IJWA

Design of an Adaptive Security Manager for Distributed Systems

Veli Hakkoymaz, İsmail Alan
Fatih University
Computer Engineering Department
Buyukcekmece, Istanbul, Turkey
{hakkoymaz, ialan}@ fatih.edu.tr

ABSTRACT: *This paper presents an adaptive policy management scheme for the security threats in distributed systems. First, we investigate the security aspects of computer networks architecture and observe that security services are provided at application, transport and network layers. Network layer security presents some important opportunities for offering controls over security threats to distributed information systems connected to the global Internet. Important details of Internet Protocol (IP) security services are studied and enhancements are proposed over handling both protected and unprotected packet traffics for a dynamic environment. Dynamism is in terms of severity of security threats. The contribution of the paper includes a security policy manager (SPM), which is an adaptive security control mechanism that would automatically configure security levels for a host in response to the frequency of security attacks.*

Keywords: Adaptive security, Computer networks, IPSec, Security model, Security policy manager

Received: 11 September 2009, Revised 19 November 2009, Accepted 1 December 2009

1. Introduction

Most of today's information systems are growing and becoming large-scale distributed information systems that operate in unbounded network environments such as Internet. In these systems, there is neither a centralized control nor a unified policy to differentiate trustable sites from non-trustable sites. This makes it difficult to provide security for these systems [1, 10, 16, 21, 22]. Security threats such as *interception* (i.e., illegal access to data or services), *interruption* (i.e., denial-of-services), *modification* (i.e., change of data or services from their original specification) or *fabrication* (i.e., appending additional data or activity) always exist in a distributed information system. Many different conditions can influence these security threats to materialize [14, 16].

Origin of threats against the systems can be deliberate or accidental. For example, the faulty protection mechanism such as security flaws in design, wrong implementations, and poor system integration and organization invites most of the attacks to take place to such a system. For that reason, an organization's system connected to an unbounded network would be vulnerable to attack from many potentially harmful sources on a public Internet. After these threats are materialized, the worst consequences are discontinuity of system services, and/or loss of system functionality and data. These in turn can have dramatic effects on many aspects of daily life, including healthcare, education, work, entertainment and environment. The fear of security breaches can lead organizations to decide to adopt a radical solution in that "secure" protected private networks are physically separated from the public Internet [8, 13]. However, this is not an ideal solution since the concept of a global Internet would be compromised. Instead, a desired solution must enable the electronic commerce and provide the open access to all kinds of public information and services. To achieve that, some kind of logical protection must be provided for information units during their transfer over the global network. In other words, an acceptable solution enables individual users or applications to communicate with each other over a global network in a secure manner, protecting the confidentiality, privacy, integrity and confidence of its users.

Before proceeding any further, we need to clarify a few concepts. A *policy* is defined as a deliberate plan of action to guide system decisions and have the system to achieve its desired outcomes. Now we can define the *security* as a measure of

confidence in the policy in place for the management of a system. In other words, the security refers to a level of confidence to the system in that (via various actions) its policy directs the system towards a state within which a desired outcome is achieved. On the other hand, *protection* (i.e., sometimes used in the same context as security) refers to a mechanism through which the access of programs and users to a set of resources are controlled.

Security requirements of a system dictate a security policy to be employed for a system in which for its entities what actions to take are clearly defined. In this paper, we are looking into approaches for providing adaptive security policies against security threats. To do that, we first investigate what is available by discussing security components of the network architecture with respect to open system interconnection reference model [7]. Then, we look into management of security policy issues and clarify interactions between various security components. Finally, we address the configuration of an adaptive security policy.

Specifically, in the remainder of the first section, we discuss employing security solutions at different layers. Related work is given in section 2. Section 3 presents important details of Internet Protocol security services (IPSec) and how inbound and outbound packets get processed. Section 4 introduces two adaptive security models for Security Policy Manager. Section 5 gives the conclusion and future work.

1.1 Security at What Level

Security in computer networks can be provided at application, transport or network layers of Open Systems Interconnection (OSI) reference model [7, 8].

At the application layer, an application that needs to transmit invaluable data over an insecure network can use internal mechanisms to achieve data integrity and confidentiality. In this case, the application programmer must implement and embed into his/her application all necessary security tools for transmitting data in a secure manner (i.e., secure channel). This extra effort involves expertise and adoption of tools in authentication, encryption and key exchange protocols on behalf of the programmer. Other programmers working on other security-critical applications would have to do the same thing for their applications all over again. However, usually not all programmers are experts in implementing security protocols. Therefore, they can introduce mistakes to their applications when implementing security protocols [3, 9, 20].

At the transport layer, socket-based security implementations are common solutions to protect users' invaluable data from illegal accesses. Usually, applications are provided with enhanced sockets that encrypt data flowing through them. The main advantage of transport layer security over application layer security mechanism is that the applications are relieved from encrypting user data that need to be protected. For the application, the communication is secured using an enhanced socket and its interface [3, 20]. The requirement of transport layer security mechanisms is that an application that needs data security must be changed to incorporate enhanced socket interface.

At the network layer, Internet Protocol (IP) security provides the data confidentiality and integrity of its IP packets, regardless of how the application used the socket interface [2, 3, 5, 8, 9]. This means that any application can benefit from the underlying secure IP network, as long as it uses IP to send data. The main advantage is that applications are unaware of usage of IP security protocols. Thus, security services are provided to applications and users in a transparent fashion. Those real time and multicast applications that are based on connectionless user datagram protocol (UDP) can also benefit from IP security [7, 8]. Characteristic features that a security protocol provides at this layer can be listed as: (i) security is provided at the network layer and (ii) it is transparent to end users and applications. The disadvantage is that for high speed networking some performance loss can be observed due to packet processing overhead at IP layer.

