LESSONS LEARNED IN THE GLOBAL DEPLOYMENT OF AN OPEN SOURCE SECURITY SOLUTION

Barry Irwin

Rhodes University, Department of Computer Science

b.irwin@ru.ac.za

P O Box 94

Grahamstown

6140

Tel: +27 46 603 8626

ABSTRACT

This paper covers the lessons learned, and the challenges facing the deployment of an Open Source derivative firewall and Virtual Private Network (VPN) solution into the established commercially driven Telecommunications arena. The lessons learned and issues discussed are drawn from the author's own experiences having worked for a global Wireless Application Service Provider over a three year period. Focus is placed on the issues surrounding the integration of the open source equipment with that of the established global Telecommunications players, where compliance to existing network standards was a requirement for connectivity.

Major stumbling blocks to the acceptance and success of the open source product were the concerns expressed by the Telecommunications Operators relating to interoperability and troubleshooting facilities when interfaced to their existing commercially available equipment and existing rigid telecommunications networks. The processes resulting in the initial decision to utilise an open source solution in preference to commercial offerings are also explored. An open source solution was found to offer higher flexibility and functionality and greater return on investment whilst maintaining a significantly reduced cost in comparison to commercial solutions available.

The issues surrounding the development, deployment and the challenges of ongoing maintenance are addressed. It was found that the very flexibility which necessitated the use of the open source product was also often a cause of frustration. The periodic review process as to the continued satisfaction of the underlining business requirements by the developed system is also covered.

KEY WORDS

Open Source, Security, Firewall, VPN, FreeBSD

LESSONS LEARNED IN THE GLOBAL DEPLOYMENT OF AN OPEN SOURCE SECURITY SOLUTION

1 INTRODUCTION

The development of advanced open source operating system such as Linux, FreeBSD and OpenBSD in recent years has opened up opportunities for high quality security systems to be effectively built from available 'off the shelf' components. These resultant systems are able to offer comparable performance and features and a substantially reduced cost to existing security offering from commercial players such as Check Point and Cisco. Consequently an increasing number of organisations, particularly in the SME market, are re-evaluating the potential savings investment in such a solution could potentially offer. In the author's experience this has been met with a mixture of apprehension and resistance, particularly by the long established players in the global telecommunications market.

This paper explores the reasons for this lack of acceptance by telecommunications organisations and some of the pitfalls that an organisation adopting an Open Source solution may not be aware of. Experience is drawn from the author's involvement in integrating a customised internally developed solution for a global Wireless Application Service Provider (WASP) with Telecommunications companies and content providers in more than nine countries in four continents over a period of nearly three years. While much has been written on the cost benefits of open source versus proprietary solutions [1,2] little has been done when looking specifically at the cost of security applications.

The remainder of the paper will deal with the process and business related criteria that lead to the selection, and subsequent development of a customised open source solution for the WASP. Finally a number of key points are raised which provide a means to evaluate at a higher management level whether an organisation should make use of existing open source offerings, build their own solution, or purchase an existent commercial offering.

2 CHOICE OF AN OPEN SOURCE SOLUTION

The decision by the WASP to initially implement an Open Source solution was initially arrived at almost by default, based on a number of factors.

- At the end of 2000 the company was expanding its single South African operation into five other countries, all having highly competitive GSM service related markets, in preparation for listing on the London Stock Exchange. Cost at this stage was a major factor, as were the time for the procurement of a commercial solution and the extensive training that would be required.
- Systems staff were already familiar with open source software, with the primary production platform being Linux based. The choice of an Open Source offering for the firewalls was a natural choice, with FreeBSD being the final selection.
- The choice of FreeBSD as a firewall over Linux was based largely on the skill sets of the staff concerned but also in order to provide a heterogeneous environment. In the event of there being a kernel level exploit, the heterogeneous environment would add a measure of protection.

