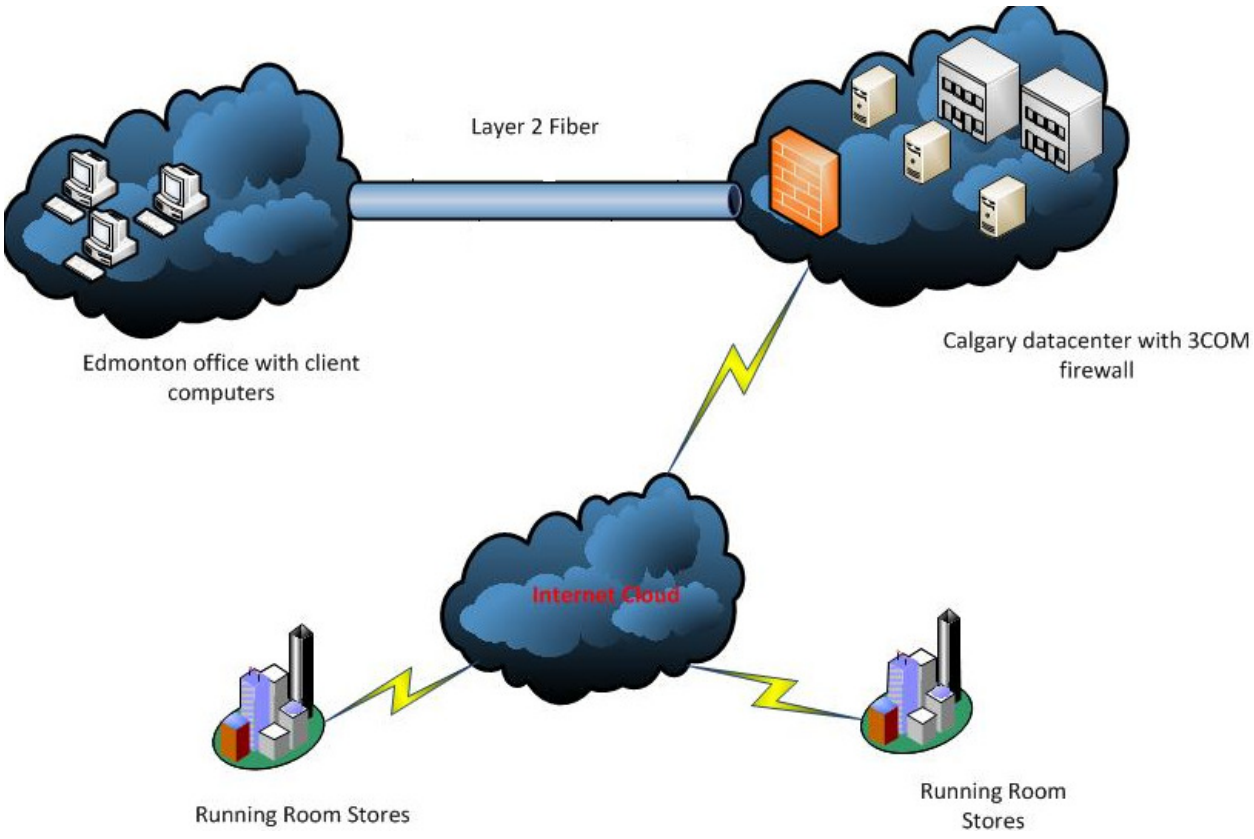

Figure 22: Running Room Web Filtering Setup before MPLS Connection Upgrade

Recently the connection between Edmonton and Calgary was upgraded to L3 MPLS which required different subnets in both locations. This posed a problem as default gateway had to be a local address in Edmonton office subnet. The temporary solution was to use Symantec End Point policies for URL filtering on the client computers in Edmonton office but it was slowing down the performance. There was a need for software based low cost web proxy solution which could synchronize its configuration with its counterpart in Calgary data center. Vyatta was selected as one of the candidate solution for testing.