2. Related Work

In recent years, the growth of information and communication networks and the need for effective protection of the data transmitted on those systems have made it necessary to develop a large number of security protocols on information transfer [2, 5, 6, 10, 11, 17, 18]. Speed and services of routers must be improved with new developments in protocols. Concept of "dynamically upgradeable router" is developed in [5]. Specifically, the design and implementation of modular and extended services router architecture are described for NetBSD OS kernel in software. *Routers* are the computer networking devices that buffers and forwards data packets toward their destinations through a packet scheduler. Routers perform packet processing operations based on the packet header information such as protocol type, destination address, source address and so on [5, 16, 18, 19]. The architecture in [5] allows dynamically addable plug-in at run time. Plug-ins are code modules which implement a specific function of the router. For example, security plug-in is used for providing security functions. Modular architecture also includes a packet classification algorithm. In addition to destination address based forwarding, state-of-the-art routers

must provide new services such as integrated services, security algorithms, enhanced routing functionalities and selective packet droppings. The standard functions provided in this architecture are IP forwarding and routing, a packet scheduler, a packet classifier, security mechanisms, a firewall module and congestion control services. Another feature of this design is to classify packets into groups and apply different policies to different groups. Difference of this with our work is that different application flows can be associated with different customized plugins in the former while the latter provides design of an adaptive security policy management for all flows regardless of their types.

Another related research area is on preparation for a situation in which security threats get materialized. *Survivability* is defined as the ability of a system to continue to provide the adequate performance of its critical services and functions even after unforeseeable attacks have taken place and succeeded [1, 4]. Providing survivability for a system can be difficult. Assuming that the security threat always exists for a system, security requirements must be defined for the system. These requirements must be classified into two categories, namely *essential services* and *non-essential services*. Essential services must be maintained even during successful attacks, non-essential services are to be recovered after intrusions have been dealt with. Main issues in survivability are: (i) *fault tolerance*, and (ii) s*ecure operating environment***.** Fault tolerance is the property that enables a system to continue operating properly in the event of the failure of some of its components [4]. Design and implementation faults or attacks (i.e., viruses, malicious software, etc.) cause a system to malfunction frequently [14]. To cope with system faults, some basic principles can be employed: (i) use of reliable components (ii) careful design, (iii) providing alternative solutions and (iv) employing redundancy at the occurrence of system faults. Such principles are building blocks for fault-tolerant systems [4]. This can be achieved by developing systems that are autonomously reconfigurable, modular and adaptive.

These changes must be taken into account for increased reliability in that as the security threats potential increases, the system must increase its protection level and when threat potential decreases, the system must decrease its protection level accordingly [12, 17].

3. IP Level Security

For individual users and organizations, in order to provide a logical protection for information units during their transfer over a global network (i.e., with secure communications) Internet protocol (IP) employs some security features, called IPsec, as a service to all traffic using it [2, 3, 6]. We can view IP as a common vehicle for various higher layer components in the network. IPsec services are provided to protect (i) a path between a pair of end systems or hosts, (ii) a path between a pair of intermediate systems and (iii) a path between a host and intermediate system.

Intermediate system can be defined as a system with a packet forwarding functionality (i.e., a router). IPsec has been developed for increasing network layer confidentiality and protection. Protection is of the form of data origin authentication, data integrity, replay detection, data confidentiality and access control [8]. These protections are provided by means of following components in IPsec:

1. two security protocols, namely *authentication header*(AH) and *encapsulating security payload*(ESP),
2. *security associations* (SA) on each IP path,
3. Encryption and authentication algorithms. Briefly, encryption protects the confidentiality of stored data and communication packets by making them useless without a particular decryption key. Authentication is for proving the identity of the sender supplied in the packet that has been received. *Cryptography*, discussion of which is beyond the scope of this paper, offers fundamental means for these services.

3.1 Packet Processing

IPsec is based on the concept of *datagram encapsulation* which means carrying IP packets across the network as cryptographically protected information units. This makes the encryption transparent to any intermediate nodes that must process packet headers for routing.

In a secure network, there are certain operations on each network node for making sure that arriving packets and leaving packets are handled in such a way that packet protections are guaranteed end-to-end. Although these features are present at IPsec, policies governing the handling of inbound and outbound traffic to and from a host running the protocol are somewhat restrictive and are subject to refinement [2]. To illustrate, in packet filtering operation, a packet is checked first if it is protected or not. If it is not protected, the proper policy to use is ambiguous in handling such a packet? In addition, a packet under the scope of some SA may not be decapsulated because of incorrect keys. The proper policy to handle such a packet is

again ambiguous. It seems that IPsec has left to the protocol implementers as to what answers are appropriate to these questions. Some can argue to *discard* it, others to *bypass* the node to forward it to its final destination. We describe two different adaptive policies and give mechanisms for their implementations in section 4.

Furthermore, in IPsec transport mode, an authentication header (AH) consisting of security parameter index (SPI), sequence number (Seq No) and integrity check value (ICV) is appended to original IP datagram. This mode is intended for end-to-end protection implemented by source and destination hosts of original IP datagram. ICV is calculated by applying a secure hash function on IP datagram with following change: its mutable fields (i.e., hop-to-hop extensions, etc.) and authentication data field are cleared (i.e., set to zero) before applying the hash function. Resulting value is recorded as ICV of IP datagram.

In tunnel mode, in addition to authentication header an encapsulating IP header which may have different source and destination addresses from those in original IP header are appended to original IP datagram. These new source and destination addresses form *a secure path* which indicates a fraction of end-to-end path between source and destination hosts of original IP header. **A secure path** between source and destination is defined with a structure called security association (SA). In other words, source and destination nodes (either end systems or intermediate systems) are connected through with a SA, providing a secure path between these two nodes. Important details of SA's are explained in next subsection.

3.2 Secure Connection

IPsec connection is described with a *Security Association (SA)*. An SA is a set of information consisting of security parameters that two network security endpoints agree upon in order to establish a secure communication. Major information contained in SA includes cryptographic keys, initialization vectors, digital certificates and so on for encryption, authentication and decryption algorithms in use for a secure channel at each endpoint. Each IP packet has a SA id. This id consists of packet destination address, protocol id and security parameters. It is used for accessing related entry in the SA database at each host. In Fig 1, a secure path in a network is shown with a box representing a security endpoint and a directed edge representing an IPsec connection.

