Proposals for a Revision of Kerberos When Run in Conjunction with the IPsec Protocol Suit

Dean Rogers
Berlin, Germany
dean@rogers.com.de

***Abstract*—IPsec is an end to end secure communication protocol operating in the IP layer of the communications stack. It secures bilateral communication interchanges between hosts by encrypting individual IP packets. Kerberos provides a means of authenticating abstract entities to each other by via a trusted third party. It does this by encrypting packages, or tickets, of information using keys, which have been retained from previous exchanges. This investigation considers how Kerberos' internal message structures may be adapted when run in an IPsec environment, possibly avoiding layered encryptions, and encryptions whose purpose is to certify identity.**

***Index Terms*—authentication, Kerberos, encryption, IPsec**

I. Introduction

The purpose of authentication is to establish a trust relationship between two parties, mostly for the exchange of confidential data. How one entity can successfully verify its identity to another for this purpose is a problem, which is difficult to solve with 100% reliability. In general, authentication involves proof of identity, by verifying at least one form of identification. This may be by trusted third party guarantee, direct measurement and comparison of a characteristic of the applicant to known values of who the applicant claims to be, or possession of Identity documentation which only the true applicant should possess. The three factors of authentication are evidence of something the applicant knows (such as a password), possession (like a Passport), or a physically unique characteristic (for example a fingerprint). How this problem is approached in the field of computer science remains a volatile subject.

At the everyday level, concern grows over the security of online payments for books, DVDs, music, flight tickets, and online banking etc. If credit card numbers could be easily intercepted during transmission, the potential for misuse would be immense. The need to secure electronic communications, and means of remote parties to achieve this secularly, is apparent to anyone involved in any kind of electronic communication today.

Network access to a computer requires authentication, requisite communications are mostly facilitated by the TCP/IP stack, and the methods of achieving security are naturally interrelated with this protocol. Facilitating the privacy of electronic communication IPsec was developed to provide encryption and integrity for messages (IP packets). Optionally it can also provide, in the sense that it can require, authentication services.

In distributed environments with multiple services available, and mobility of users between client machines, Kerberos provides a centralised authentication system for user and service. Mutual authentication is necessary to prevent compromise of data and or communications between users and/or services by rogue hosts (clients or services) being introduced into a network.

II. Prior Knowledge

Prior knowledge of subjects such as encryption algorithms, hash functions, and Internet Key Exchange, IKE [1], is assumed so that the paper may proceed at an appreciable pace. The interested reader is directed to suitable material [2].

III. Problem Statement

It can be observed that where the encryption systems integral to Kerberos were to be implemented over communication channels protected by encryption algorithms such as those provided by IPsec there appears to be a duplication of effort; where IPsec at the IP layer would further encrypt those Kerberos encrypted tickets passed to it; which would clearly be computationally wasteful. This could mean that in these circumstances some of the encryption built into Kerberos may be superfluous, and investigation may reveal that alternative message passing procedures between the entities could be more appropriate than seen in the present Kerberos implementation.

Those already familiar with Kerberos V5 internals may proceed directly to Section IX.

IV. Prelude

Computer security involves three objectives,

- Authenticity, or how one can prove to a conversation partner that one is who one claims to be, and conversely how can one verify the other parties identity
- Confidentiality, how can one ensure that only ones conversation partner can read a message when it is desired that its content not be revealed to third parties

Manuscript received May 30, 2013; revised September 13, 2013.

doi: 10.12720/ijoee.1.4.228-233

- Integrity, ensuring that messages are received intact in their original condition and have not been modified or otherwise tampered with during transit.

V. KERBEROS

Users are required to identify themselves by name, ID, and password to prove who they are, in other words authentication is performed before the use of network resources is permitted. Because of the risk impersonation this password should not fall into the wrong hands, here the excuse 'it was not me' would be invalid. The role of Kerberos is to protect network resources from unapproved access.

VI. A LITTLE HISTORY

The Kerberos concept was instigated as project Athena at Massachusetts Institute of Technology, MIT, in 1983. And was summarised in 1988 [3]. The intention was to integrate the University's Digital Equipment Corporation, DEC, minicomputers running Berkley 4.2 Unix into a unified network. At that time, each minicomputer supported several dumb terminals used by students. Networking the DEC's would enable full access to their files for students from any computer or terminal on campus. An early paper analysing the topic was published in 1990 [4].

