An investigation into System Security Requirements and Implementation in an Application Service Provider (ASP) Environment

M.Sc. Thesis

By

Sabrina McNeely, B.Sc.

Department of Mathematics & Computing,

Galway-Mayo Institute of Technology, Galway.

Research Supervisor

John Healy

Submitted to the Higher Education and Training Awards Council,

July 2006

Declaration

I hereby declare that the work presented in this thesis is my own and that it has not been used to obtain a degree in this Institute of Technology or elsewhere.

_____

Sabrina McNeely

To my husband Kenneth and my family,
for their continuous encouragement and support.

Table of Contents

List of Figures

Acronyms

3DES	Triple Data Encryption Standard
ACL	Access Control Lists
AES	Advanced Encryption Standard
AH	Authentication Header
APIs	Application Programming Interfaces
ASCII	American Standard Code for Information Interchange
ASP	Application Service Provision
ASPs	Application Service Providers
CCTV	Closed Circuit Television
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
CORBAsec	CORBA Security Service Specification
COTS	Commercial Off The Shelf
DCOM	Distributed Component Object Model
DES	Data Encryption Standard
DMZ	Demilitarised Zone
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
EFF	Electronic Frontier Foundation
EIS	Enterprise Information Systems
EIT	Enterprise Integration Technologies
EJB	Enterprise JavaBeans
ESP	Encapsulating Security Payload
EU	European Union
FTP	File Transfer Protocol
FTPS	File Transfer Protocol over Secure Sockets Layer
GUI	Graphical User Interface
HTTP	Hypertext Transport Protocol
HTTPS	Hyper Text Transport Protocol over Secure Sockets Layer

IAB	Internet Architecture Board
IDC	International Data Corporation
IDL	Interface Definition Language
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPsec	Internet Protocol Security
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISO	International Standards Organisation
IT	Information Technology
ITU	International Telecommunications Union
J2EE	Java Platform Enterprise Edition
JAAS	Java Authentication and Authorisation Service
JCA	Java Cryptography Architecture
JCE	Java Cryptography Extension
JDBC	Java Database Connectivity
JDK	Java Development Kit
JRMP	Java Remote Method Protocol
JSP	JavaServer Pages
JSSE	Java Secure Socket Extension
KPIs	Key Performance Indicators
LAN	Local Area Network
MAC	Message Authentication Code
MD5	Message Digest algorithm 5
MIT	Massachusetts Institute of Technology
CryptoAPI	Crypto Application Programming Interface
MTS	Microsoft Transaction Server
MVC	Model View Controller
NIST	National Institute of Standards and Technology
NTLM	NT LAN Manager
ODL	Object Definition Language

OMG	Object Management Groups
ORB	Object Request Broker
OSI	Open Standards Interconnect
PINs	Personal Identification Numbers
PKI	Public Key Infrastructure
RDBMS	Relational Database Management System
RMI	Remote Method Invocation
RMI/IIOP	Java RMI over Internet Inter Orb Protocol
RPC	Remote Procedure Call
SaaS	Software as a Service
SDLC	Software Design Life Cycle
S-HTTP	Secure Hypertext Transfer Protocol
SLAs	Service Level Agreements
SMTP	Simple Mail Transfer Protocol
SSL	Secure Sockets Layer
SQL	Structured Query Language
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TSSG	Telecommunications Software & Systems Group
UML	Unified Modelling Language
URL	Uniform Resource Locator
UTF-8	Unicode Transformation Format
VPN	Virtual Private Network
VPNs	Virtual Private Networks
WAN	Wide Area Network
WSS	Web Services Security
WS-Security	Web Services Security
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XP	eXtreme Programming

Acknowledgements

I would like to express my appreciation to everyone who encouraged and assisted me throughout the course of this research project. In particular, I would like to thank Mr John Healy for his technical and academic support and guidance. Thanks also to Mr Gabriel Hicks, Head of the Department of Maths and Computing, and Dr John Lohan, Head of Research, for facilitating this research project. Thanks to the lecturers and the technicians, in the Department of Maths and Computing, for their constant support and assistance. Thanks to family and friends for their continuous understanding and encouragement. A special word of thanks to Patricia Breslin for proofreading this thesis. Finally, words alone cannot express the thanks I owe to Kenneth, my husband and best friend, who was beside me every step of the way.

Abstract

Although the ASP model has been around for over a decade, it has not achieved the expected high level of market uptake. This research project examines the past and present state of ASP adoption and identifies security as a primary factor influencing the uptake of the model. The early chapters of this document examine the ASP model and ASP security in particular. Specifically, the literature and technology review chapter analyses ASP literature, security technologies and best practices with respect to system security in general. Based on this investigation, a prototype to illustrate the range and types of technologies that encompass a security framework was developed and is described in detail. The latter chapters of this document evaluate the practical implementation of system security in an ASP environment. Finally, this document outlines the research outputs, including the conclusions drawn and recommendations with respect to system security in an ASP environment. The primary research output is the recommendation that by following best practices with respect to security, an ASP application can provide the same level of security one would expect from any other n-tier client-server application. In addition, a security evaluation matrix, which could be used to evaluate not only the security of ASP applications but the security of any n-tier application, was developed by the author. This thesis shows that perceptions with regard to fears of inadequate security of ASP solutions and solution data are misguided. Finally, based on the research conducted, the author recommends that ASP solutions should be developed and deployed on tried, tested and trusted infrastructure. Existing Application Programming Interfaces (APIs) should be used where possible and security best practices should be adhered to where feasible.

Published work associated with this Thesis

Sabrina McNeely, Kenneth Kirrane, John Healy and Sean Duignan
Perception: The Real Inhibitor to ASP Adoption?
Information Technology & Telecommunications (IT&T) Conference, National Maritime College, Cork Institute of Technology, Cork – 2005.

Chapter I : Introduction and Thesis Structure

1.1 Introduction

The term Application Service Provision (ASP) is defined as the supply of online software functionality to multiple clients on a subscription/rental basis, remotely via the Internet or via a private network [2]. The ASP provider agrees a service level with several organisations and each client is provided access to the software on a one-to-many basis. The ASP model dates back to the late 1990's, although it is still classed by many as a new and emerging trend. Despite the initial hype, uptake of the ASP model has been slow to materialise. Several researchers highlight security, availability and scalability as the primary inhibitors with regard to the realisation of an ASP solution. This thesis examines system security requirements in an ASP environment and provides an objective evaluation based on a practical implementation.

In order to gain a thorough understanding of security in an ASP environment, the author adopted a multilateral view of security. Firstly, the author examined ASP security from a high level perspective, by dividing security into three distinct considerations: physical security, solution security and the security/integrity of client data. Secondly, the author reviewed ASP security from an operational perspective, by evaluating the relevance of both a security policy and a disaster recovery plan to ASP security. Thirdly, the author delved beneath the surface of ASP security and investigated security across all tiers of the service architecture, through the analysis of application, middleware, platform and database security technologies. Finally, the author gained an understanding of security best practices, through the exploration of both current technology standards and security standards such as the Open Standards Interconnect (OSI) security model 7498-2 and the International Standards Organisation (ISO) 17799 security standard.

The extensive research conducted provided the author with the knowledge required for the practical implementation of a security framework for an ASP billing solution known

formally as Billing4Rent. The Billing4Rent project was realised using the industry proven Java Platform Enterprise Edition (J2EE) development environment, interacting with an Oracle 9i database, deployed on an Apache Tomcat webserver. The practical implementation was supplemented through the documentation of best practices with regard to the development of both a security policy and a disaster recovery plan. The entire Billing4Rent security framework enabled the author to perform an objective evaluation of the practical implementation of system security in an ASP environment using the knowledge gained from a comprehensive system security literature and technology review.

1.2 Thesis Motivation

Enterprise Ireland, along with Industry partners Intec and an interested 3rd Party, provided the finance for the implementation of an ASP billing solution formally known as Billing4Rent. It is envisaged that the aforementioned project will lead to the formation of Billing4Rent Ltd. which will be tasked with bringing an ASP-based billing solution to market.

The Billing4Rent solution enables tier 3 and 4 network operators, content providers, service aggregators and other genres of service providers to access state-of the-art billing functionality on a subscription or a rental basis. Despite clear market demand from small and entrant providers, there is no similar service currently operational. This can be attributed to the fact that there are significant barriers inhibiting the realisation of such a solution. These include the huge amount of intellectual property embedded in incumbent billing vendor's commercial-off-the-shelf (COTS) products and the major technical challenges that need to be addressed to support a scalable, secure and robust ASP-based solution that can be effectively used by personnel with limited technical expertise.

The Billing4Rent solution was implemented as part of a joint venture between The Telecommunications Software & Systems Group (TSSG) at the Waterford Institute of

Technology and the Informatics Research Group at the Galway-Mayo Institute of Technology.

1.3 Thesis Objectives

The Informatics Research Group at the Galway-Mayo Institute of Technology had overall responsibility for two distinct areas of research -:

- An Investigation into system security requirements and Implementation in an Application Service Provision (ASP) environment.

- A Framework for high availability and optimum system performance in an Application Service Provision (ASP) environment.