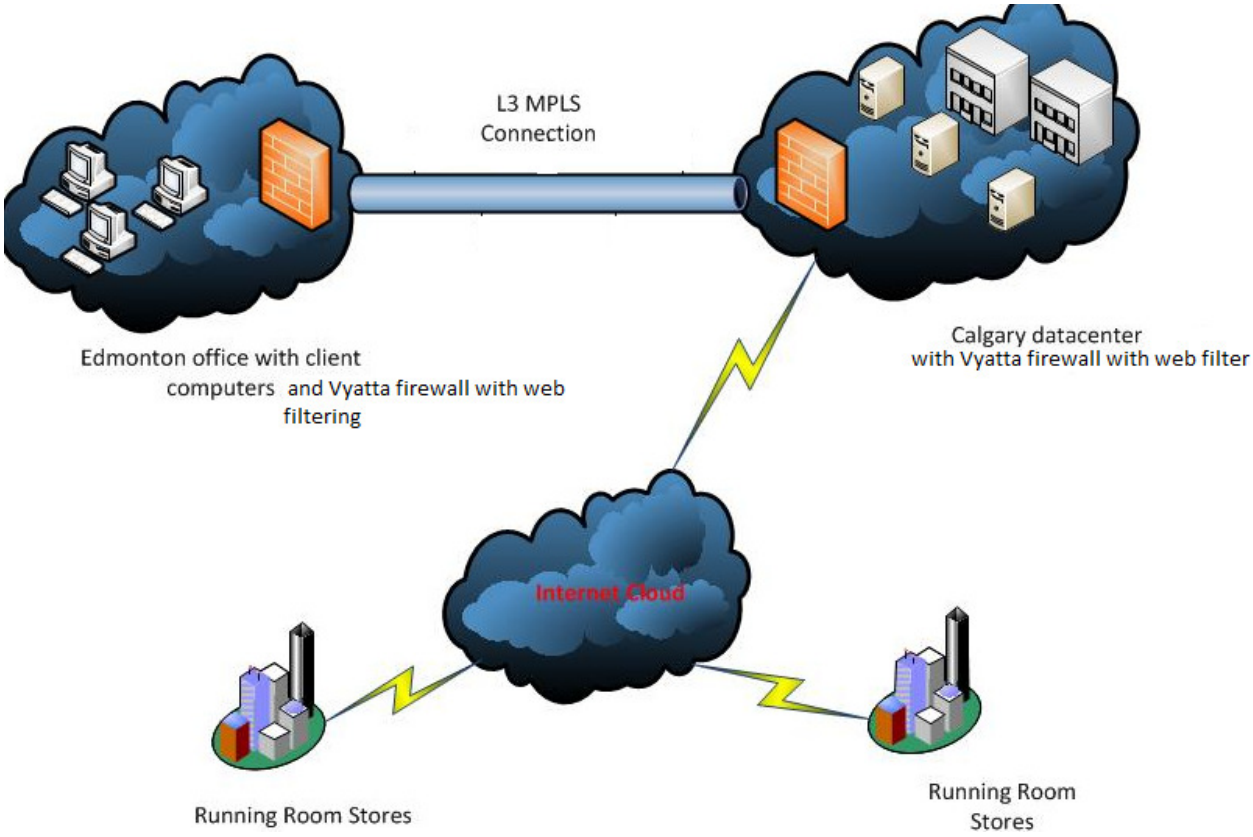

Figure 23: Running Room Web Filtering Setup after MPLS Connection Upgrade

The following steps were taken to configure Vyatta web filtering:-

- Setup the address to listen for requests.

  runroom@RRFW1# set service webproxy listen-address <LAN Facing Address>

- Redirect blocked requests to Running Room internal support system.

runroom@RRFW1# set service webproxy url-filtering squidguard redirect-url “http://rrsupport”

- Define a group for Running Room Stores using IP subnets.

  runroom@RRFW1# set service webproxy url-filtering squidguard source-group STORES address <Running Room Stores Subnet>