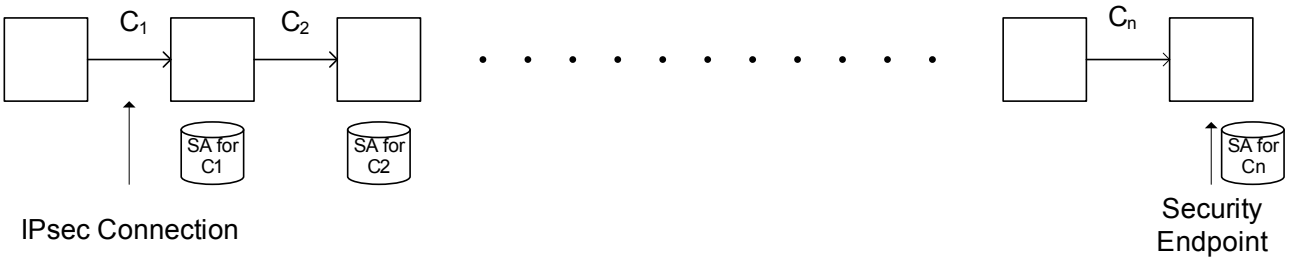

Figure 1. A secure network where a box representing a security endpoint and a directed line representing an IPsec connection

Basic packet processing involves checking first whether or not the packet is under the scope of some SA. A SA contains various information items for performing network layer security check; how and when a new SA gets created and so on. A new SA gets created when a communication is about to be established with a new host. It is important to create a new SA for a secure path between two hosts. The problem of policy management for SAs is different than the problem of filtering individual packets in that: (i) SAs lifetime is long: an SA cannot be dropped immediately after a communication is over because new communication may occur in near future, (ii) policy controls on SA creation need more resources than creating a packet, (iii) when an SA is established between two hosts, packet filtering operation decreases.

3.3 Inbound and Outbound Packets Processing

Policies related to handling of packet traffic entering and leaving a host which is running the IPsec protocol are described in this subsection. Network packets are an important source of information used by several commercial intrusion detection products. They are also seen as an efficient means for collecting information about events that occur on network infrastructure. For example, most accesses to task-critical computers take place over a network. Thus, capturing the packet before they enter a host can be considered as an efficient way to monitor the system. First, we give a flowchart that depicts how inbound and outbound packets are processed in Fig. 2.

For an inbound packet arriving at a host from the network, the host goes through following steps [2]:

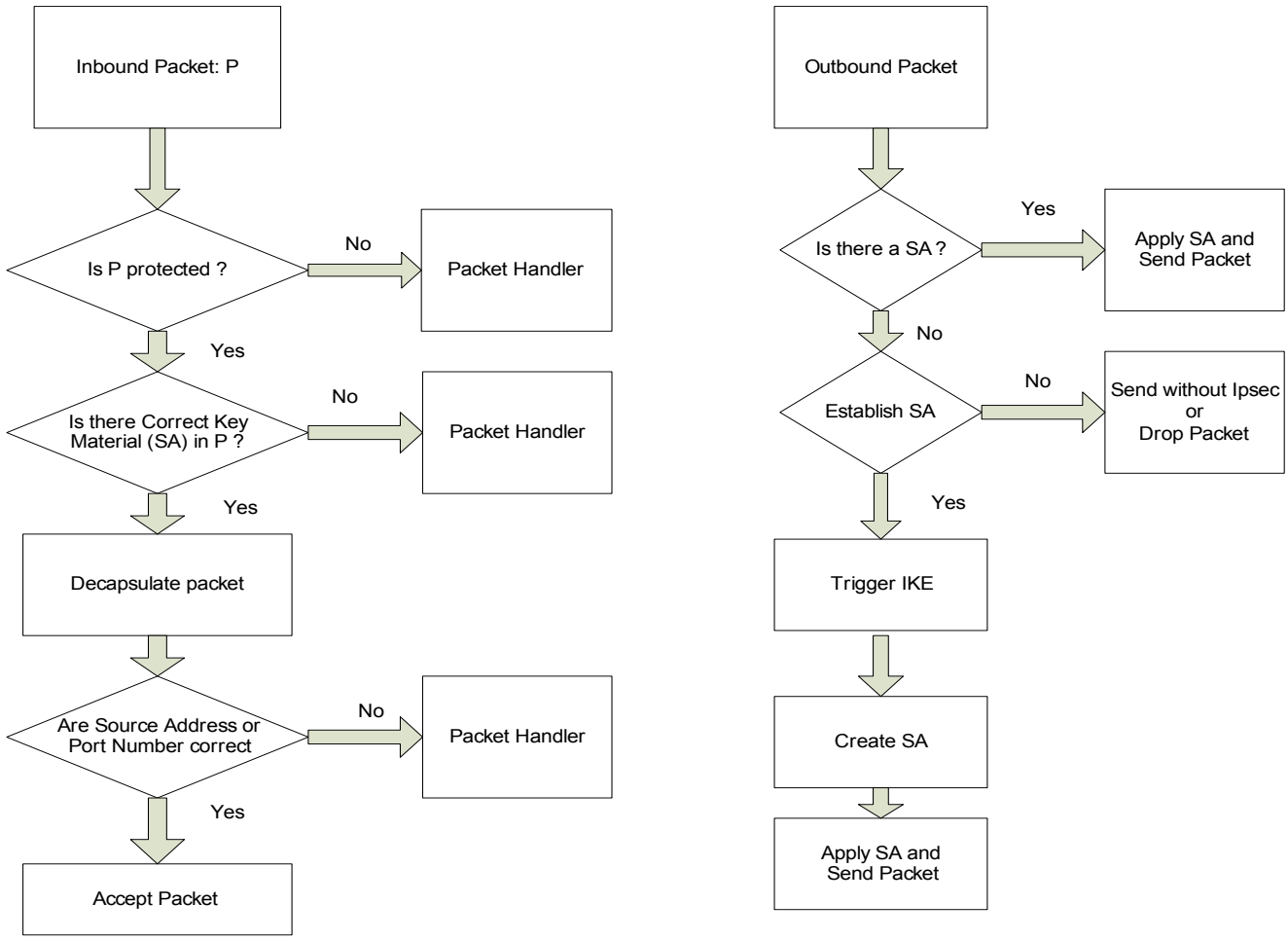

Figure 2. Flowcharts for processing Inbound and Outbound Packets in IPsec

Step 1(*packet filtering*): check if the packet is protected or not. If it is protected, go to Step 2. If it is not protected, should it be accepted? This decision is left to network firewall.

Step 2(*protected packet handling*): for protected packets, either one of the following two cases are true.

Case1: there is a correct key in the SA to decapsulate the packet. If so, decapsulate it. This does not mean that packet is acceptable. It may contain invalid source address or unauthorized destination port number. If that is the case, what should be done with it?