This thesis is concerned solely with the former, 'An Investigation into system security requirements and Implementation in an Application Service Provision (ASP) environment'. It is expected that this investigation will yield a comprehensive framework of best practice with respect to system security within the ASP domain.

The primary objective of the programme of research can be broken down into five distinct goals -:

- Perform a comprehensive literature review of ASP material and system security requirements in particular.

- Review of technical material with regard to the implementation of system security in general and in a J2EE application server environment, the development platform chosen for the realisation of the ASP solution, in particular.

- Examination of security best practices, through the exploration of both current technology standards and security standards such as the Open Standards Interconnect (OSI) security model 7498-2 and the International Standards Organisation (ISO) 17799 security standard.

- Implementation of system security in the Billing4Rent ASP project based on the project goals and the aforementioned research.

- An objective comparison of the practical implementation of system security in the Billing4Rent ASP environment with both the comprehensive ASP system security literature and technology review and established best practices in middleware security.

1.4 Thesis Structure

The remaining chapters of the thesis are structured as follows:

Chapter II describes the research framework used. This includes an overview of both exploratory research and agile development methodologies, and their applicability to the programme of research.

Chapter III details the results of a comprehensive literature and technology review of ASP and system security material, focusing primarily on security requirements in an ASP environment. It examines security technologies across all tiers of the service architecture and outlines the role of security policies, disaster recovery plans, service-level agreements and security standards in an ASP environment.

Chapter IV introduces the Billing4Rent solution prototype developed to enable tier 3 and 4 operators and communications providers take advantage of a low cost billing service. It details the implementation of system security in the Billing4Rent project and

illustrates the range and types of technologies that are encompassed by the security framework.

Chapter V provides an objective comparison and analysis of the practical implementation of system security in the Billing4Rent environment using the best practices and frameworks described in the system security literature and security technical material review.

Chapter VI concludes the thesis with a summary of the work completed, the conclusions drawn and provides a list of recommendations for future work.

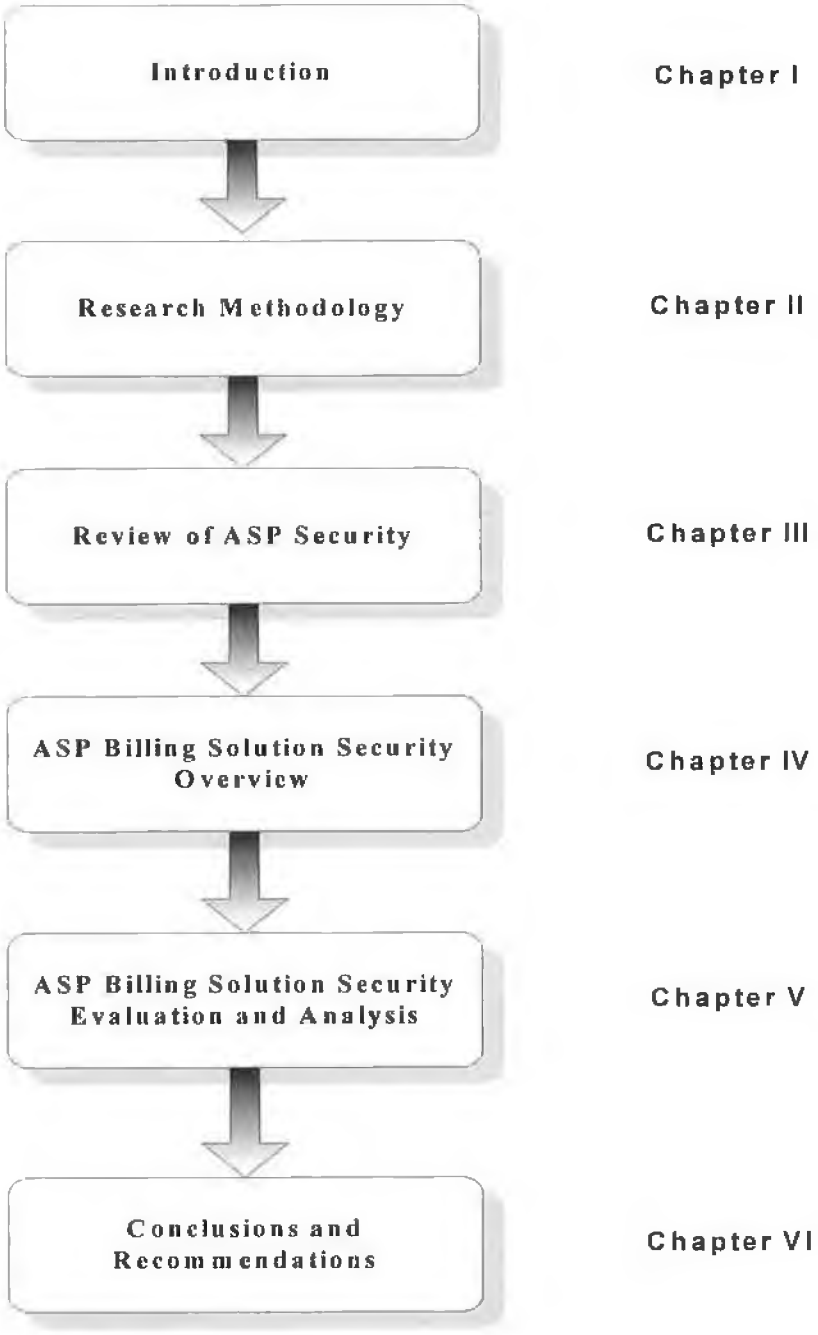


Figure 1.1 – Thesis Structure

Chapter II : Research Framework and Methodologies

2.1 Introduction

This chapter outlines the author's approach to the work and the methodologies applied throughout the course of the research. In order to gain an in-depth understanding of Application Service Provision (ASP) and ASP security in particular, the author adopted an exploratory research methodology. The knowledge gained from the exploratory research laid the groundwork for the implementation of a secure Billing4Rent ASP solution using an agile development methodology.

2.2 Exploratory Research Methodology

Exploratory research is defined by Crawford as a means of providing a better understanding of the research problem [2]. *"Exploratory research often relies on secondary research such as reviewing available literature and/or data, or qualitative approaches such as informal discussions with consumers, employees, management or competitors, and more formal approaches through in-depth interviews, focus groups, projective methods, case studies or pilot studies [1]."*

In order to fully understand the nature of ASP and, in particular, system security requirements in an ASP environment, the author conducted an extensive literature review. The literature review was further supplemented by a comprehensive technology review to evaluate the practical implementation of system security in a J2EE application server environment. In addition, the author examined security best practices, through the exploration of both current technology standards and security standards such as the Open Standards Interconnect (OSI) security model 7498-2 and the International Standards Organisation (ISO) 17799 security standard. Finally, the Billing4Rent case study enabled

the author to perform an objective evaluation of the practical implementation of system security in an ASP environment using the knowledge gained from a comprehensive system security literature and technology review.

2.3 Development Methodology

The waterfall model is the classic software lifecycle model. The model describes a development methodology that is both linear and sequential. The agile development methodology provides an alternative development methodology to the popular waterfall model. However, agile development can actually incorporate the waterfall model on a small scale, by splitting the project into several iterations and repeating the entire waterfall cycle for each iteration. Alternatively, development can be carried out simultaneously so that there are no distinguishable phases e.g. eXtreme Programming (XP) [3], [4].

2.3.1 The Waterfall Model

The waterfall model is a sequential software engineering process where development is seen as flowing steadily downwards, like a waterfall, through the phases of development life cycle. The waterfall model, documented in 1970 by W. W. Royce, was the first publicly documented life cycle model [5]. Ironically, Royce himself advocated an iterative approach to software development.

Overview of waterfall model development phases (Figure 2.1):

- Analysis of the problem and documentation of requirements.
- Documentation to describe in detail how the system should be built.
- Development of the system based on the design documentation.
- Unit and system testing of the system.

- System integration and end-user testing.
- Software is updated to correct errors previously undetected in the testing phases and to meet the changing customer needs.