- Define a group for Running Room Server Farm using IP subnets.

  runroom@RRFW1# set service webproxy url-filtering squidguard source-group SERVERS address <Running Room Server Farm Subnet>

- Define a group for Running Room Home-Office users using IP subnets.

  runroom@RRFW1# set service webproxy url-filtering squidguard source-group HOMEOFFICE address <Running Room HO Subnet>

- Create the rule to filter requests from the SERVERS group. Nothing gets filtered for this group.

  runroom@RRFW1# set service webproxy url-filtering squidguard rule 10 source-group SERVERS

- Create the rule to filter requests from the STORES group. Block URLs by explicitly specifying them using the local-block option.

  runroom@RRFW1# set service webproxy url-filtering squidguard rule 20 source-group STORES

  runroom@RRFW1# set service webproxy url-filtering squidguard rule 20 local-block <Specify URLs>

- Create the rule to filter requests from the HOMEOFFICE group. Block URLs by explicitly specifying them using the local-block option.

  runroom@RRFW1# set service webproxy url-filtering squidguard rule 30 source-group HOMEOFFICE

  runroom@RRFW1# set service webproxy url-filtering squidguard rule 30 local-block <Specify URLs>

25. Vyatta Test Environment Results and Exclusions

Vyatta lived up to its expectations when it came to firewall and IPS features but its lacking few features in web filtering and remote access arena. Vyatta supports configuration synchronizing in Master Standby setup. As per the requirements of Running Room network, I was hoping to set it up in Master Master setup with list of blocked URLs synchronizing between two firewalls. This requires further research and working with Vyatta technical support to see if it's doable. However for the test environment I skipped this part. Vyatta does not support filtering based on LDAP groups however this limitation can be overcome using source IP address based filtering available in Vyatta. I used Web filtering feature that is available as part of the Vyatta Core system for testing which provides access to a list of filtering categories in a community-updated blacklist. If Running Room proceeds to go with this solution then VyattaGuard advanced web filtering which is available as a subscription-based Vyatta Plus service should be tested and deployed. For Vyatta IPS i used free snort service avialble with Vyatta core. Vyatta offers the subscription-based Vyatta PLUS Snort VRT Service which should be used if this is to be deployed in production network at Running Room.

Running Room is using Juniper SA 700 SSL based VPN devices for remote users. We were looking for a backup VPN server in case Juniper fails. Vyatta's remote access VPN feature is based on SSL/TLS security which is comparable to Juniper device. However its lacking a few critical functionality features which are needed as followings:-

- Vyatta does not support different login realms based on Active Directory group memberships.

- It does not support restricting access to certain hosts in our corporate networks based on VPN group memberships. This feature is critical as remote users are not allowed to access certain assets in corporate networks.
- Vyatta VPN does not support Single Sign On (SSO)) for Microsoft Sharepoint and Exchange Outlook Web Access. This is critical for remote users at Running Room.
- Vyatta needs a client software on remote user stations which is not very user friendly for the end users. Juniper SSL VPN solution on the other hand support any web browser as its VPN client.

Given its limitations, we decided to proceed with setting up Juniper SSL VPN solution in a failover setup by using two Juniper SA 700 VPN devices.

26. Conclusion

The security features differ from one mobile operating system to the next. Some mobile operating systems are more ready for the enterprise than others in terms of end-to-end security features, but each has its unique benefits. Organizations should not determine which mobile device has the strongest security features and then, settle on that one for the entire organization because that view would be too narrow. Instead, the organization should have a separate security policy ready for each mobile device expected within the enterprise. The refusal to have a security solution for each device expected in the enterprise may mean that the corporate data is walking away in an unsupported and uncontrolled fashion. Therefore, making the choice not to support a device is much more risky. When compared against the other mobile platforms, BlackBerry is the clear leader in on-device security and manageability. The BlackBerry Enterprise Solution has been approved for storing and transmitting the data by North Atlantic Treaty Organization (NATO) as well as government organizations in the United States, Canada, the United Kingdom, Austria, Australia and New Zealand. The BlackBerry Enterprise Solution is the first wireless platform to earn Common Criteria EAL 4+ certification. When it comes to the user functionality, iOS devices are ahead of Blackberry. However, the security implementation of iOS devices in the corporate environment using ActiveSync is still in its infancy and need more time to mature. Android is good for personal use but offers less centralized control to administrators when it comes to the security features. There is need for good third party Mobile Device Management (MDM) solutions to manage all these different devices in the enterprise in a secure way.