Case2: there is no correct key in the SA, thus unable to decapsulate the packet. If this is the case, what should be done with this packet?

For an outbound packet, similar decisions are made per packet basis, as described below:

Step 1(*check SA*): check if there is an SA applicable to this packet? If there is only one SA applicable, apply it to the packet and send. If multiple applicable SAs exist, which one gets selected?

Step 2(*handling packet with no SA*): If no SA is applicable to the packet, what are the options available to handle such a packet? Some options are *forward* to some network interface, *drop, or queue* until an SA becomes available.

These operations may slow down network data rates when applied per packet basis. In order to avoid such a slow-down, special care must be taken to cause the degradation of network data rates into intolerably low level by setting up threshold parameters t_1 and t_2 properly in the proposed security models, to be described next.

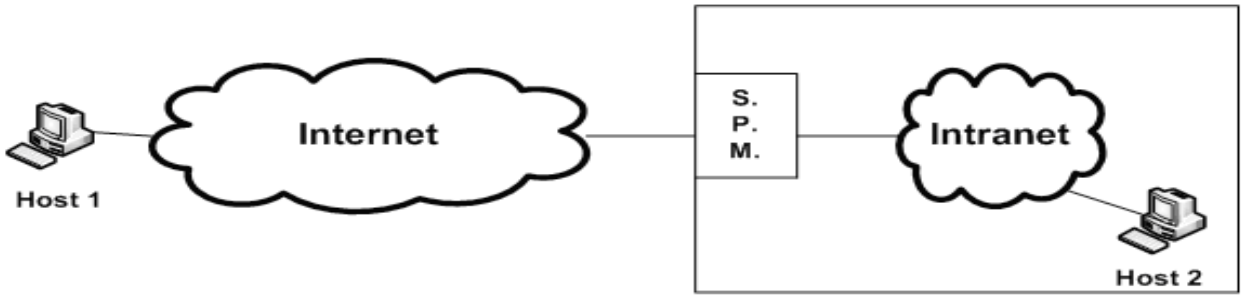

Figure 3. SPM environment for both security model 1 and 2. Host 1 is a source, Host 2 is a destination

4. Security Policy Manager

We design an adaptive security policy manager (SPM) in that as the frequency of attacks increase for a host in its responsibility area, security barrier (i.e., protection) for that host will be made higher; as more time passes without any attacks to the host, the security barrier for that host will be lowered. As shown in Fig. 3, SPM has a responsibility area consisting of N hosts. Let G denote the set of these hosts in the responsibility area of SPM (i.e., $|G| = N$, host h_2 is in G, and host h_1 is not in G).

In case any unprotected packet arrives with a destination not in G, that packet will be delivered to packet scheduler for routing to its destination, thereby bypassing SPM. Otherwise, if an unprotected packet arrives at SPM for a host in G, there could be further processing at SPM with related to this packet based on a feedback from destination host in G. For the feedback mechanism, the destination hosts run an *expert system* as a knowledge-based *intrusion detection system* (IDS), such as the ones in [23, 24]. Expert system contains a set of rules that describes various security attacks. In addition, each of the security related events is translated to a fact with a significance level. Afterwards, the inference engine draws a conclusion based on these rules and facts. Finally, the SPM is informed about this conclusion through the feedback mechanism.

SPM models use a network-based intrusion detection approach by focusing on the attacks to network [15]. Network attacks can be detected by specific tools which have been developed to examine network packets in real-time, searching for specific network attacks. Specifically, some security attacks can be detected by parsing the payload of the network packets and looking for suspicious commands.

4.1 Security Model 1: A Statefull SPM

In this particular configuration, in case any unprotected packet arrives with a destination not in G, that packet will be delivered to packet scheduler for routing to its destination bypassing SPM. Otherwise, if an unprotected packet arrives at SPM for a host in G, there could be further processing at SPM with related to this packet based on feedback from destination host in G.

In this model, a host provides a feedback to SPM when a faulty packet arrives and SPM will make a note of the source of the faulty packet in the list maintained for this particular host. We assume that each host has a means –such as an expert system based Intrusion Detection System (IDS)-- to determine a faulty packet from the rest when they are not protected [15, 23, 24]. For each such packet, SPM writes <source h#, dest h#, seq no> into its security log records, together with a time-to-live(TTL) parameter set to $2\Gamma + dt + \varepsilon$, where 2Γ represents the message delays between the SPM and the host, *dt* the time to detect an intrusion (IDS' computation time) and ε some additional delays for the host to generate a feedback. If the IDS running on the host (i.e., destination host specified as dest h#, say h_j) generates a positive alarm on the delivered packet, a feedback consisting of <dest h# h_j, seq no> is sent to SPM while the entry in security log records is still alive. Upon arrival of the feedback, bad-packet-count for this particular host is incremented in SPM. Regardless of whether or not a feedback arrives at SPM within its TTL period, a background security log filter process (SLFP) clears that entry from the security log records, sometime after that record's TTL period expires. In addition, a feedback causes SPM to mark the source h# in security log records as the suspect source.

Figures 4 and 5 shows how an unprotected packet gets processed when there exist no faults and there exists faults, respectively.

In other words, SPM maintains two pieces of information for each host h_i in G:

1. bad-packet-count to keep track of the severity of the attacks to host h_i,
2. a list of host addresses of suspected sources (i.e., we call it suspect hosts).

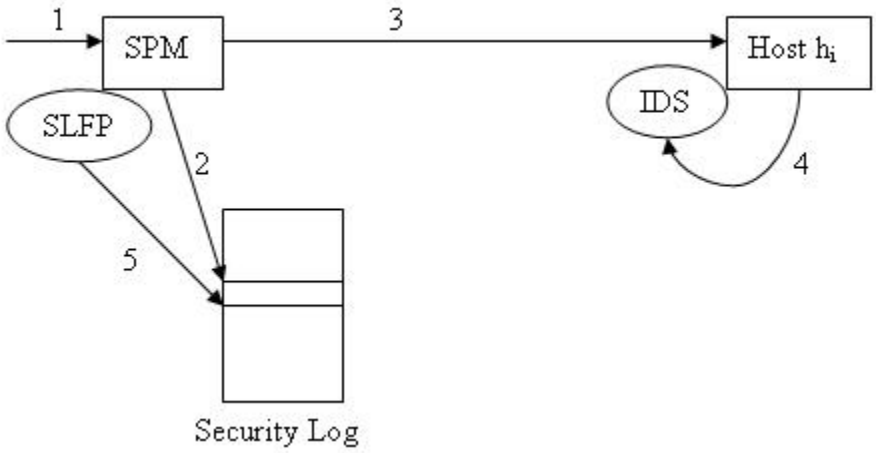

Figure 4. Unprotected packet processing without a fault. 1- unprotected packet arrives at SPM. 2- SPM inserts a note into the security log records, 3- delivers it to the intended host, 4- IDS runs on the packet, finds no fault, 5- Security Log Filter Process (SLFP) removes the corresponding entry from the security log.