It was developed to authenticate users logging into a workstation running appropriate client software. The system issued users with an encrypted 'ticket' by the Kerberos server. This ticket could only be decrypted with the users' password, and contained information for obtaining further tickets, which were necessary for accessing network services, each requiring a ticket.

The specification for V5 was formalised in 1994 [5], and updated in 2005 [6]. The legacy of Kerberos's original design brief that it needed to secure authentication messages across an insecure network is still evident in its architecture today.

VII. KERBEROS INFRASTRUCTURE

A certain amount of prior setup is required for a Kerberos system to function as intended. An Authentication Service, AS, provides authentication of users and clients for the realm, sometimes also known as a domain. A realm constitutes the boundary of connected clients and services for which the Kerberos system is authoritative. To facilitate this, the AS maintains a database, DB, of User IDs, Host IDs, collectively known as principals, and their associated passwords or secret-keys, which, for security reasons are stored as hash functions. The AS holds a copy of the TGS (*cf*) secret key, and those of all clients of the realm. The ticket it issues is called a Ticket Granting Ticket, TGT.

The Ticket Granting Service, TGS, is a service that checks client requests submitted to it have been validated by the AS, and returns to the client a ticket of authority to be used to validate itself with the requested SS (*cf*). The TGS holds its own secret-key, which it copies to the AS, and holds copies of secret-keys for each SS in its realm.

The Server Service, **SS**, is a service selected by A from possibly many requested by the User: though not exclusively, services can request service of each other. This service has its own secret key (SS secret-key) which it copies to the TGS.

The User/client, A, is the User we refer to as the person wishing to make use of networked resources, which are protected from unapproved usage by the implemented authentication protocol. Client software has to be in place to recognise the protocol being used in the realm to which they are attempting to connect; otherwise authentication cannot succeed.

VIII. KERBEROS V5 MESSAGE INTERCHANGE

Step One, the User logs into Client A with their ID and password. The client software then hashes the password, and sends a message to the AS including the clients ID, the ID of the TGS, and a request for timestamp settings, Fig. 1 Note the novel use of colour coding indicating the elements of message exchanges to illustrate the relationship of those elements to each other, during their passage through the message exchange process.

A → AS: $A+R_A+TGS+N_1+Tf$

Here: *the '+' sign indicates concatenation.*
R_A this is the Realm of the client
N is a nonce
Tf are client options that can be requested, from(start) till(expiration or time-to-live) rtime(renew till time request)

Figure 1. Client to authentication server.

The clients' ID is fundamental to the authentication process, its verification is essential; the Timestamp establishes that the message is not a replay of an earlier one. The ID of TGS is the indication to the AS that the client requests authentication (since there should only be one TGS within the realm, the AS should already know this.) The client can determine the ID-TGS by various means outside this discussion, for example it could be included in DNS information. Note that at this stage the SS-ID is not indicated.

Step Two, The AS looks up the User ID in its database, and if found, it then checks that the password hash stored matches the one sent to it. The user has authenticated to this stage of the procedure upon success; which does not yet allowed the use of network resources. To enable this, the AS returns a message to the client for further processing, Fig. 2.

The message passed back to the client contains two sets of encrypted data; one is a ticket for presentation to the TGS (it includes additional flags not relevant here:

omitted for simplicity), the other is session information validating the communication between client and TGS.

AS → A: $A+R_A+K_{TGS}(K_{ATGS}+R_A+A+IP_A+Tf)+K_A(K_{ATGS}+R_{TGS}+TGS+N_1+Tf)$

Here: *K_{TGS} is the encryption key of TGS R_{TGS} is the realm of TGS*
K_{ATGS} is a session key for A and TGS use
IP_A is the IP address of A in any format consistent with the realm
K_A is the clients' encryption key as previously discussed

Figure 2. Authentication server returns tickets to client.

Colour coding: green section denotes Ticket Granting Ticket; the orange shading denotes session information.

The TGT is a remit from the AS to the TGS to issue a ticket to A for granting access to an SS. It includes duration stipulation, IP address, realm identifier, ID of client, and a unique session key denoting the communication between A and TGS; all encrypted with the TGS key (in practice a hash of a value provided by the realm administrator). Facilitating single-sign-on the TGT is reusable by specifying different SS-IDs.