Vyatta offers a low cost security suite compared to other industry expensive solutions. It does not offer all the features which a high end security solution might offer. However, it is a great alternative for small to medium size organization like Running Room Canada with a low IT

budget. Vyatta can also be implemented easily in the virtualized data center lowering the power and space cost in the data center environments. As per my recommendations, Vyatta is good security solution for the organization with cost effective intermediate security needs.

27. References

1) Blue Coat Systems. (2008). Technology primer: Secure sockets layer (SSL), 1-20. Retrieved from http://www.bluecoat.com/

2) Dwivedi, H., Clark, C., & Thiel, D. (2010). *Mobile application security*. (pp. 220-337). New York: The McGraw-Hill Companies.

3) Guérin, N. (2008). Security policy for the use of handheld devices in corporate environments copyright. *SANS Institute Reading Room*, 2-60.

4) IceWarp Unified Communications. (2011). Exchange activesync guide. 2-56.

5) Imperva. (2011). Hacker Intelligence Intiative, Montahly Trend Report (6), 3-9. Retrieved http://www.imperva.com/

6) Juniper Networks. (2011). Securing Today's Mobile Workforce, 4-6.

7) Lopez, M. D. (2009). Successful mobile deployments require robust security. *Lopez Research LLC*, 1-9. Retrieved from http://www.lopezresearch.com/

8) Mah, P. (2011). Reconsider deploying the iphones, ipads with exchange server for now. *TheEmailAdmin*,1-4. Retrieved from http://www.theemailadmin.com/2011/02/reconsider-deploying-the-iphone/

9) Nachenberg, C. (2011). A window into mobile device security. 1-23. Retrieved from http://www.symantec.com/

10) Net-Scale Technologies. (2009). Does exchange Active Sync (EAS) Solve All your Mobile Data Needs? (1.0), 2-13. Retrieved from http://www.net-scale.com/

11) O'Connor, J. (2007). Attack surface analysis of blackberry devices. *White Paper: Symantec Security Response*, 7-33.

12) Research In Motion Limited. (2011). Blackberry enterprise server 5.0 sp3 and blackberry 7 version: 5.0. *Security Technical Overview*, 3-167.

13) Research In Motion Limited. (2004). Blackberry wireless enterprise activation release (4.0), 3- 15. Retrieved March13, 2012: http://www.blackberry.com/

14) Schiffman, J. (n.d.). Blackberry security model report 3. *Systems and Internet Infrastructure Security*, 1-18.

15) Todd, J. (2010). Deploying a vyatta core firewall. *The SANS Institute*, 10-15. Retrieved http://www.vyatta.org/

16) Trend Micro. (2007). Protecting mobile data and increasing productivity. 3-14. Retrieved from http://www.trendmicro.com/

17) Wallach, D. (2011). Smartphone security: Trends and predictions. *Rice University.* 1-10

18) Vyatta System. (2012). Firewall reference guide. 15-237. Retrieved from http://www.vyatta.org/

19) Vyatta System. (2012). High availability reference guide. 24-256. Retrieved from http://www.vyatta.org/

20) Vyatta System. (2012). Security reference guide. 16-224. Retrieved from www.vyatta.com

21) LasD'Aguannot. (2006). *Blackjacking – 0wning the enterprise via blackberry*. Paper presented at Defcon® hacking conference. Retrieved from www.defcon.org