- FreeBSD has also been acknowledged as having a highly robust and scalable IP stack – both important considerations for the environment in which these systems were intended to be operating. [3]

Following the initial deployment the requirements for a security solution were re-evaluated. Once again an open source solution was selected. The reasons for the continued use of a non proprietary solution were similar to those originally put forward, but with the added benefit of flexibility for what at that time was (and remained) a very dynamic environment. In particular some configurations for interfacing with telecommunications companies required up to thirteen network segments protected by a pair of firewalls. The cost assuming a viable commercial solution could be found would have been prohibitive. Given these rather divergent requirements for high network port density, flexibility of configuration, scalability, and relatively low cost, all while keeping within as much of the systems staff skill-set as possible, in order to minimise delays in deployment, an Open Source solution began to look like the only viable solution.

2.1 Hardware platform

The platform that the software would run on was another concern. In some locations, lead time in the event of equipment failure was too long and the costs of maintaining spare components too high. The standard PC (Intel compatible *x86* family) architecture was the only one that provided the scalability and flexibility for designing solutions to satisfy the requirements. The fact that the solution could be run on a wide range of hardware from 'no-name' clone systems to high end Compaq, Dell and IBM servers was an added benefit. Where possible enterprise grade equipment was used, but with the knowledge that in the event of a disaster a much cheaper interim solution could be sourced and commissioned relatively quickly.

Regarding the hardware platform, the only real stumbling block was in the provision of drivers for support of network adapters and SCSI hard disk controller, particularly those offering some kind of raid support. A decision was made to attempt to standardise on Intel Network Interface Cards (NIC) wherever possible and to use the D-Link quad-port cards where higher port densities were required. Support for the Intel Gigabit network cards was added at a later stage through the use of a kernel module. RAID cards proved to be more of an issue, particularly with vendors such as Dell having similar named cards, requiring very different drivers. Once again the solution was to provide support for a limited range. In most instances RAID was not required, and standard ATA/IDE hard disk assembly was used. The cost of components was found to be significantly cheaper than equivalent proprietary solutions particularly when looking at high port density NIC. A proprietary solution from Cisco or Sun Microsystems could cost in excess of four times the equivalent standard PC components.

2.2 Selection criteria

The selection criteria used were based not only on the network architecture and connectivity requirements of the remote sites, but also on the WASP's centralised management of its security systems.

Remote manageability was a major factor, with particular attention being paid to the performance over low bandwidth connectivity. The command line based interface of FreeBSD was found to be ideal for use over 9600 baud GSM links or when international Internet connectivity was poor. Related to this was the need for the software to be able to be administered from a secure laptop in the event of an after hours problem. Several commercial offerings investigated required a dedicated centralised management station. Logging and reporting were another area where an open source solution was found to provide at least similar functionality to commercial offerings at a vastly reduced price tag. Logs could be collected, analysed, produced and processed with freely

available tools. Additional logging requirements were easily added through modification of the software used.

Customisability was another important factor due to the differing requirements for connecting to carrier and content providers around the globe. Most Telco providers would allocate a particular IP address range to be used for communications with their systems, often from one of the private IP address block [4]. Some form of Network Address Translation (NAT) was required to ensure that the various production servers could communicate with multiple Telco systems. Means of connectivity were just as varied and required support, ranging from Virtual Private Network (VPN) connections using Generic Routing and Encapsulation (GRE)[5], IP Security (IPSec)[6] and various combinations thereof, to direct Ethernet, ISDN and Frame Relay connections. The latter two were offloaded to a dedicated router, although direct support would have been possible given appropriate hardware.

3 DEVELOPMENT AND DEPLOYMENT

The solution selected by the WASP matured over the course of its three year lifetime, going through three distinct phases, discussed below. These developments were in response to direct requirements at the business level either as a requirement from a partner, or in order to be able to provide new services. The implementation of each of these new requirements brought about their own technical challenges.