Session information includes the session key (large random number, generated by the relevant host), duration stipulation, realm and ID of the TGS (which should match the realm of A), and a nonce value (if this value is seen again the ticket will be discarded); all encrypted with the hash of the clients password.

Step Three: Using its own key, the client can decrypt and retrieve the contents of the session information, and data for connecting to the TGS. The TGT is stored for forwarding, Fig. 3.

A→TGS: $SS+Tf+N_2+K_{TGS}(K_{ATGS}+R_A+A+IP_A+Tf)+K_{ATGS}(A+R_A+T_{S1})$

Here: *N2 is a second nonce in the protocol message exchange*
T_{S1} is a timestamp

Figure 3. Client requests a ticket granting ticket.

The client is now ready to make a request to the TGS for access to the SS. It is of no concern how A, in complex networks, determines the ID of the relevant SS from possibly hundreds, whether the User selects it from a list, it is pre-configured in a profile, or via DNS. The request is made by sending a message consisting of the session duration, SS-ID, a new nonce, the ticket granting ticket saved from earlier, and an 'authenticator,' certifying the clients identity (note: this effectively means the client is certifying its own ID, but only they could do this). The authenticator consists of client-ID, and clients' realm, together with a time-stamp; all encrypted using the client-TGS session key recovered from the session information that the AS had encrypted with the clients key, which it holds in its DB.

Step Four: the TGS can now issue a ticket, which the client can present to the SS verifying to the SS's satisfaction that the client has been authenticated, Fig. 4.

TGS→A: $A+R_A+K_{SS}(K_{ASS}+R_A+A+IP_A+Tf)+K_{ATGS}(K_{ASS}+Tf+N_2+R_{SS}+SS)$

Here: *K_{SS} is the SS secret key*
K_{ASS} is the session key generated by the TGS for the sole use of A and SS
The pink encryption block is the Service Granting Ticket SGT

Figure 4. Ticket granting service returns ticket to client.

The message returned from the TGS to the client has a similar structure to that sent by the AS to the client, it consist of ID and realm of the client together with a re-usable ticket SGT facilitating a single-sign-on procedure verifying the authentication of the client; together with an encryption of the session information relevant to the client and SS communication using the client-TGS session key previously issued by the AS. This session information includes a client-SS session key, the realm, ID, and IP address of the client, along with duration stipulations.

Step Five: the client stores the authentication ticket for presentation to the SS, and creates a new authenticator, which is encrypted with the client-SS session key that was previously retrieved. It contains the clients' realm and ID, with a new time-stamp, an optional Sequence Number to detect replays, and an optional session sub-key for the subsequent client SS communications, Fig. 5.

A→SS: $K_{SS}(K_{ASS}+R_A+A+IP_A+Tf)+K_{ATGS}(A+R_A+T_{S1})+O$

Here: *O refers to optional fields that A can request*

Figure 5. Client requests service from the server service.

The optional field indicates the client can request the SS verify its identity, showing that it really is the SS that the client intended to communicate with, thus insisting on mutual authentication.

It should be noted that although A presents an authenticated realm SGT to the SS in question, the TGS

does not perform any match of ID_A to ID_{SS}. Meaning that the SS can later return a message to A indicating that despite authentication A is not authorised to used that SS. The usual means of achieving this is by Access Control Lists, but that is topic which does not concern us here [7].

Step Six: the final authentication message returned to the client from the SS is of the form, Fig. 6.

SS → A: $K_{ASS}(T_{S2}+S_K+SQ_{N0})$

Here: T_{S2} *is a new timestamp*
SQ_{NO} *is a sequence number*
S_K *is a flag requesting a new sub-key*

Figure 6. Server service verifies its ID, mutual authentication.

The SS decrypts the SGT using its key, and then maps the ID and IP address of the user's workstation and the ID of the server SS. Confirmation of SS ID takes the form of, where applicable, the sub-key (if S_K is absent the previous client server service session key K_{ASS} is used) and sequence numbers contained in the message, together with a new timestamp T_{S2} (it is not possible for an attacker to re-construct this massage without prior knowledge of the session key $K_{ASS,}$ and so T_{S2} can safely be returned without modification), encrypted with client-SS session key. The inclusion of successively incremented sequence numbers, upon iteration of message exchanges is intended to prevent replay attacks within the session. Only the client and the correct SS have the session key, in this way the SS has authenticated itself to the client.