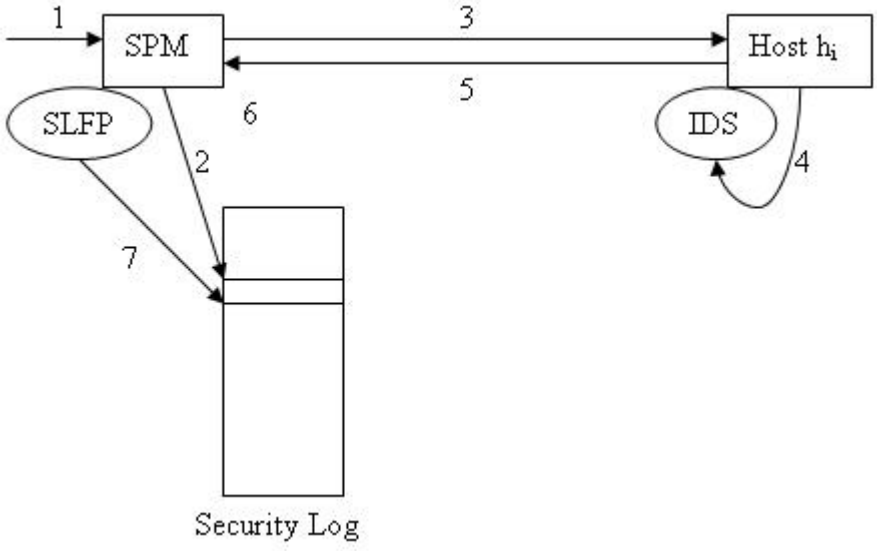

Figure 5. Unprotected packet processing with a fault. 1- unprotected packet arrives at SPM. 2- SPM inserts a note into the security log records, 3- delivers it to the intended host, 4- IDS runs on the packet, finds a fault, 5- a feedback is generated and sent to SPM, 6- SPM marks the source ID as suspect, 7- Security Log Filter Process (SLFP) removes the corresponding entry from the security log.

Consider a host h_i in responsibility area of SPM. *Suspect hosts* for h_i (i.e., denoted by SH_i) would be classified as *own suspects* (OS_i) and *imported suspects* (IS_i). OS_i are those source hosts that send bad packets to destination h_i. Similarly, IS_i are suspect hosts that have sent bad packets to those hosts h_j in G other than host h_i (i.e., h_j in G, $1 \leq j \leq N$ and $i \neq j$). Note that $IS_i = \bigcup OS_j$ where $1 \leq j \leq N$ and $i \neq j$. Suspect hosts list SH_i grows as the new attacks take place for host h_i. It would include source addresses of those bad packets addressed to h_i. Although potentially any host on the Internet can send bad packets to a host (thus, their source addresses would be added to the list), not all of them will appear in the suspect hosts list. In fact, list size due to own suspects are bounded by threshold t_1 (i.e., $|OS_i| \leq t_1$). In some extreme cases, the list size can grow to a maximum of $N*t_1$ since there can be a maximum of t_1 many number of h_i's own suspects, and $(N-1)*t_1$ many numbers of imported suspects. However, since multiple suspect lists may contain same entries (i.e., SH_i and OS_i are not disjoint sets), we anticipate the size of the list SH_i for each host h_i to be much smaller.

If enough time passes without a security incident (we call this interval *recovery period*), then security barrier will be lowered to reflect this change in the environment. More specifically, as the time passes without any bad packet addressing a particular host h_i (after a

recovery period), first, those safe sources (i.e., hosts not in SH_i) are allowed to communicate with host h_i. As more time passes without a security incident (after a second recovery period) for host h_i, then those sources that are suspects to other hosts (i.e., imported suspects in IS_i) would be allowed to communicate with h_i. Another two recovery periods without a security incident would clear the entire list of suspect sources. Therefore, SH_i becomes empty for host h_i. Fig. 6 shows pseudo code of this security model.

Security Model 1: SPM runs the following steps on arrival of a faulty packet for host h_i. Assume that packet is not protected (no SA, no encryption); t_1, t_2 are thresholds ($t_2 > t_1$)

```
Increment bad_packet_count
Tag source as blocked (add to OS_i)
If bad_packet_count > t_1
  Tag all others' suspect sources as blocked (IS_i=union OS_j's, 1≤j≤N, i≠j)
else if bad_packet_count > t_2
  Tag all sources as blocked (deny all)
  Start a recovery time interval
If no bad packet arrives for a time interval ≥ 4 * recovery period
  Remove tags from own suspects(in OS_i)
  Drop bad_packet_count to 0.
else if no bad packet arrives for a time interval ≥ 2*recovery period
  Remove tags from those hosts in IS_i
  Drop bad_packet_count to t_1/2
else if no bad packet arrives for a time interval ≥ recovery period
  Remove tags from those safe sources (hosts in SH_i are still denied)
  Drop bad_packet_count to t_1+(t_2-t_1)/2
```

Figure 6. Pseudocode of adaptive security model 1

4.2 Observations on Security Model 1

One can make varioous observations after investigating the above pseudocode for security model 1. As bad packets arrive, suspect list size will grow; however, number of suspects in the suspect list is bounded by $N*t_1$ for each host. Fig. 7 illustrates algorithm's behavior with respect to bad-packet-count versus list size.

Another observation can be stated with respect to the passage of time without a security incident. As the time passes without bad packets for host h_i, the security barrier will be lowered and system will accept packets from ever growing set of sources gradually. Figure 8 illustrates algorithm's behavior with respect to bad-packet-count versus time passing without security incidents.