3.1 Development

3.1.1 The Initial System

The first iteration of the Firewall was based on a stock FreeBSD installation. The installation was customised on a per site basis as it was installed. This fulfilled the needs of a rapid deployment, but resulted in serious manageability and scalability issues. These were as a result of the fact that site configurations differed widely, since the configuration had evolved during the deployment period. While the ad-hoc configurations fulfilled the requirements at the time, together with the problems experienced provided valuable insight as to what features were most important, and what was lacking in terms of management functionality and support for business requirements.

3.1.2 Advanced Networking

Having the benefit of several months' worth of hands-on data and feedback from management of existing units, as well as clear business related technical requirements; the second firewall iteration was produced. The primary requirements were support for Virtual Private Network (VPN) connections to Telco's and content providers, and increased standardisation and management capabilities. The majority of these would be using the then newly adopted IP Security (IPSec) protocol for providing a transport with confidentiality and integrity checking capabilities. Both of these requirements could have been satisfied by commercial offerings, an open source based solution was chosen for its flexibility and easy of adaptation and modification to meat the changing demands of the environment.

This development phase also satisfied a number of other key requirements including a uniform automated installation process, unified basic security policy and a more intelligent configuration system. Initial support was also added to provision for some basic redundancy in the event of system failure. A significant benefit of the move towards a more commercial grade offering was that the distribution and application of system patches and updates was also much simplified, with a standardised testing, staging and rollback framework put in place.

3.1.3 Failover and Reliability

The third iteration added support for automated failover and increased reliability, and ability to implement certain changes such as modification and debugging of Network Address Translation (NAT) rules with a minimised requirement for downtime. This was an enhancement of work done

previously allowing the re-loading and live modification of the translation tables. The stock FreeBSD NAT daemon [] requires a restart in order to be able to load modified rules – something which results in downtime not viable in a 'five nines' Telecommunications environment. Despite initial concerns regarding the feasibility of fully automated failover, it was found to be possible with little modification. Initially problems were found with sharing state information between the nodes, but this was easily dealt with. The actual failover detection and heart-beat system implementation was based around the use of the Virtual Router Redundancy Protocol (VRRP) [7] for detecting a failed state and switching a node between master and slave mode.

3.2 Deployment

The advent of a standardised platform meant that the system could be deployed remotely by relatively unskilled staff at data centres with minimal intervention. Initial network and security related configuration was based on a provided configuration file used at installation time. Post installation configuration and site customisation then completed by the skilled administrators from a central location – resulting in significant cost savings in terms of travel. A goal of the development process was to enable a fairly junior level technician or at worst a technically savvy member of staff to be able to initiate a remote installation or rebuild of a firewall in the event of a disaster. The result was an automated installation procedure that only required prompting for the location of a site-specific initial configuration file. A system installed in such a manner would start up in a secure manner, only allowing configuration from the system console, or from a pre-designated remote location. As sites grew and their requirements changed, changes were merged back into the bootstrap configurations, further minimising the need for remote tweaking post installation. This provided for a relatively robust disaster recover capability.

4 INTEGRATION AND OPERATIONAL ISSUES

Packet filtering firewalls, even those incorporating newer techniques such as stateful inspection, are well understood, and have been around in increasingly advanced forms for the last twenty years. As such very little integration trouble was experienced with the firewall side of the solution. The major technological issue experienced by the author when integrating with partners was in setting up IPSec based VPN connections. IPSec is a relatively new standard and systems and network administrators have not yet generally gained experience to the level they would otherwise have with normal IP based a communications. This was further complicated by the lack of easily available intuitive debugging tools, for diagnosis when problems occurred. An issue relating to IPSec is some of the idiosyncrasies of the interoperability of between equipment from different vendors – both proprietary and open source. It is interesting to note that the KAME IPSec stack [8] used by the FreeBSD and OpenBSD operating systems is the stack of choice, used by the VPN Consortium for validating interoperability conformance [8,9].