Without wishing to digress too far, a nonce is generated and used only once. For security reasons, to prevent guessing its value, it should be created by a random number generator. An SQn is incremented upon each iteration; the original may also be a random number.

IX. IPSEC

Communications at the IP stack layer can be secured by a protocol suite known as IPsec [8]; this facilitates encrypted host to host transfers across an insecure channel by modifying each IP packet during a communication session [9] and [10]. Without going into the detail of Transport Mode and Tunnel Mode, the latter is mostly used for Virtual Private Network connections. IPsec can protect any and all application traffic over an IP network. In Transport Mode it also includes protocols for establishing mutual authentication between entities at the start of a session (entity/host authentication does not concern us), and the negotiation of cryptographic keys for use during that session. In short, Transport Mode can secure communication across an insecure network.

The IPsec architecture uses the concept of security associations, SA's, to form the basis for implementing security functions in IPsec. An SA is a grouping of algorithms (such as keys, signatures, polices) used to encrypt and authenticate a particular unidirectional flow of messages. An SA is defined on a particular host for communications with a unique remote host, and the policies therein are applied to selected packets intended for that host before transmission. Thus, for meaningful bi-directional traffic, these flows are secured by a pair of SA's, one located at each host.

X. KERBEROS OVER IPSEC

It is evident that the encryption of message exchange sequences between client and server to achieve authentication in the Kerberos protocol is intended to preserve confidentiality and integrity between the concerned parties. It is equally clear that where IPsec is already implemented between those parties it is more than capable of providing suitable encryption services and associated guarantees of confidentiality and integrity.

The purpose of encrypting the tickets is to hide information from those parties through which it must travel before reaching its destination, and where the act of decryption verifies identity by virtue of possessing the necessary key.

where a communication channel can implement IPsec for network traffic, the initial message of a client requesting services of a Server Service, SS, is as before to the Authentication Server, AS, Fig. 7.

A → AS: $A+R_A+TGS+N_1+Tf$

Figure 7. Initial message from client to authentication server.

New Step two: the AS reply now differs from that previously indicated, but the TGT still needs to be encrypted with $K_{TGS.}$ Nonces are still necessary because IPSec's sliding window does not provide sufficient protection against replay attacks.

AS→A: $A+R_A+K_{TGS}(R_A+A+IP_A+Tf)+R_{TGS}+TGS+N_1+Tf$

The TGT is still encrypted with K_{TGS} (A to TGS session key) but now as client to AS communications are protected by IPsec encryption there is no need for extra encryption of the session information between them.

New Step Three: the client now sends a request to the TGS for a TGT

A → TGS: $SS+Tf+N_2+ K_{TGS}(R_A+A+IP_A+Tf)+A+R_A+T_{S1}$

It is no longer necessary to encrypt the authenticator with the K_{ATGS} (A to TGS session key), therefore this key was not relayed to the client in the previous message. Since A cannot process the TGT it passes this on intact

New Step Four: We now examine the TGS reply to A.

TGS→A: $A+R_A+K_{SS}(K_{ASS}+R_A+A+IP_A+Tf)+$

$K_{ASS}+Tf+N_2+R_{SS}+SS$

As the session information between them does not need to be encrypted, the K_{ATGS} was omitted from the

previous TGT message. The SGT passing through A still needs to be encrypted, A cannot process it.

New Step Five: the client now conveys the SGT (Service Granting Ticket) it has received from the TGS to the SS, together with authentication information.

$$A \rightarrow SS: K_{SS}(K_{ASS}+R_A+A+IP_A+Tf)+A+R_A+T_{S1}+O$$

New Step Six: Where mutual authentication of the SS is requested by the client, the final message back to the client is no different to the pre-IPsec version, and is of the form

$$SS \rightarrow A: K_{ASS}(T_{S2}+S_K+SQ_{N0})$$

The overall message exchange scheme remains Kerberos like, Fig. 8.