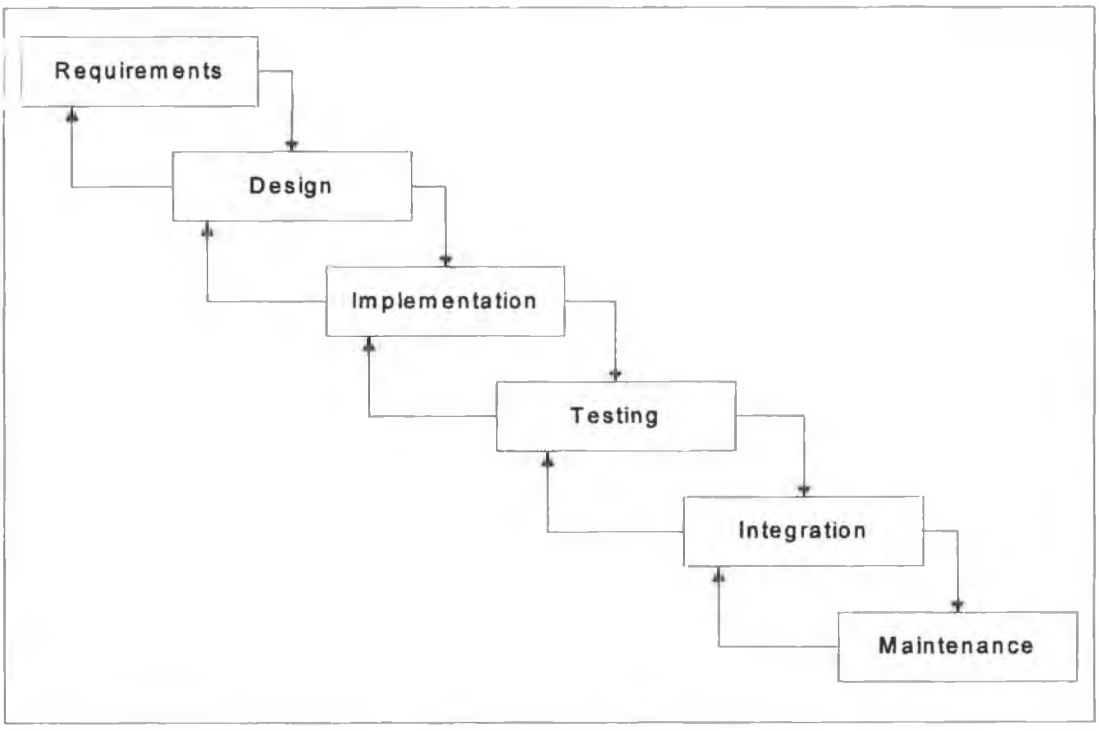


Figure 2.1 – Waterfall Model

The primary advantages of the waterfall model are the clearly defined milestones and deliverables and the model's emphasis on documentation throughout the entire lifecycle. A major drawback of the model is the difficulty of accommodating change throughout the course of the project. Figure 2.1 depicts the waterfall model.

2.3.2 The Agile Development Methodology

In 2001, a group of seventeen software methodologists met in Utah to bring together the various development methodologies that had been developed over the previous decade or

so. They amalgamated the principles of those methodologies into a simple statement, now known as the Agile Manifesto [4], [6]:

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- *Individuals and interactions over processes and tools*
- *Working software over comprehensive documentation*
- *Customer collaboration over contract negotiation*
- *Responding to change over following a plan*

That is, while there is value in the items on the right, we value the items on the left more [6]."

Although there are a number of agile software development methods, this section concentrates solely on the XP and the iterative waterfall development methodologies.

2.3.2.1 eXtreme Programming (XP)

XP is a set of values, principles and practices that facilitate the rapid development of software that provides the highest value for the customer in the fastest way possible. Kent Beck, Ward Cunningham and Ron Jeffries devised XP during their work on the Chrysler Comprehensive Compensation System (C3) project [1]. XP takes twelve software development best practices and applies them to the extreme.

Overview of XP core practices:

- **Planning:** Customers define and prioritise the system features with user stories.
- **Small Releases:** Functionality is released into production early and often, adding additional features with each release.
- **System Metaphor:** XP teams use a common system of names and descriptions.

- **Simple Design:** XP teams emphasise simply written, object-oriented code that meets requirements.
- **Continuous Testing:** Designers write automated unit tests prior to development and run them throughout the project.
- **Refactoring:** XP teams frequently revise and edit the overall code design.
- **Pair Programming:** Programmers work side by side in pairs, continually seeing and discussing one another's code.
- **Collective Code Ownership:** All programmers have collective ownership of the code and the ability to change it.
- **Continuous Integration:** XP teams integrate code and release it to a repository every few hours and under no circumstance hold on to it longer than a day.
- **40 Hour Work Week:** Programmers work only 40 hours per week.
- **On-site Customer:** Development teams have continuous access to a customer representative throughout the development project.
- **Coding Standards:** Programmers must follow a common coding standard so all the code in the system looks as if a single individual wrote it.

The primary advantages of the XP methodology are customer involvement throughout the project, the emphasis on teamwork and communication and the flexibility to deal with changing requirements. However XP is not without its flaws. The main disadvantages of the XP methodology is that it is code-centered instead of design-centered and the lack of emphasis on project documentation.

2.3.2.2 Iterative Waterfall Development Model

The iterative waterfall agile development methodology subdivides the development lifecycle into several iterations. Each iteration is in itself a self-contained subproject composed of activities such as requirements analysis, design, implementation, testing, integration, and maintenance [4]. The iterative waterfall agile development methodology benefits from the advantages of the pure waterfall model such as clearly defined

milestones/deliverables and its emphasis on documentation throughout the entire life cycle. However, this model supplements the pure waterfall model through incremental development and by facilitating changing requirements throughout the course of the project.

2.4 Chapter Summary

Due to the unpredictable nature of the Billing4Rent project the author elected to adopt an agile development methodology. The iterative waterfall agile development methodology was chosen over the XP agile development methodology as the development team did not have continuous access to a customer representative and the team was separated geographically. The Billing4Rent project was subdivided into a number of distinct iterations. Each iteration was in turn subdivided into the six waterfall phases i.e. requirements analysis, design, implementation, testing, integration, and maintenance. The industry standard Unified Modelling Language (UML) was used by the author to design the Billing4Rent security framework. As the development platform was subject to change, the author adopted an iterative approach to the development of a loosely-coupled architecture.

Chapter III : Literature and Technology Review

3.1 Introduction

This chapter details the results of a comprehensive literature and technology review of Application Service Provision (ASP) and associated system security material, focusing primarly on security requirements in an ASP environment. The chapter is further broken down into a number of distinct sections that:

- Provide an overview of the ASP model by drawing comparisons with traditional technology models, such as time-sharing and outsourcing. ASP solution implementations past, present and future, in the technology sector are examined, in order ascertain the risks and benefits of the ASP model.

- Examine ASP system security, paying particular attention to physical security, solution security and the security and integrity of customer data. The roles played by Service Level Agreements (SLAs) and independent trust service providers are reviewed from an ASP environment perspective.

- Explore existing security technologies and how they ensure the privacy, security and integrity of data stored locally and transferred over the Internet. In particular, the security provided by both the Java platform and Oracle's 9i database offering are investigated in order to ascertain best practice with regard to security.

- Detail existing network and information security standards such as the Open Standards Interconnect (OSI) security model 7498-2 and International Standards Organisation (ISO) 17799.

3.2 Application Service Provision (ASP)

ASP refers to the supply of online software functionality to multiple clients on a subscription or rental basis, remotely via the Internet or a private network. Application Service Providers (ASPs) agree service levels with several organisations and each client is provided with access to the software on a one-to-many basis. On-demand and utility computing are alternative terms for ASP. However, these models often incorporate concepts such as server or storage leasing. The ASP model dates back to the late 1990's, although it is still classed by many as a new and emerging trend [7], [8], [9].

3.2.1 Comparisons of ASP with Traditional Technology Models

Although ASP is still a nascent technology, the concept stems from both traditional technology models of time-sharing and outsourcing. Traditional mainframe systems are a prime example of time-sharing, given that multiple users share the mainframe resources. Technology may have changed dramatically over the past 40 years, but its application appears to have come full circle. Substitute 'application service provider' for 'time-sharing' and this 1967 definition still works well: *"time-sharing is a communication-oriented method of using computers"* [10]. Braunstein [11] also makes a comparison between the ASP model and time-sharing, by specifying that although the term ASP is relatively new, the concept of application hosting dates back to the 1960s. Known then as time-sharing, corporations and educational institutions would take turns renting mainframe-processing capabilities, as no single organisation could afford the total cost of computing [11].

Traditionally many businesses found it cost effective to outsource certain business functions. Outsourcing typically involves employing a 3rd party to provide services on behalf of an organisation. Most outsourcing initiatives are driven by both economic and strategic motives [12]. ASP gives a new dimension to outsourcing and is thus perceived

by many, as the third wave of outsourcing (B. Desai et al 2003, refers to Currie and Seltsikas 2001) [7].

3.2.2 Implementation of the ASP Model Past, Present and Future

In spite of the promise and potential of improving the way organisations develop, operate and maintain Information Technology (IT) applications, ASPs have fared poorly in terms of attracting a large client base [13]. High growth level predictions, by leading consulting firms such as Dataquest, Tankee Group, Forrester Research and International Data Corporation (IDC) group varied considerably to anywhere between $7 billion and $23 billion by 2003 [14], [12], [8], [15]. Although ASPs have not achieved the rapid market acceptance foreseen by many researchers, some proponents argue that the attractiveness of the ASP model is currently on the increase [16], [17]. Seltsikas and Currie believe the future is still bright for ASPs, stating that the ASP industry remains immature and will likely evolve into a more consolidated market, where the term ASP is likely to be superseded by software-as-a-service (SaaS) [17]. Current estimates indicate that worldwide spending on SaaS and associated software license revenue will reach $13.4 billion by 2007 [8], [18].