4.1 Debugging Tools

The lack of debugging tools was often a major stumbling block with both parties having to rely on arcane debug output from their respective systems, and resort to a trial and error method for iterating through possible mis-configuration issues. In this case all vendor implementations that the author has dealt with are defiantly at fault, providing very poor debugging output. It is envisaged that as the IPSec standard continues to mature and becomes more widely implemented and utilised and administrators become more familiar with the intricacies of the protocol, many of these problems will become less serious.

IPSec provided an additional challenge of a different sort. Once VPN links were in place it was found to be very difficult to monitor the contents of the packet payloads traversing these tunnels, due the very nature of the encryption used. Recently tools such of TCPdump [10] have incorporated a mode for the decryption of IPSEC Encapsulated Security Payload (ESP) [11] based

traffic, but this is at times not possible. The majority of commercial offerings encountered were unable to provide high quality debugging output in a usable format. TCPdump is a widely used tool for debugging of network traffic, allowing one to see what is actually transmitted 'on the wire' often proving key in helping detect Issues such as faulty NAT functioning, routing and firewall filtering problems.

The open source platform due to its generic operating system origins was also able to provide and execute a much richer selection of other debugging tools ranging from customised scripts and tools, to more generic tools for debugging DNS and other connectivity problems. In almost all cased the proprietary offerings were only able to offer a small feature set able to be executed on the appliance.

4.2 Flexibility

One scenario where the flexibility of an open solution is well illustrated is when a IP based round-robin load balancer was required. The Telco in question had a legacy piece of equipment that imposed a rate limit for SMS submission on a per IP address basis, however they were able to support a significantly higher rate than this device was able to allow. The solution was to install a custom written round-robin proxy server on the security gateway, which cycled through a pool of addresses, thereby providing an *n*-fold increase in throughput – all integrated transparently to the actual SMS submission application and receiver equipment. This is something which could have been potentially difficult to implement on a commercial closed system.

The flexibility inherent in an open source system was found to be a somewhat contentious feature. While proprietary commercial offerings are seen as a clearly defined bundle of feature and if a feature was lacking this was accepted. It was often viewed by management that if an open system did not have a feature it was a serious shortcoming. In a number of cases the perceived level of flexibility or adaptation of the open source solution far outstripped its actual practical ability. In one particular case a requirement was raised for establishing a VPN connection by tunnelling IPSec traffic over a GRE tunnel. While this is very much a 'belt-and-suspenders' type solution, with both GRE and IPSec offering encapsulation of their payloads, it is also not necessarily the most efficient. A problem was found with the manner in which the system kernel processed the interaction of GRE and ESP packets, which lead to much criticism, and ultimately the use of an external router to perform the GRE encapsulation.

4.3 Modification

By its very nature, an open source system exposes its code for public review. This proved to be of benefit when, following an upgrade of the Operating System, the interaction between the IPSec packet handler and the Network Address Translation process was found to have changed, resulting in packets not being handled correctly on the firewall in some very particular and specific configurations that had not been picked up in vendor testing. Since the source for both versions was available, it was relatively easy to track down the change and prepare a customised patch for the system kernel in order to change the handling to result in the previous desired outcome. A problem like this could have taken several weeks or longer to resolve with a commercial vendor, particularly if the number of systems affected is small.

As mentioned above, the availability and potential to deploy a wide variety of debugging and traffic analysis tools provided an advantage over commercial systems. In almost all cases, use of these tools with commercial systems, would have required separate monitoring systems, in addition to the firewall/security systems, something which is not always feasible for ad-hoc monitoring. The open source platform also provided a much richer toolset for running customised ad-hoc applications for testing or enhanced logging when trying to debug intermittent problems. Languages such as Perl and Python allowed for rapid development of tools where needed, something that would have been considerably more difficult on proprietary embedded platforms.