4.3 Security Model 2: A Stateless SPM

As a second scenario, through SPM, a host on the Internet may make an SA with a host in the responsibility area G of SPM. For this scenario, we consider another security model that is slightly different from the first one. Its procedural steps are depicted in Fig. 9 and explained below.

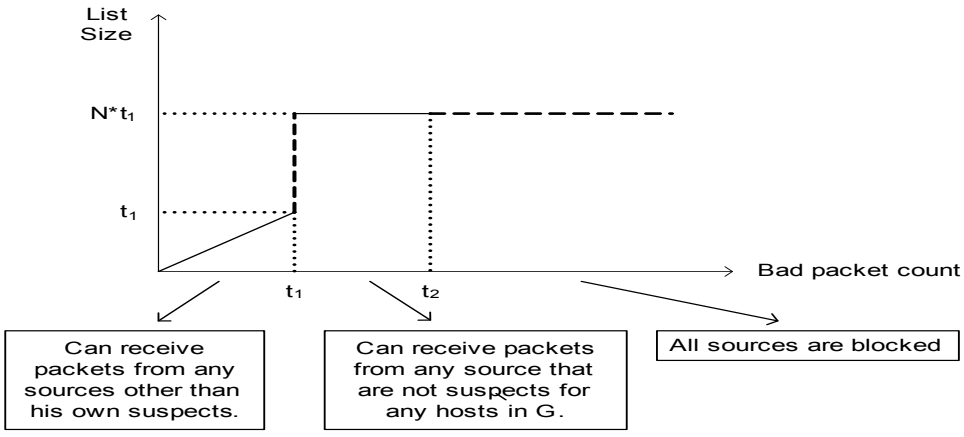

Figure 7. Growth of host Hi's list of suspect sources as security attacks taking place for Hi

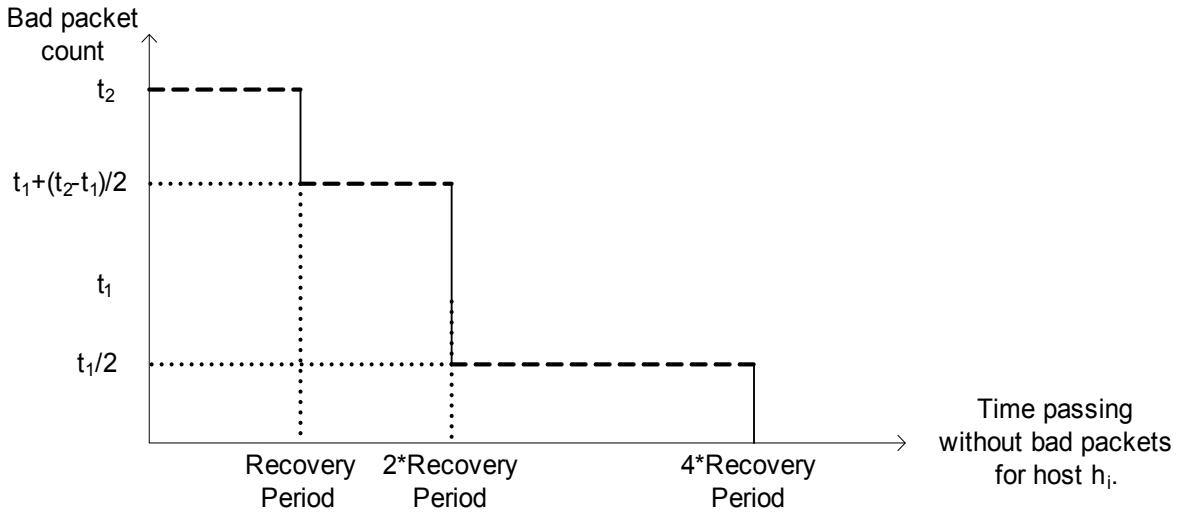

Figure 8. Resetting bad packet count back to 0 as time passes without security attacks

The main difference between security model 1 and security model 2 is that the former deals with the unprotected packets while the latter deals with the protected packets (i.e., exchanged under the scope of an SA). For the protected packets, SPM does not need a feedback from the destination host in its responsibility area G for determining if the packet security is breached since enough information is already available in SA to check for it. Furthermore, during the interval between bad-packet-count of 0 and the threshold t_1, SPM will only tag a source for host h_i as blocked, without removing the SA. This is so since the SA creation is costly procedure, and if threats disappear in the near future, the source and destination hosts would communicate again under the scope of this SA.

However, if security threats persist (i.e., number of faulty packets exceeds the threshold t_1), from then on, all sources in the bad packets not only get blocked, but also their SA's with SPM are removed. If threats continue to persist even more (i.e., number of faulty packets exceeds the threshold t_2), those SA's untouched during the initial stage (i.e., time interval of bad-packet-count of 0 and the threshold t_1) will be removed as well. The way for the security model 2 to recover from a security attack is the same as that of security model 1.

Security Model 2: SPM runs the following steps on arrival of a faulty protected packet (under scope of SA) for host h_i. Assume that packet is encapsulated under an SA (SA exists, but, its keys are incorrect); t_1 and t_2 are thresholds, $t_1<t_2$.

```
Increment bad_packet_count
Tag source as blocked (add to OS_i)
If bad_packet_count > t_1
   Drop SA between this source and dest h_i
   Tag others' suspect sources as blocked(IS_i=⋃OS_j, 1≤j≤N,i≠j)
else if bad_packet_count > t_2
   Drop SAs between hosts in OS_i and dest
   Tag all the sources as blocked
   Start a recovery time interval
If no bad packet arrives for a time interval ≥ 4*recovery period
   Remove tags from hosts in OS_i
   Drop bad_packet_count to 0
else if no bad packet arrives for a time interval ≥ 2*recovery period
   Remove tags from hosts in IS_i
   Drop bad_packet_count to t_1/2
else if no bad packet arrives for a time interval ≥ recovery period
   Remove tags from hosts not in SH_i
   Drop bad-packet-count to t_1+(t_2-t_1)/2
```

Figure 9. Pseudocode of adaptive security model 2

4.4 Observations on Security Model 2

Similar to the previous one, the list size for a host h_i grows linear in this model,as faulty packets arrive at the host until the threshold t_1. The behavior of model 2 suggests that after the threshold t_1 the list size becomes $N*t_1$ again and remains the same from then on until the first recovery period. Fig. 10 shows how the security model 2 behaves with respect to the SA removal. As depicted, between bad_packet_count of 0 and t_1 no SA's are removed. Between t_1 and t_2, the number of SA's dropped grows linear with the number of bad-packets arrived (i.e., at t_2, number of SA's dropped is $t_2–t_1$). After t_2, number of SA's dropped gets to a maximum of t_2 as the algorithm has determined that it is under the attack.