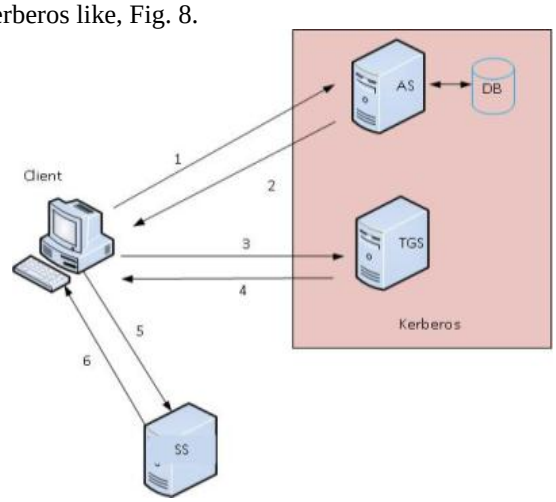

Figure 8. Kerberos message exchange sequence.

XI. Implementation

Not directly stated though implied is the feasibility assessment of running Kerberos in its present form of over an IPsec connection. Although this infers double encryptions, this would be possible to implement in a similar way to Kerberos over TLS [11] and [12] – all that would be needed is a similar hook, to that within Kerberos for TLS, such as STARTIPS, instead of STARTTLS. With minor modifications to the IP packet header and usage, it would be possible to initialise cooperation between Kerberos and IPsec in such a minimalistic fashion. A flag in the IPsec header could indicate that the payload data contains Kerberos information.

However, it is not such a great advantage for IPsec to be aware of its payload. It is more important for Kerberos to know that it can rely on IPSecs' encryptions to use the modified structure proposed above.

Kerberos would need to detect if IPsec could be enacted so that the newer protocol can be used. One possibility might be to inspect IP packets for the presence of the IPsec header; another could be detecting the presence of an SA already existing for the desired host. Otherwise, upon initiating authentication requests Kerberos would need to initialise the IPsec SA set up negotiations. This would be solved programmatically; the Kerberos software will need to interact with the IPsec software on its host to trigger the first IKE phases of SA setup necessary for IPsec communication with the intended partner host.

For backwards compatibility it might be necessary to communicate with legacy Kerberos systems, the possibility of falling back to Kerberos V5 would be required. In which case there is no need to insist on the use of IPsec unless it is anyway useful to know that the IP packet pay load contains Kerberos data. Further, those legacy hosts possibly may not be IPsec enabled, but should anyway be allowed to bargain for authentication.

XII. Global Application

It is anticipated that reducing the volume of encryptions processed together with a simplified Ticket structure would involve benefits for lowering the bandwidth and processor cycles required on large scale enterprise installations, particularly when one realm is joined to another by a router to router connection, where these reductions could be significant for such bottlenecks.

Further, there is the general rule of thumb that less complex systems should be easier to maintain.

XIII. Phase II

It has been shown above that in principle the idea of streamlining Kerberos is valid when IPsec can be relied upon to provide encryption services. The required procedures for Kerberos to instigate IPsec for its ticket exchanges were also indicated.

The next phase of the investigation considers removing in entirety encryptions from the Authentication system, and the associated further modifications to the structure of messages exchanged. Such open ticket exchanges without encryption boundaries places far more emphasis on the installed program code to perform the necessary information filtering and routing. This is nothing new, for it has been part of Kerberos from the beginning.

To this end the reliability of IPsec encryption must be considered, if only briefly. In particular during SA (security association) setup the protocol decrypts packets in dynamic memory before re-transmission. If we take this as being secure, similar methods can be employed with the revised Kerberos, entirely eliminating the need for encryptions within the Authentication Protocol. Relying on updated program code to deconstruct any elements purpose of whose temporary existence on a host is merely intended for re-transmitted.

Revised message structures:-

Step 1: proceeds exactly as before

$$A \rightarrow AS: A+R_A+TGS+N_1+Tf$$

Step 2: AS reply to A

$$AS \rightarrow A: R_A+A+IP_A+R_{TGS}+TGS+N_1+Tf$$

The AS does not return any duplicated information, but still confirms to A that this is who it is answering.

Step 3: A requests an SGT from the TGS

$$A \rightarrow TGS: SS+Tf+N_2+R_A+A+IP_A+T_{S1}$$

The major change here is in the reduction of session information transmitted.

Step 4: TGS returns a SGT to A

$$TGS \rightarrow A: A+R_A+IP_A+Tf+N_2+R_{SS}+SS$$

It is no longer necessary to transmit encryption keys inside of encrypted tickets.