5 REVIEW CYCLE

As with most business processes, periodic review and evaluation of a security solutions are required. It was found that the use of the following metrics when reviewing the continued use of the in-house solution were most relevant. The overriding question to be answered at these review sessions was whether to continue building and maintaining the existing solution, or if it was more economical to migrate to a commercially available offering.

- The cost of continued development and support needs to be evaluated in terms of both direct monetary costs as well as the indirect costs of the time spent by systems staff acting in a (usually non-core) development role. The knock-on effect that these costs could have on other projects and operational activities, both short and longer term needs to be considered. This is also true of a commercial offering however much of the work in this scenario is offloaded onto the vendor, with the systems staff becoming consumers of information and product rather than producers. Timeframes for development and implementation of features should also be compared against the implementation or conversion costs of purchasing a solution already offering these.
- The level to which the current solution satisfies both the current and longer term requirements both known and forecast; and to what extent changes must be affected in order to provide this support should also be quantified, costed and evaluated.
- Staff skill sets are another important factor when considering a change of product. The time and cost required for training should be considered. An additional cost may also be the requirements for external consulting services and decreased productivity during a ramp-up period post implementation. This applies both with the tooling-up of existing staff when migrating to a new system, be it an upgrade of an existent product or a radical shift to a new platform. Intakes of new staff will also incur training costs of some degree.

6 CONCLUSION

6.1 Critical Factors

Open Source solutions can offer a cost effective viable solution, however care should be taken to understand the limitations inherent in such an offering. An organisation that embarks on it's on in-house development or customisation of an open source based security products, in particular needs to be aware of the following:

- Time commitments for staff to continue with maintenance and development of the solution. This work is what would normally be performed by the product vendor and the internal staff would only be concerned with the application and testing of updates
- A reasonable set of limits should be placed on the intended scope of the functionality of the solution. Just because one arcane piece of hardware is desired to be supported, and this could potentially be satisfied by the open source solution, doesn't mean it should be. Where special or non-common solutions are required, a business case analysis should be performed. Development of non component-based solutions takes time and can have other undesirable effects if not correctly managed.
- Training and documentation are two other essential components, particularly in a global environment. This material is usually provided by a vendor or third party training provider. With an in-house solution it needs to be developed – another time constraint which can place severe strain on systems staff when taken in combination with normal odat to day operational activities.

- The question of where traditional 'tier three' or vendor support for a product will be sourced from. There are arguments for and against the traditional commercial model in comparison with the plethora of self-help websites and mailing lists available for Open Source products. There is an increasing number of companies offering dedicated support on open source offerings. However if an open source product has already been heavily customised, the effectiveness of utilising such an organisation may be of limited value. In cases like this careful consideration should be given as to which route to follow. The underlying decision that needs to be made is where to go for help in the event of a major problem not able to be solved by one's organisational staff.
- Management needs to have a clear understanding of the limits of what is feasible, and the fact that this may differ significantly from what is possible given unlimited resources. In the author's experience, the majority of serious functionality issues addressed were related directly to management selling vapourware, and committing to unreasonable technical requirements in order to close a deal. Only once the deal was signed were systems staff (usually on both sides) alerted to the fact that requirements needed to be satisfied. The result was often drawn out communications between the technical staff of both parties trying to cobble together a solution that would work usually resulting in significant delays in project lunches. Managements view was often found to be that the open source solution was clearly lacking in capability.
- Following from the above, senior management buy-in and support of any kind of in-house development is imperative. While many managers may jump at the chance to save money through the use of free software, the other cost factors need to be borne in mind. However without senior management backing a solution of either kind is unlikely to attain its full potential. This backing is particularly important when evaluating accusations of inadequacy or malfunction from regional or branch offices. A strong backing will also ensure that sufficient resources are dedicated for the ongoing maintenance that is required as part of the natural evolution of a product.
- Scalability can impact heavily on the decision-making process. A solution that works well for a single small office is highly unlikely to scale to work efficiently across numerous sites, with marginally differing requirements in a global organisation. The opposite is also true, although to a lesser extent. Manageability is usually the key factor. In order to be able to administer in excess of 50 systems globally with only a small team of staff, the systems need to be well designed understood and maintained – something that is challenging enough on small clusters of local systems. Sufficient resources should be dedicated to fully understanding the potential impacts of this.
- The staff involved in the development, administration and maintenance of both open source and commercial systems are one of an organisations most valuable asset. Due care should be taken to ensure that in the event of a disaster or staff leaving, their skills can be replaced fairly quickly. As with all staff working with secure systems, care should also be taken to ensure that these individuals do not become a risk to an organisation – something particular to pay attention to in the development of an open source solution where system code could potentially be modified. The development work of these staff should be clearly detailed in an ongoing maintenance and project plan, and if at all possible periods of dedicated time set aside for this work to be completed. An ideal would be to have staff solely responsible for development, and a second component responsible for the administration of these systems. Unfortunately this is often not feasible.