3.3 Application Service Provision (ASP) System Security

Several researchers refer to security as one of the key factors influencing the uptake of the ASP model [10], [7], [19]. Fears of inadequate security and privacy have prevented many firms from fully investigating and integrating the ASP business model [7]. Winch [19] outlines data security and integrity, along with disaster recovery and backup, as the two most important Key Performance Indicators (KPIs) for potential and existing ASP customers. A survey conducted by B. Desai and W. Currie [20], designed to capture the KPIs used to evaluate the benefits and risks relating to the ASP model, further strengthens the notion that data security and integrity, along with disaster recovery,

backup and restoration as key reasons why customers are reluctant to adopt the ASP model [20]. Although many researchers touch on the area of security on examination of the ASP model, surprisingly there is a distinct lack of in-depth research into the key factors influencing the adoption of the ASP model and ASP security in particular.

Tao [14] specifies that ASP security can be divided into both client data and server availability. However, Linthicum [21] and Anderson [22] take the concept of ASP security one step further, breaking down server availability into hardware and software security breaches. Anderson [22] examines security problems through its classification into three areas: hardware, software and data. Linthicum [21] adds weight to the above, citing examples of each type of security breach. Firstly, inadequate solution security may result in unauthorised network access. Secondly, inadequate physical security may result in internal security breaches. Finally, he poses a question as to protection of sensitive data from competitors using the same ASP solution [21].

Anderson [22] and Linthicum [21] subdivide ASP security into three distinct considerations: physical security, solution security and the security and integrity of client data. However, no overview of system security would be complete without an examination of a security policy and a disaster recovery plan, and their direct correlation with each of the above considerations. Based on the results of the aforementioned survey conducted by B. Desai and W. Currie [20]; disaster recovery, backup and restoring procedures were equally relevant as data security and integrity with regard to the uptake of the ASP model.

3.3.1 Physical Security

Physical security is concerned with securing the ASP solution hardware and its location against theft, tampering and damage, either intentional or unintentional [23], [24]. Bhagyavati and Hicks [25] add weight to this definition, stating that physical security involves the locking up of assets such as networking infrastructure, computing systems

and data storage, in order to provide protection from unauthorised monitoring, theft, corruption, and natural disasters. An organisation wishing to protect valuable data in a computing system should devise a physical security policy, which offers deterrence to an attacker and controls access to the system. [23]. Wang [26] defines hardware security as the physical protection of devices through the provision of a secure environment. In general, data centers are used to house electronic equipment, such as computer systems and communications equipment. ASPs may choose to maintain their own data centers, or alternatively they may opt to outsource data center services to a third party. The major benefit of adopting an outsourcing strategy is that organisations are free to concentrate on core business [27], [28]. In either case it is a primary responsibility of the ASP hosting company to safeguard electronic and telecommunications equipment against environmental damage, in addition to physical damage or theft by both employees and individuals external to the organisation. Baldwin, Shiu and Mont [29] highlight the fact that contrary to popular belief, the majority of security problems are, of course, not results of external intruders but are the result of internal attacks. Anderson [22] furthers this hypothesis stating that employee actions, both intentional and unintentional can be extremely damaging to the organisation.

Examples of physical security breaches:

- ASP solution hardware and communications equipment may be damaged or destroyed by natural disasters, electrical surges, fire or water.
- Access to the data center may be compromised and ASP solution hardware may be damaged, tampered with or stolen by unauthorised personnel.
- ASP solution hardware may be damaged, tampered with or stolen by disgruntled or former employees.
- Authorised personnel may inadvertently damage ASP solution hardware.

Decisions made with regard to the location, construction and layout of the data center, are often based on efficient workflow, with security being a secondary consideration [30]. The location and layout of the data center is relevant to physical security from environmental, internal and external security perspectives. In areas prone to natural

disasters such as floods, earthquakes and tornados, special care needs to be given to the construction and the layout of the data center. Data centers meet the security needs of ASPs as they invest heavily in security systems including sophisticated personal controls. Personal controls include fingerprint or IRIS identification, along with passwords and armed guard protection of facilities [10]. It is often the creation of a secure and survivable operational environment that proves to be a challenge, rather than the creation or integration of the software system [29].

Recommendations for ensuring physical security of ASP solution hardware:

- ASP solution hardware should be placed in a secure data center.
- ASP solution hardware should be further secured physically, through the use of security cables, padlocks and other such devices.
- Access to the data center should be restricted to authorised personnel, through the use of biometric scanners plus user pins or passwords.
- Access control policies and procedures should be implemented and adhered to by all personnel e.g. a policy of two-in-two-out should be implemented in order to gain access to the data center.
- Closed circuit television should be installed in order to monitor activities throughout the data center.

3.3.2 Solution Security

Solution security is concerned with restricting access to the ASP solution to authorised personal and securing both the exchange and the storage of customer data within the ASP environment. As ASPs supply online software functionality on a subscription/rental basis, they are particularly vulnerable to penetration by outsiders. ASP solutions are susceptible to threats at the remote terminal and along the communications link, in addition to threats at the physical machine [31]. Particular attention should be paid to the security risk associated with badly written software and poorly configured solutions, as many of the well-publicised computer security and virus problems relate to bugs in

software, errors in code or application logic [32], [33], [34] and poorly configured solutions. As such it is not surprising that best practices recommend considering security early in the Software Design Life Cycle (SDLC), knowing and understanding common threats, designing for security, and subjecting solutions to thorough objective risk analyses and testing [35].

Examples of solution security breaches:

- Software bugs (errors in code or application logic) and poorly configured systems leave the computer systems open to external attacks.
- Computer systems may become infected by viruses (malicious applications loaded onto a computer system without ones knowledge) and worms (malicious applications that can replicate themselves across a network). Both viruses and worms could compromise ASP solution usability or availability.
- Computer systems may become infected by spyware (software that gathers information from a computer system without ones knowledge and relays that information via the internet to the creator) and adware (software that gathers information from a computer system without ones knowledge and uses that information to display relevant advertisements). Both spyware and adware could compromise sensitive data stored on machines in an ASP environment in addition to solution usability or availability.
- Hackers or crackers (individuals who attempt to gain unauthorised access to computer systems with the sole purpose of causing damage or stealing information) may gain unauthorised access to sensitive data stored in the ASP environment.
- The security and integrity of sensitive data exchanged over communication links may be compromised if unencrypted data is intercepted by external entities.
- Denial-of-service attacks such as the:
    - Consumption of scarce, limited, or non-renewable resources.
    - Destruction or alteration of configuration information.
    - Physical destruction or alteration of network components.

- Other attacks such as 'ping of death' and 'teardrop' may compromise ASP environment usability or availability, by overloading or crashing the computer system.

It is a primary responsibility of ASPs to guard against malicious attacks that monopolise communications and server resources [14] and compromise system usability or availability. On a technical level, the solution needs to take care of identity management, authentication, authorisation, audit, assurance of communication, information retention and disaster recovery [29]. Identity management, authentication, authorisation and assurance of communication are all relevant to our exploration of solution security. Access to computer systems and authorisation for use of computer systems services and software components in an ASP environment should be kept to an absolute minimum. Both auditing or logging and disaster recovery will be dealt with in the discussion with regard to the documentation and implementation of an ASP security policy and disaster recovery plan.

Recommendations for ensuring the physical security of an ASP solution:

- The ASP solution should be tested rigorously to ensure that neither the software nor the configuration of the ASP environment pose a security risk.
- Appropriate service packs and patches should be applied to both the operating system and applications as soon as they are made available.
- Anti-virus and anti-spyware software should be installed on all machines and updated periodically.
- Firewalls should be configured to filter undesired traffic between the Internet and the ASP solution.
- Ensure there are no unnecessary services running on the ASP solution hardware by shutting down unused ports and daemons.
- Access to ASP computer systems should be restricted to authorised personnel by enforcing strong passwords.
- Remote access to ASP solution hardware should be limited to an absolute minimum.

- Data sent over the Internet should be encrypted to ensure the security and integrity of the data being transmitted.

3.3.3 Security and Integrity of Client Data

Data security is also concerned with ensuring the privacy and integrity of client data in an ASP environment. Data security is intrinsically twofold: it includes the privacy protection and the soundness of the stored information [36]. ASP clients ascribe a significant amount of risk to trusting an external organisation with vital data and critical applications [10]. As each client is provided access to the ASP solution software on a one-to-many basis, it is critical that the ASPs reassure clients that their data is protected from their competitors.

Examples of data security and integrity breaches:

- Sensitive customer data may be compromised if data stored on ASP solution hardware is not permanently erased prior to disposal.
- Software bugs and poorly configured systems may result in sensitive data falling into the wrong hands.
- Poor solution security may result in sensitive data falling into the wrong hands.

There is a distinct overlap between both ASP physical and solution security, and the security and integrity of customer data, as unauthorised access to the ASP hardware and/or solution may result in the security and integrity of client data being compromised. ASPs must establish stringent procedures to avoid compromising the integrity of customer data while it is under their care [14]. ASPs need to be aware of their legal obligations with regard to the protection of client data. In Europe this legislation comes in the form of the 'Privacy and Electronic Communications Directive' and the 'Data Protection Directive':

Privacy and Electronic Communications Directive (Directive 2002/58/EC):
Directive 2002/58/EC is composed of a set of European Union (EU) rules to ensure the protection of privacy and personal data in electronic communications. The directive includes provisions on security of networks and services, confidentiality of communications, access to information stored on terminal equipment, processing of traffic and location data, calling line identification, public subscriber directories and unsolicited commercial communications [37].