Step 5: A forwards the SGT to the SS

$$A \rightarrow SS: R_A+A+IP_A+Tf+T_{S1}+O$$

The SS can now accept A as being authenticated.

Step 6: the SS verifies its identity to A

$$SS \rightarrow A: T_{S2}+S_K+SQ_{N0}+O$$

Here the major change is that for mutual authentication the SS return to A the only information it has received origination from that source, that is O.

The revised procedure indicates that it is no longer necessary for any of the hosts involved to store encryption keys, as these are no longer needed for encryption/decryption, and their transmission is also therefore superfluous.

XIV. Related Work

These proposals should not be confused with Kerberised Internet Negotiation of Keys, KINK [13], which utilises Kerberos as a Trusted Third Party to manage peer host authentication for the construction of Security Associations, SAs, during the initialisation phases of IPsec communication setup.

Indeed, the protocol proposed would negate KINK, as this relies on the legacy encryption protocol, which this new protocol does without.

XV. Conclusion

It is possibly fair to say that had IPsec been known at the time Kerberos was designed that these factors may well have been taken into consideration.

XVI. Suggestions for Further Research

From the above a running prototype of the proposed modifications to the Kerberos structure in network environment, in both first and second phases, would be an obvious next stage for anyone with the resources to set this up. This should be done along with security resilience testing and performance testing.

Investigations should also be made into implementing similar measures for other aspects of IPsec such as Tunnel Mode and Encapsulating Security Payload, ESP [14].

Acnowledgment

The author wishes to thank Dr. Robert Askwith of LJM University for his guidance during the preparation of the dissertation that led to this paper.

Referrences

[1] P. C. Cheng, "An architecture for the internet key exchange protocol," *IBM Systems Journal*, vol. 40, no. 3, pp. 721-746, 2001.

[2] W. Stallings, *Network Security Essentials*, 4th Edition, Pearson Publishing, March 2010.

[3] J. Steiner, C. Neuman, and J. Schiller, "Kerberos: An authentication service for open network systems," in *Proc. Usenix Conference*, February 1988, pp. 191-202.

[4] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," *ACM Transactions*, vol. 8, no. 1, pp. 18-36, February 1990.

[5] B. C. Neuman and T. Ts'O, "Kerberos an authentication service for computer networks," *IEEE Communication Magazine*, vol. 32, no. 9, pp. 33-38, September 1994.

[6] B. C. Neuman, T. Yu, S. Hartman, and K. Raeburn. (July 2005). The Kerberos network authentication service (V5), RFC 4120. [Online]. Available: http://tools.ietf.org/html/rfc4120

[7] R. S. Sandhu and P. Samarati, "Access control: Principles and practice," *IEEE Communication Magazine*, vol. 32, no. 9, pp. 40-48, September 1994.

[8] R. Atkinson. (August 1995). Security architecture for the Internet protocol. RFC 1825.(Online). Available: http://tools.ietf.org/html/rfc1825

[9] K. G Patterson, "A cryptographic tour of the IPsec standards," *Information Security Technical Report*, December 2005.

[10] C. A. Shue, M. Gupta, and S. A. Myers, "IPsec: Performance analysis and enhancements," presented at IEEE International Conference on Communications, Glasgow, Scotland, 24-28 June, 2007.

[11] S. Josefsson. (May 2011). Using Kerberos version 5 over the transport layer security (TLS) protocol. RFC 6251. [Online]. Available: http://tools.ietf.org/html/rfc6251

[12] K. Bhargavan, C. Fournet, R. Corin, and E. Zalinescu, "Cryptographically verified implementations for TLS," in *Proc. 15th ACM Conference on Computer and Communications Security*, October 2008, pp. 459-468.

[13] S. Sakane, K. Kamada, M. Thomas, and J. Vilhuber. Kerberised. (March 2006). Internet negotiation of keys (KINK). RFC 4430. (Online). Available: http://tools.ietf.org/html/rfc4430

[14] S. Kent and R. Atkinson. IP encapsulating security payload (ESP). RFC 2406. November 1998. Available: http://tools.ietf.org/html/rfc2406

Dean Rogers was educated at Porth Grammar Technical School, South Wales, and Oxford Kellogg College, and has just completed (June 2013) an MSc in Computing, specialising in Computer Security, at LJM University, UK.He has worked as a network administrator, and a DBA, for some household names, and is currently considering his future.