4.5 Evaluation of Security Models

In this section, we discuss emprical evaluations of the two security models just described by giving the efficiency measures to be applied to. We evaluate the efficiency in terms of accuracy, performance and completeness. A security model is considered as *inaccurate* when a non-faulty network packet is flagged as containing a security threat. After examining such a network packet, the expert system running on the destination host may erronously arrive at a conclusion that the received packet is anomalous or intrusive and a feedback indicating that consclusion is sent to the SPM even though the network packet has no security threats. *Performance* is measured as a rate at which packets are processed and feedback provided to SPM. When the performance is low, then real-time detection of security attacks become impossible (i.e., too late to respond to such attacks). As for the *completeness*, if the security model is unable to detect when an attack occurs, then we consider the model as incomplete. Through a prototype system, the interaction of the various components of SPM are observed and efficiency with respect to aforementioned measures ougth to be evaluated. We are planning to report the results of these experiments in the near future.

5. Conclusion and Future Work

In this paper, we have investigated the security aspects of computer networks architecture and observed that security services are provided at application, transport and network layers. Network layer security presents some important opportunities for offering controls over security threats to distributed information systems connected to the global Internet. We have looked at the important details of Internet Protocol (IP) security services and proposed enhancements over handling both protected and unprotected packet traffics for a dynamic environment (i.e., in terms of the severity of security threats).

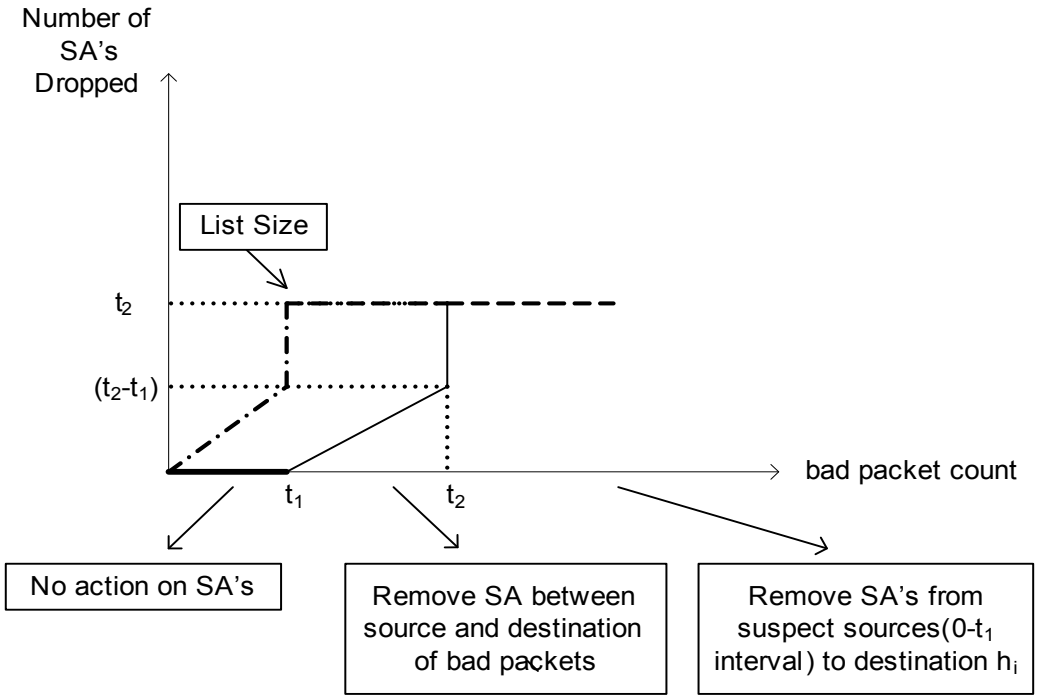

Figure 10. Growth of host Hi's list of suspect sources and removal of SA's as security attacks continues for Hi

Our contribution includes the security policy manager (SPM), an adaptive security control mechanism that would automatically configure security levels in response to the frequency of security attacks. In particular, when security attacks become more frequent, SPM increases the security level by blocking interaction with the sources causing packet traffics suspected of containing security threats. When the time shows that the attacks are no longer occurring, SPM decreases the security level by gradually allowing interaction with the sources of packet traffics, starting from the least-likely towards the most-likely of sending packets containing the security threats.

As for future works, although empirical evaluation seems promising and related research papers [5, 11] suggests a notable performance increases with a fast hash table implementation, how much processing overhead this adaptive policy and its implementation mechanisms bring about must be investigated further.

We are currently conducting simulations to observe the interaction of the various components of SPM, to develop interfaces for each of these components and to monitor the packet processing for both inbound and outbound packets. We hope to find out more about ideal threshold values for each of the algorithms presented.

References

[1] Kyamakya, K., Jobmann, K., Meincke, M. (2000). Security and Survivability of Distributed Systems, IEEE.

[2] Blaze, M., Ioannidis, J., Keromytis, A. (2002). Trust Management for IPsec, *ACM Transactions on Information and System Security* 5 (2) May.

[3] Insolvibile, G. (2002). The IP Security Protocol, *LINUX Journal*, September.

[4] Avizienis, A (1997). Toward Systematic Design of Fault-Tolerant Systems, IEEE.

[5] Descaper, D., Dittia, Z., Parulkar, G., Plattner, B. (2000). Router Plugins: a Software Architecture for Next-Generation Routers, IEEE.

[6] Stallings, W (2006). Cryptography and Network Security, 4th ed., Prentice Hall.

[7] Tanenbaum, A. S (2002). Computer Networks, 4th ed., Prentice Hall.

[8] Molva, R. (1999). Internet Security Architecture, *Computer Networks & ISDN Systems Journal*, 31 (8) April.

[9] Oppliger, R. (1998). Security at Internet Layer, *Computer*, 31 (9) 43-47, September.

[10] Tanenbaum, A.S. (2002). Distributed Systems: Principles and Paradigms, Prentice Hall, 2002

[11] Morris, R. (1999). The click modular router, 17th ACM Sym. on Operating Sys. Principles, Dec.