The Data Protection Directive (Directive 95/46/EC):
Directive 95/46/EC is composed of a set of EU rules to safeguard personal data by providing technology-neutral legislation that is applicable to all types of current electronic communications. Under this directive, EU member states are responsible for the protection of a persons fundamental rights and freedoms, and in particular their right to privacy with respect to the processing of personal data. However, in doing so they must ensure that they do not restrict or prohibit the free flow of personal data between member states [38].

Recommendations for ensuring data security in an ASP environment:

- Ensure redundant hardware is disposed of in an appropriate manner. Software and data should be uninstalled and erased to guarantee that sensitive data is not accessible to unauthorised individuals.
- The ASP solution software and the configuration of the ASP environment should be tested rigorously to ensure the security and the integrity of client data.
- Solution security threats should be identified and the appropriate action should be taken to combat unauthorised access to sensitive data.

3.3.4 ASP Security Policy and Disaster Recovery Plan

"The only system which is truly secure is one which is switched off and unplugged, locked in a titanium lined safe, buried in a concrete bunker, and is surrounded by nerve gas and very highly paid armed guards. Even then, I wouldn't stake my life on it" (G. Spafford). Although Spafford's [39] statement may seem very far-fetched, the underlying meaning is well grounded in reality. Unfortunately, no system will ever be 100% secure and thus it is crucial that ASP hosting companies identify potential security issues and follow up by documenting and implementing an effective security policy and an efficient disaster recover plan. Avolio [40] strengthens the above hypothesis stating that there is no such thing as complete security in a usable system and consequently, it is important to concentrate on reducing risk as opposed to wasting resources trying to eliminate risk completely.

3.3.4.1 Security Policy

ASPs and, where appropriate, data centers are advised to devise a security policy, which meets the needs of the ASP and their clients. The function of a security policy is to outline guidelines for ensuring optimal security of the ASP solution. The Internet Engineering Task Force (IETF) [41] defines a security policy in the form of a formal statement of the rules by which people who are given access to an organisation's technology and information assets must abide. A security policy is an ever-changing document, constantly amended to cater for updates to an organisation's systems and procedures. The security policy should be composed of a combination of guidelines with regard to physical security, solution security and the security and integrity of information.

Both risk analysis and system monitoring are essential to the creation of an effective security policy. Risk analysis is defined as the identification of the most probable threats to an organisation and the analysis of the related vulnerabilities of the organisation to these threats [42]. The objective of risk analysis is to identify business processes and

supporting technologies, examine the possible threats to both with a view to at best eliminating the threat, or at worst minimising the potential impact of the threat. Although risk analysis is an efficient means of protection from known threats, it is generally ineffective against unperceived threats. System monitoring, both passive and active, provides a successful means of highlighting unforeseen threats in a timely manner and thus limits their impact [41], [43]. In both cases, system and network activity is continuously logged. However, active monitoring provides the added benefit of triggering an event when a pre-specified condition is met.

3.3.4.2 Disaster Recover Plan

A disaster recovery plan is defined by Toigo [44] as a set of activities intended to prevent avoidable instances of unplanned interruption, regardless of cause, and to minimise the impact of interruption due to unavoidable events. A disaster recovery plan should be implemented, to ensure ease of recovery in the event of a natural disaster, a man-made disaster or any event, which causes interruption to normal business processes. Earthquakes, hurricanes and tornadoes are examples of natural disasters; terrorism, theft and arson are examples of man-made disasters; viruses, security breaches and denial-of-service attacks are examples of interruption to normal business processes. A disaster recovery plan should incorporate ASP business processes in addition to systems, networks and information, given their importance to the success of the ASP solution.

Recommendations for creating an optimal disaster recovery plan:

- Prioritise ASP business processes according to their criticality to the organisation.
- Design a complete network diagram that includes details of system redundancy and failover procedures.
- Provide in-depth detail of backup procedures for ASP solution hardware, software and business processes. This detail should include the method and frequency of backups and the location of remote backup facilities.

- Detail recovery procedures for ASP solution hardware, software and business processes to ensure a timely recovery in the event of a disaster.

It is essential that down-time in an ASP environment be kept to an absolute minimum. In order to meet this requirement, it is imperative that the disaster recovery plan is updated regularly to reflect changes to ASP systems and business processes, and tested regularly to ensure both validity and ease of recovery in the event of a disaster.

3.3.5 Service Level Agreements (SLAs) and Trust Service Providers

The success of the ASP model depends on the establishment of trust between the parties involved. SLAs are essential to the success of the ASP model and are fundamental to meeting or exceeding client expectations and developing a high degree of trust between both parties. SLAs outline the relationship between the service provider and the client. They detail the agreed ASP solution and outline expectations with regard to availability, performance and security [45].

Independent Trust Service Providers also play a significant role in establishing trust, by alleviating client concerns with regard to security and privacy. ASPs who display a trust seal obtained from an Independent Trust Service Provider, demonstrate that they have undertaken an independent auditing process to verify their solution meets International Trust Services Standards. Both SLAs and trust seals assist in the elimination of fears of inadequate security and privacy.

3.4 Security Technologies

The increasing growth in the number of computer networks, the use of the Internet and the use of distributed computing applications, has increased concerns with regard to privacy, security and integrity of information exchange, and further increased interest in

the use of technologies to protect information transferred over these networks [46]. This section examines the various system, network and information security technologies in use today.

3.4.1 Cryptography

Cryptography is the practice and the study of encryption and decryption and comes from the Greek word 'kryptos' meaning hidden and 'graphia' meaning writing [47]. Encryption is the translation of data into an unintelligible form through the use of a secret code, in order to ensure confidentiality. Encrypted data is referred to as 'cipher' or 'cipher text', whereas unencrypted data is commonly known as 'plain text'. Decryption is the process of restoring data to its original form through the use of a secret code. Encryption is an effective means of ensuring the privacy and integrity of information stored locally or transferred over a network.

Data stored locally:
Sensitive data stored locally should be encrypted to ensure that data cannot be interpreted by individuals gaining unauthorised access to the computer system. Encryption thus provides additional protection in the event local or remote computer system security breaches.

Data transmitted over a network:
Sensitive data transmitted over a network should be encrypted to ensure that data cannot be interpreted should a third party intercept it. Figure 3.1, adapted from [48], provides an example of how the privacy and integrity of data transferred over the Internet may be compromised. Joe could intercept unencrypted data transferred over the Internet from Mary to John. Both John and Mary would be unaware that Joe has read or updated Mary's personal message.

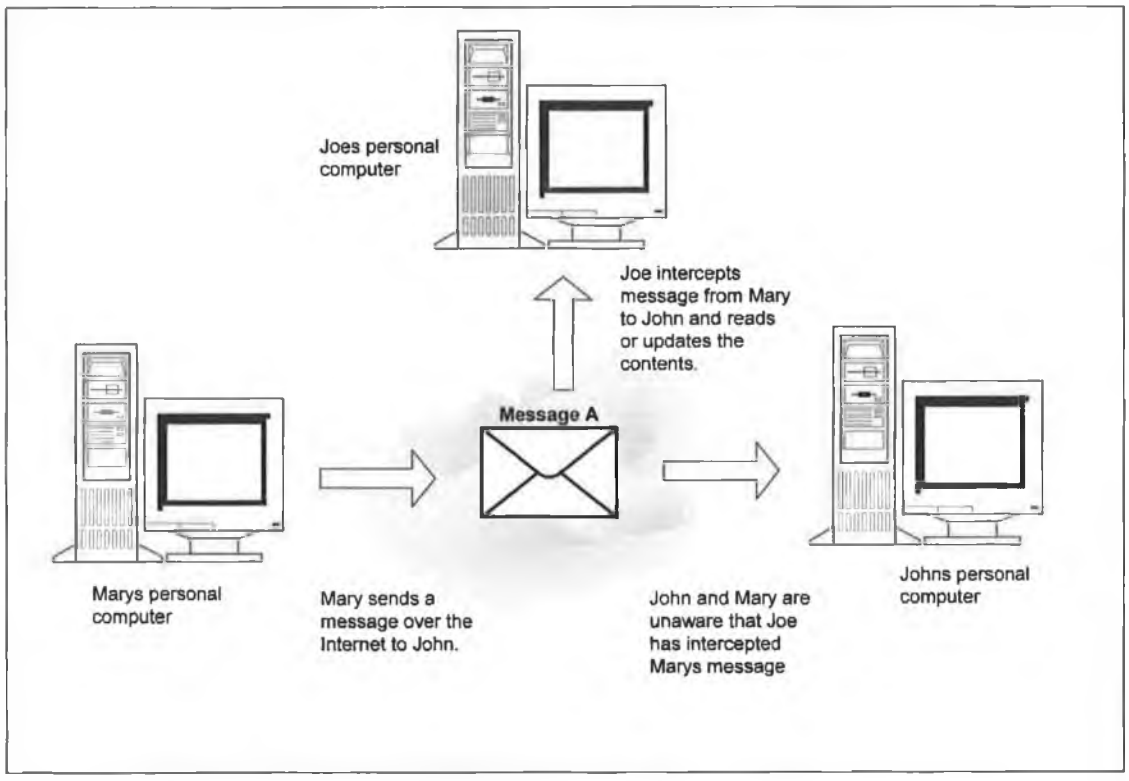


Figure 3.1 – Interception of unencrypted data

The interception of unencrypted sensitive data (i.e. credit card details, bank details, sensitive company data etc.) transferred over the Internet, could have disastrous consequences. The solution is for both parties to agree on a method of encryption, which can be used to encode the data prior to it being sent over the Internet.