In the case in which the author was involved, the decision to finally consider a commercial solution was motivated primarily by the near simultaneous loss of the two senior technical staff involved in the maintenance development and deployment of the existing solution. However the

system had also begun to age rather badly, and had lacked any significant updating due to other priorities arising – in short the organisation had outgrown the initial solutions in the face of new business developments and strategies. The loss of these staff spelled a death knell to the project, and provided the catalyst needed to provide a pressing need for the organisation to re-evaluate its security requirements and migrate to a commercial offering, despite the perceived cost barrier. It is interesting to note that even at the point where it was decided to be replaced, on a feature by feature comparison, the open source solution scored significantly higher than a number of commercial offerings.

6.2 The Final Word

An IT manager faced with the decision as whether to adopt an Open source solution, whether it is based on Linux, FreeBSD, or OpenBSD should carefully consider the advantages and potential pitfalls surrounding a solution. Each organisation will attach its own weighting to the various criteria used in the decision making process. In many small businesses and non governmental organisations, cost is often a significant factor, and 'free' solution can have a very attractive lure. Probably the single most important factor group to evaluate are what the staffing skill currently present are, and what the cost of training is likely to be for a new solution, and if the staff responsible were to leave, could replacements be found? Without skilled competent staff, an in-house solution could become a major security and maintenance problem, the same however is true of proprietary offerings, with the exception that vendors and external consulting groups are often able to assist in times of need.

7 REFERENCES

[1] Northwest Educational Technology Consortium: *Open Options: Total cost of ownership* Online: http://www.netc.org/openoptions/pros_cons/tco.html

[2] GeodSoft 2004: *Linux, OpenBSD, Windows Server Comparison: Total Cost of Ownership* Online: http://geodsoft.com/opinion/server_comp/tco.htm

[3] FreeBSD Project 2004 *About FreeBSD's Internetworking* Online: http://www.freebsd.org/internet.html

[4] Rekhter Y, Moskowitz B, Karrenberg D, de Groot G. J, Lear E 1996 : *RFC1918/BCP5 Address Allocation for Private Internets*. IETF Publication.

[5] Farinacci D, Li T, Hanks S, Meyer D, Traina P 2000: *RFC 2784 Generic Routing Encapsulation (GRE)*. IETF Publication.

[6] Kent S, Atkinson R 1998: *RFC 2401 Security Architecture for the Internet Protocol*. IETF Publication.

[7] Hinden R, Ed. 2004: *RFC 3768 Virtual Router Redundancy Protocol (VRRP)*. IETF Publication.

[8] KAME IPv6 Integration Project. Online: http://www.kame.net/

[9] Virtual Private Network Consortium (VPNC) IPSec interoperability testing. Online: http://www.vpnc.org/conformance-logos.html

[10] Tcpdump Online: http://www.tcpdump.org/

[11] Kent S, Atkinson R 1998: *RFC 2406 IP Encapsulating Security Payload (ESP)*. IETF Publication.