[12] Alan, Ismail (2007). Trust management policies for distributed computing systems, M.S. thesis, Dept. of Computer Engineering, Fatih University, Istanbul, Turkey, July.

[13] *Walter,* Fumy Ingbert, Haas (1998). *Security techniques for the global* information infrastructure , *Proceedings of the IEEE Globecom*, p. 3141-3136.

[14] De Vivo, M., De Vivo, G. O., Isern, G. (1998). *Internet security attacks at the basic levels, ACM SIGOPS Operating Systems Review, 32 (2)* 4-15, *April.*

[15] Debar, Hervé Dacier, Marc Wespi, Andreas (1999). Towards a taxonomy of intrusion-detection systems, Computer Networks: *The International Journal of Computer and Telecommunications Networking*, 31 (9) 805-822, April.

[16] Bellovin, S.M. (1994). Cheswick, W.R Network firewalls, *IEEE Communications Magazine*, 32 (9) 50–57.

[17] Hakkoymaz, Veli., Alan, Ismail (2007). *An Adaptive Security Policy Design and Management for Distributed Systems,* ISC'Turkey: Information Security and Cryptology Conf., p.238-243, December.

[18] Schwartz, B., Jackson, A. W., Strayer, W. T., Zhou, W., Rockwell, R. D. Partridge, C. (2000). Smart packets: applying active networks to network management, *ACM Trans. on Computer Systems* (TOCS), 18 (1) 67-88, Feb.

[19] Conoboy, B., Fichtner E. IP filter based firewalls how to. Available from: http://www.obfuscation.org/ipf

[20] Oppliger, Rolf (1997). Internet security: firewalls and beyond, Communications of the ACM, 40 (5) 92-102, May.

[21] Yalagandula, Praveen., Dahlin, Mike (2004). A scalable distributed information management system, Proceedings of the 2004 conf. on Applications, technologies, architectures, and protocols for computer communications, Portland, Oregon, USA

[22] Coulouris, George F. Dollimore, Jean Kindberg, T.(2005). Distributed Systems: Concepts and Design, 4th Ed., Addison-Wesley.

[23] Yu, Z., Tsai, J. P., Weigert, T. (2008). An Adaptive Automatically Tuning Intrusion Detection System, *ACM Trans. on Autonomous and Adaptive Systems*, 3 (3) August.

[24] Li, X., Ye, N (2003). Decision tree classifiers for computer intrusion detection, Real-Time System Security, Nova pub., 77-93.

Authors Biographies

Veli Hakkoymaz received BS degree in computer science and engineering from Hacettepe University in Ankara, Turkey, in 1987. He went to the USA for graduate study in 1989, and received MS degree in computer science from University of Pittsburgh, Pittsburgh, PA, in 1992, and PhD degree in computer science and engineering from Case Western Reserve University in Cleveland, OH, in 1997.
He is currently an assistant professor at Computer Engineering Department of Fatih University, Istanbul, Turkey. His research interests are in operating systems, multimedia databases and multimedia presentation systems.

İsmail Alan received his BS and MS degrees in computer engineering from Fatih University, Turkey in 2005 and 2007, respectively. He worked as an instructor in Vocational School of Fatih University for two years before joining Yildiz Technical University, where he is currently an instructor in the Informatics Department. His research interest includes computer networks, computer security and machine learning.

Book Review

Vladimir A Fomichov
Semantics-Oriented Natural Language Processing
Mathematical Models and Algorithms
Series Editor: George Klir
Springer, New York, Heidelberg, London
ISBN 978-0-387-72926-8

Natural Language Processing is a complex issue where researchers deploy varied systems and models. As the basic nature of languages is characterized by heterogeneity, the language for understanding semantics required multidimensional approaches. Through this premier the author has addressed the semantics issue with the help of ten chapters. The author took voluminous efforts to address the semantics required for natural language.

Part 1 – Mathematical Framework for the development of semantic technologies six chapters-Part 2 five chapters – Formal methods and algorithms for design of semantics oriented linguistic processors The book is the outcome of the program on Integral Formal semantics of Natural Language. The authors have extensively discussed the formal models and methods of Natural semantics. The languages and formal models for natural semantics are described with many examples and illustrations. The models are build with a good level of description of problems required to gain understanding. The content is highly structured. The book is not just an initial reader but the contents orient the readers to gain comprehensive understanding of semantics oriented natural language processing.

Daisy Jacobs
University of Zululand
South Africa

Database Modeling & Design: Logical Design
Toby Teorey
Sam Lightstone
Tom Nadeau
4th Edition, Morgan Kaufmann Pub., Inc. (Elsevier), 2006

This book has nine following chapters.
Chapter 1 Data and database management
2- The Entity relationship model
3- UML
4- Requirement analysis and conceptual data modeling
5- Transforming the conceptual data model and SQL
6- Normalization
7- Logical database design
8- Business intelligent
9- case tools for logical database design Appendix – basics of SQL

Data, data base, data modeling are the basic set of concepts required to all levels of reader in information science. The literature is filled with a large collection of them. In the two decades development in fundamental area of database is significant and the readers are looking at comprehensive treatises on the database domain. This book caters such requirement with good support of examples and illustrations by the authors.

Three chapters form the introductory part ,which describe the Database Lifecycle, from Requirements Analysis to Physical Design. The authors then discussed extensively the Entity-Relationship Modeling and the UML chapters where in they focused the ER constructs and basic UML notation used.

In the next part the authors described the Data Modeling. Requirements Analysis and Conceptual Modeling . Further they addressed the description of how the Conceptual Data Model to SQL can be transformed with good number of illustrations. These illustrations explain the relationship types.

Next chapter is about transforming the Conceptual Data Model to SQL and contains a very useful set of figures that summarize how different relationship types, i.e. one-to-one, one-to-many, many-to-many, are translated into sets of SQL table creation constructs, including primary and foreign key attributes.

In the further chapter in the discussion on Normalization, normal Forms are explained here, with focus on practical discussions. Finally, there is a discussion on 'An example of Logical Database Design'.

The features of this book are the good illustrations, extensive bibliography and descriptions of modeling and applications. The chapter on business intelligence is intended to address the business readers.

This book content is presented in a simple as well as elegant way and it is so lucid.

Daisy Jacobs
University of Zululand
South Africa