This section examines two alternative strategies for data encryption: conventional encryption otherwise known as symmetric encryption and public key encryption otherwise known as asymmetric encryption.

3.4.1.1 Symmetric key encryption

Computers wishing to communicate securely, agree on a secret key, which will be used to both encrypt and decrypt data in the form of packets sent over a network [49], [50], [51],

[48], [52]. Figure 3.2 highlights the use of symmetric key encryption in network communication.

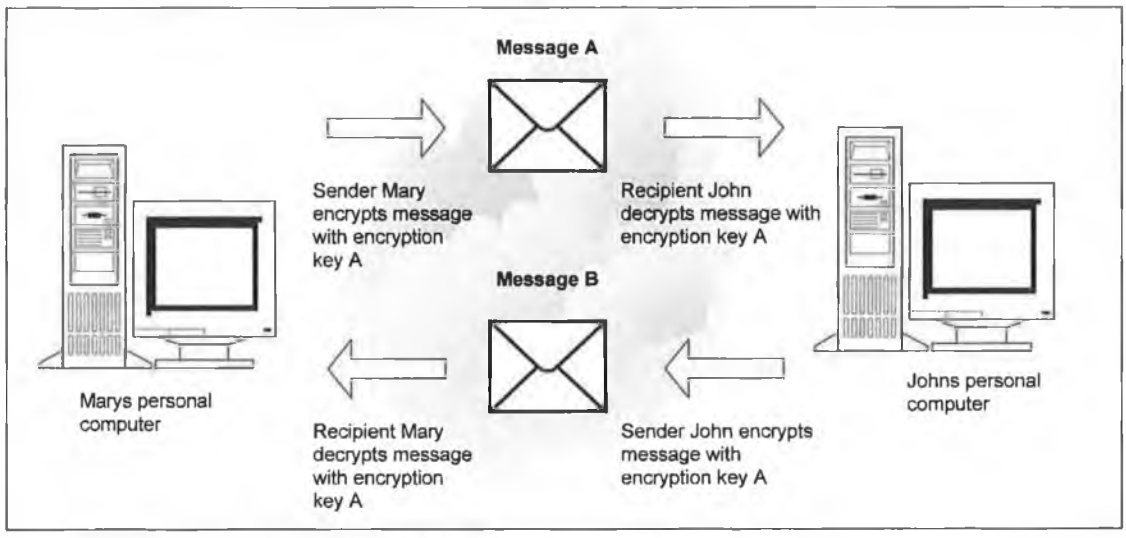


Figure 3.2 – Symmetric key encryption

The Data Encryption Standard (DES), Triple Data Encryption Standard (3DES) and Advanced Encryption Standard (AES) are the three most prominent symmetric key encryption algorithms. DES was developed by IBM and was adopted by the National Institute of Standards and Technology (NIST) as a standard in 1977 [49], [50], [51], [48]. DES divides plaintext into 64 bit blocks and uses a cipher key consisting of 64 binary digits, 56 of which are randomly generated, the remaining 8 bits are used for error detection. The 56-bit key is used as a parameter for the encryption/decryption algorithm and is composed of sixteen iterations of substitutions and transpositions. Since its adoption as a standard, there have been concerns with regard to the level of security provided by DES. These concerns proved to be well justified. Due to advances in computing power, the Electronic Frontier Foundation (EFF) succeeded in breaking DES in 1998, using a special purpose DES cracking machine costing less than $250,000 [49], [50], [51], [48]. DES was eventually superseded by 3DES, which was incorporated as part of the Data Encryption Standard in 1999 [49]. 3DES uses three separate 56-bit cipher keys and three executions of the DES algorithm instead of one, thus giving an overall key length of 168 bits. 3DES was popular due to its enhanced security and reuse

of the DES algorithm; however due to its limited block size and inefficient implementation it in-turn was superseded by AES. In 2001 NIST published the AES standard based on the Rijndael algorithm submitted by Belgian cryptographers Joan Daemen and Vincent Rijmen [50], [48]. AES consists of 128-bit blocks with a 128-bit, 192-bit or alternatively a 256-bit cipher key length. The number of rounds/iterations of the encryption/decryption algorithm is dependent on the cipher key length. A major drawback of symmetric key encryption is the security risk associated with the distribution of the encryption/decryption key to all system users.

3.4.1.2 Public key encryption

In 1976, Whitfield Diffie and Martin Hellman, researchers at Stanford University, proposed a radically new kind of cryptosystem, one in which the encryption and decryption keys were different [49]. Known as public key encryption or asymmetric encryption, it provides a solution to the above key distribution problem through the use of a public key, which can be widely distributed and a private/secret key known exclusively by the recipient. Data encrypted using an encryption/decryption algorithm and public key, can only be decrypted using the encryption/decryption algorithm and the corresponding private key [49], [50], [51], [48], [52]. Figure 3.3 highlights the use of public key encryption in network communication.

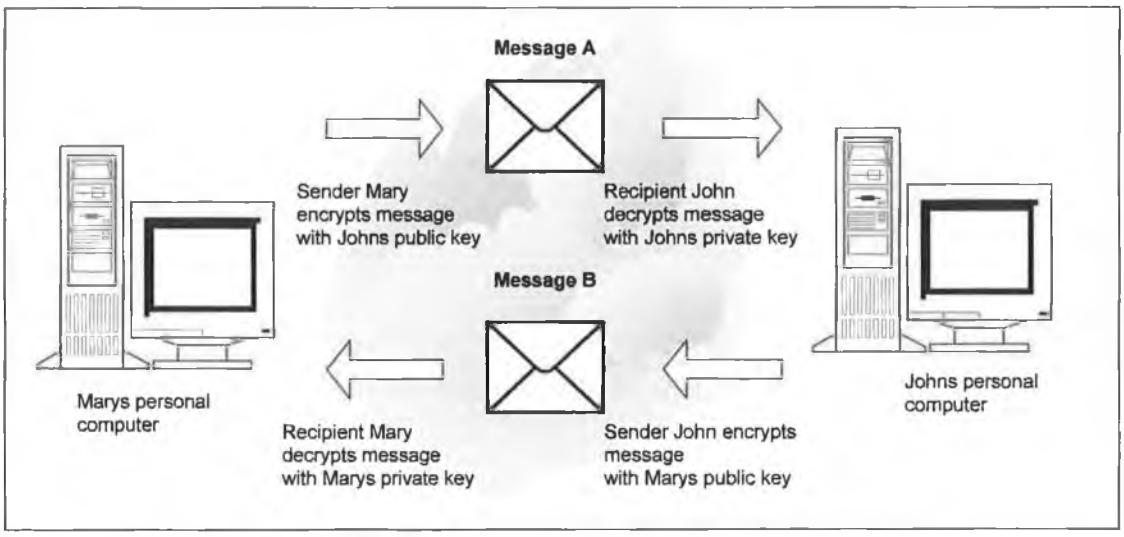


Figure 3.3 – Public key encryption

RSA encryption is an example of public key encryption and was developed in 1977 at the Massachusetts Institute of Technology (MIT), by professors Ron Rivest, Adi Shamir and Len Adleman [49]. The acronym RSA is constructed from the first initial of each of the researchers surnames. RSA encryption security is based on the difficulty in factoring large numbers, with a key that varies depending on the implementation used. RSA can also be used as a digital signature to authenticate the originator and ensure the integrity of the data. This thesis further examines digital signatures with regard to authentication in section 3.4.3.

3.4.2 Public Key Infrastructure (PKI)

A Public Key Infrastructure (PKI) comprises a system of certificates, certificate authorities, subjects, relying partners, registration authorities, and key repositories that provide for safe and reliable communications [53]. Certificate authorities and digital certificates play a major role in the distribution of public keys used for public key encryption. A certificate is a digitally signed binding between a public key and one or more attributes of its owner, such as name, e-mail address Uniform Resource Locator (URL) or properties that can be used to grant permissions or capabilities [54]. A

certificate authority is any organisation or individual that issues digital certificates i.e. signed certificates with a public key [50]. If party A and party B wish to communicate securely using public key encryption, how does party B know that the public key belongs to party A and not party C masquerading as party A? The solution to this problem is to obtain a digital certificate, which verifies details of the individual, organisation or server that owns the corresponding public key. A certificate authority is responsible for the issue of digital certificates and must confirm the identity of the certificate applicant. As the certificate authority is a trusted third party, anyone wishing to communicate securely with the owner of public key can confirm his or her identity through the digital certificate. The digital certificate also contains the digital signature of the issuing certificate authority, so that a recipient can verify that the certificate itself is valid. On receipt of the certificate, the recipient uses the certificate authority's public key to decode the certificate and retrieve the owner's identity information and public key. The public key is in turn used to send a secure message to the certificate owner. X.509 is the most widely used standard for digital certificates.

3.4.2.1 X.509 Standard

The International Telecommunications Union (ITU) developed X.509, the most widely used standard for digital certificates, in 1988 [49]. X.509 certificates are generated by trusted certificate authorities and are stored in an X.500 global directory, which is used as a central storage of digital certificates. X.500 is an ISO and ITU standard that defines the structure of global directories [47]. Figure 3.4, which outlines the X.509 V3 certificate components and corresponding overview, has been adapted from [47], [51], [48].

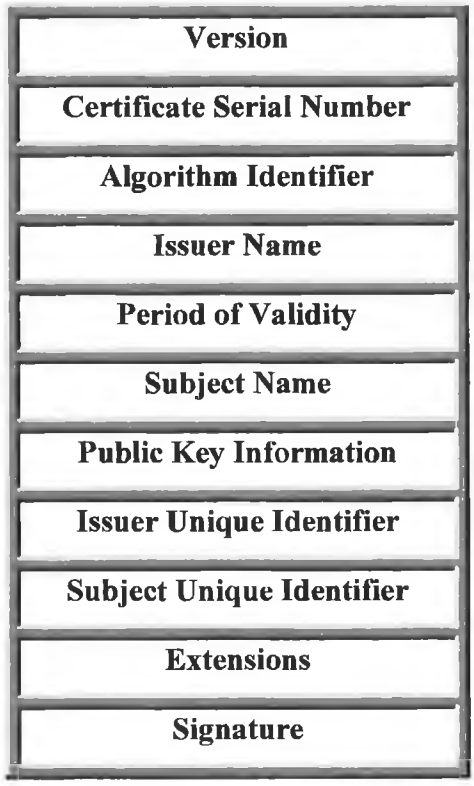


Figure 3.4 – X.509 V3 Certificate

Version:

Differentiates the version of the X.509 certificate formats, which determines the data stored in the certificate. X.509 has gone through two revisions since it was made available in 1988. The Issuer Unique Identifier and Subject Unique Identifier fields were introduced in X.509 V2, and the Extensions fields were introduced in X.509 V3 in 1996.

Certificate serial number:

Specifies an integer value, unique within the issuing certificate authority, assigned to the digital certificate by the certificate authority.

Algorithm identifier:

Identifies the algorithm and the associated parameters used to sign the digital certificate.

Issuer name:

Details the X.500 name of the certificate authority that created and signed the digital certificate.

Period of validity:

Consists of two dates, specifying from what date and to what date the digital certificate is valid.

Subject name:

Specifies the name of the entity who owns the digital certificate and the private/public key pair.

Public key information:

Identifies the public key of the entity that owns the digital certificate in addition to the algorithm and associated parameters used by the entity to sign messages.

Issuer unique identifier:

An optional ID introduced in X.509 V2. It is used to uniquely identify the certificate authority.

Subject unique identifier:

An optional ID introduced in X.509 V2. It is used to uniquely identify the owner of the private/public key pair.

Extensions:

A number of optional extensions were introduced in X.509 V3, consisting of an extension identifier, a criticality indicator and an extension value. Extensions provide a more flexible approach than adding additional fields to a fixed format.

Signature:

The certificate authorities digital signature is composed of a hash code of the other fields encrypted with the certificate authorities private key. In addition this field contains the algorithm and the associated parameters used by the certificate authority to sign the digital certificate.

3.4.3 Digital Signatures

A digital signature is an electronic signature, which allows the author of a digitally represented message to sign it in such a fashion that the signature has properties similar to a signature written in ink for the paper world [46]. It confirms the claimed identity of the originator and guarantees the validity of the message.

Digital signatures must have the following properties [47] [49]:

- The receiver can verify the claimed identity of the originator.
- The sender cannot later repudiate the contents of the message.
- It must be verifiable by third parties to resolve disputes.

Digital signatures are realised through the use of public key encryption. Encrypting the entire message with the originators private key or alternatively encrypting a hash code of the message with the originators private key constitutes a digital signature [48], [50], [51], [52]. In both cases, it is assumed that the recipient knows the originators public key and uses it to decrypt the message or hash code. By successfully decrypting the message or hash code with the originators public key, the recipient verifies both the originator and the validity of the message, as only the corresponding public key in the private key pair can decrypt the message or hash code. Figure 3.5 outlines the use of private and public keys for authentication.

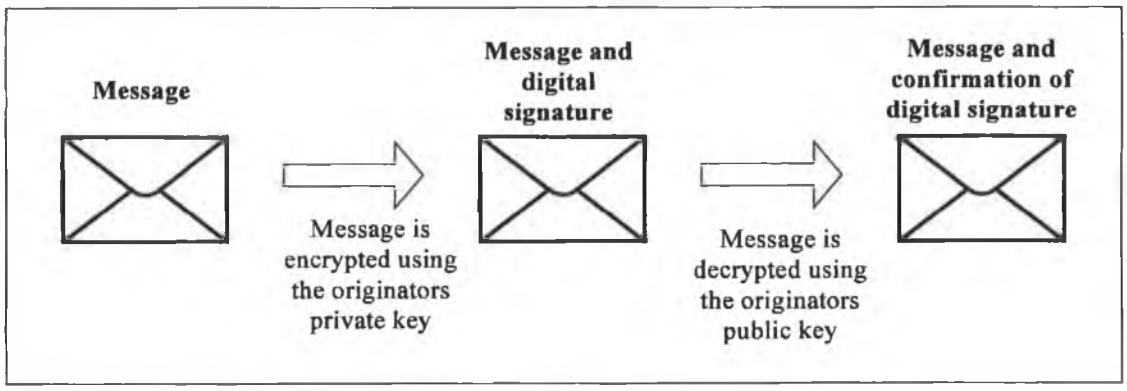


Figure 3.5 – Digital Signature

Although the recipient can verify the author of the message and if the message has been altered, they are unable to confirm if a third party has read the message. Digital signatures are used solely to verify identity and content and alone do not provide a means of confidentiality. To ensure confidentiality, the message should be further encrypted using either symmetric or public key encryption strategies. Figure 3.6 outlines the use of a digital signature and public key encryption.

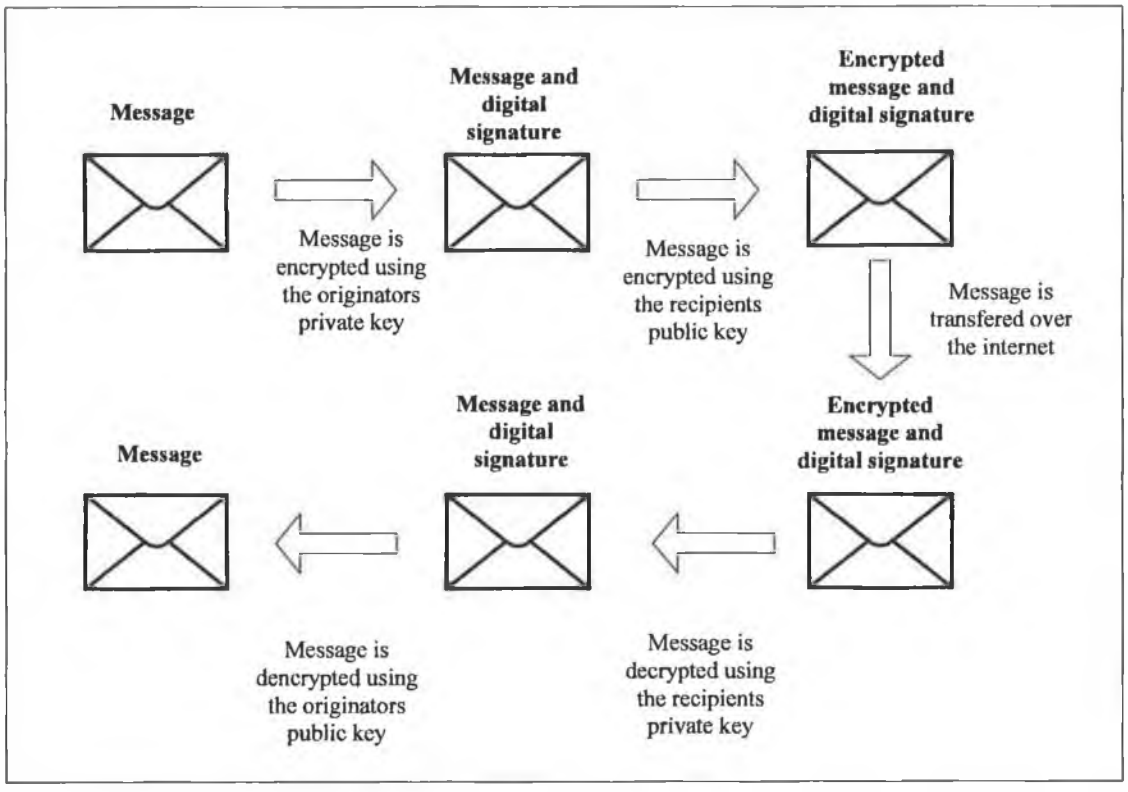


Figure 3.6 – Digital Signature and Encryption

Digital signature standard:

The Digital Signature Standard (DSS) is a cryptographic standard based on Digital Signature Algorithm (DSA), originally published by the National Institute of Standards and Technology (NIST) in 1991, revised in 1993 and again in 1996 [47], [51], [48]. DSS only provides authentication capability, unlike RSA and some DSA implementations that provide both authentication and encryption. A summary of the message data, called a message digest, is created through the use of a hash code. The message digest is used in conjunction with the DSA algorithm and a private key in order to create a digital signature. On receipt of the message and the message signature, the originator and integrity of the messages is verified using the message digest, hash function, DSA algorithm and originators public key.

3.4.4 Secure Sockets Layer (SSL) and Transport Layer Security (TLS)

Secure Sockets Layer (SSL) is a network protocol, which was originally developed by Netscape Communications in 1995. However, the protocol later gained the support of other Internet vendors and became the *de-facto* standard [47], [49], [50], [51], [55]. SSL is used to ensure secure communication over the Internet and provides authentication of the server, and where required the client, through the use of digital certificates in addition to data confidentiality and integrity through the use of encryption technologies. SSL operates at the transport layer, sits on top of the Transmission Control Protocol (TCP) and is independent of higher-level application layer protocols such as Telnet, File Transfer Protocol (FTP), Hypertext Transport Protocol (HTTP) etc. Both Netscape Navigator and Internet Explorer along with the majority of web servers support SSL. Users are able to confirm session security by the presence of 'https://' as opposed to 'http://' at the start of the URL at the top of the browser, or alternatively the presence of the lock icon in the lower-left corner of the screen.

The Transport Layer Security (TLS) working group was formed in 1996 when SSL was submitted to the IETF for standardisation [49]. The TLS protocol supersedes SSL and although TLS is based on SSL 3.0 with a few extensions, the two protocols are not interoperable. However TLS is backward compatible with SSL 3.0 [56].

SSL/TLS consist of the following four protocols:

- The Record Protocol
- The Handshake Protocol
- The Change Cipher Spec Protocol
- The Alert Protocol

Figure 3.7, which outlines the SSL components and corresponding overview, has been adapted from [47], [57].

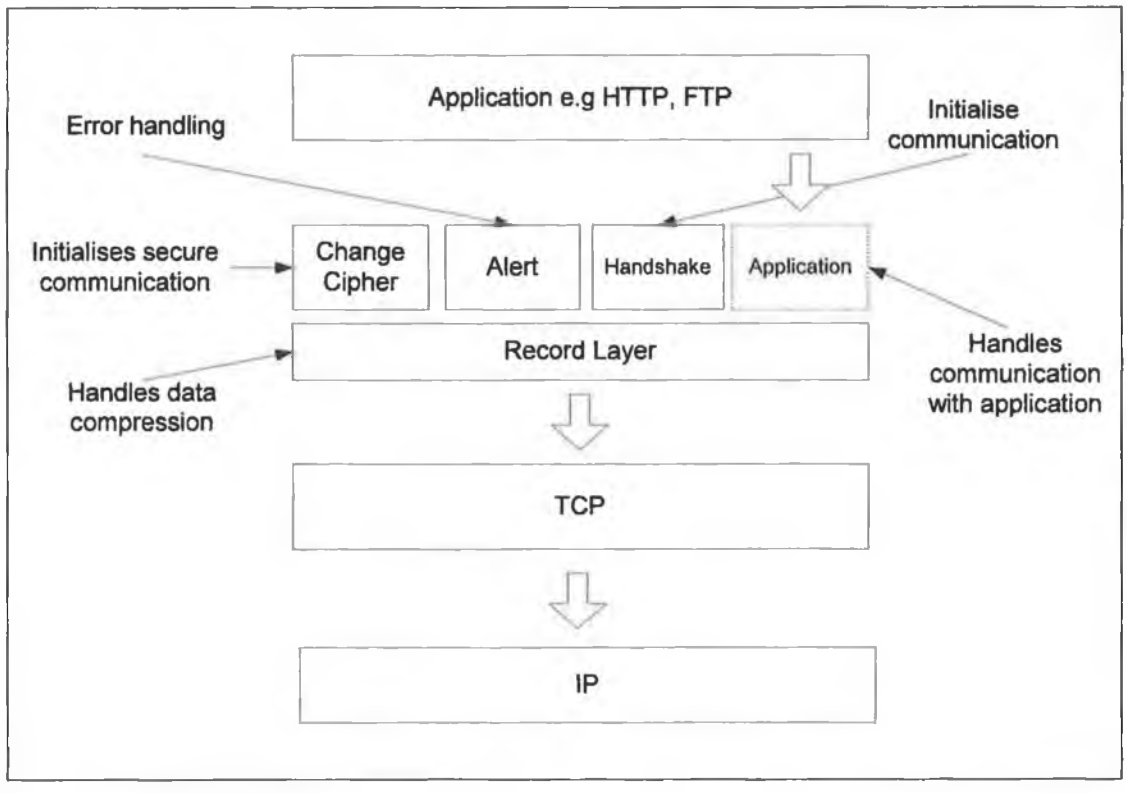


Figure 3.7 – SSL/TLS Protocol

The Record Protocol:

The record protocol provides both confidentiality and integrity of data transferred over the Internet. The record protocol is responsible for fragmentation and the optional compression of data, the application of a message authentication code to ensure integrity, the encryption of the data to ensure confidentiality, the addition of header data and the transmission of the data in the form of a TCP packet. The handshake protocol, change cipher spec protocol and alert protocol all sit on top of the record protocol.

The Handshake Protocol:

The handshake protocol allows the server and the client to authenticate each other and for both parties to agree on encryption and messaged authentication code algorithms and exchange secret keys.

The Alert Protocol:

The alert protocol is responsible for composing, compressing, encrypting and conveying alert messages. An alert consists of two bytes, one to convey the severity of the message and the other a description of the alert. Fatal alerts result in the immediate termination of the connection.

The Change Cipher Spec Protocol:

The change cipher spec protocol generates a message with a single byte of value 1, generated by both and server and the client and used to convert the current state to a pending state. This conveys to the recipient that subsequent transmissions will be protected under the negotiated algorithm and keys.

Hyper Text Transport Protocol over Secure Sockets Layer (HTTPS) and File Transfer Protocol over Secure Sockets Layer (FTPS):

HTTPS and FTPS are both application-layer protocols, which run on top of the SSL protocol. HTTP is a request/response protocol used to retrieve web resources from a server and render them in a client application. HTTPS is a secure version of HTTP where requests and responses are encrypted using SSL or TLS prior to being sent over the network. HTTPS uses port 443 as opposed to port 80, which is used by the HTTP protocol. FTP is a file transfer protocol used to download files from and upload files to an FTP server over the Internet. FTPS is a secure version of FTP, which uses SSL or TLS to transfer data securely. FTPS uses port 990 as opposed to port 21, which is used by the FTP protocol.

Secure Hypertext Transfer Protocol (S-HTTP):

S-HTTP is an extension of HTTP, developed by Enterprise Integration Technologies (EIT). S-HTTP should not be confused with the HTTPS protocol. S-HTTP enhances the HTTP protocol with security services at the application level and is an alternative to SSL, the socket level security protocol. S-HTTP provides confidentiality, authentication, and data integrity, however is not tied to any particular cryptography format or key infrastructure. SSL is designed to

establish a secure session, whereas S-HTTP is designed to send secure individual messages and are thus seen as complementary rather than competitive protocols.

3.4.5 Digital Identification

Digital identification is best described as the use of technology as a means of authentication and authorisation. The simplest of the systems are based on usernames/IDs and passwords, while others are based on special purpose hardware that can measure unique distinguishing human characteristics. Earlier this thesis examined the role played by digital certificates in message identification, thus this section concentrates on passwords, physical tokens and biometrics identification techniques.

Usernames/IDs and passwords:

Identification based on ID/password combination is the most fundamental form of digital identification. An ID and corresponding password are stored on the system for each user and are used as a means of authentication and authorisation. Each user must enter their password in order to gain access to the system. Personal Identification Numbers (PINs) are a type of password that are used in every day life to access computer systems such as secure buildings, voice mail systems, house alarms etc. Unfortunately passwords are subject to various security threats such as dictionary attacks and network eavesdropping [58]. In addition, Sutton [52] highlights the fact that a significant degree of responsibility with regard to ensuring password confidentiality is attributed to the password owner. Despite their well-known security weaknesses username password combinations remain the most common authentication mechanism [55].

Physical tokens:

Physical tokens such as magnetic access cards are an alternative means of authentication and authorisation, where access is granted based on the identity of the cardholder, which is determined through the use of magnetic strip readers. Every card has a unique number, which is associated with an identity. Physical tokens are used in order to gain access to

buildings, rooms, toll bridges, gyms etc. Often passwords or PINs are associated with the physical token, providing an extra level of security.

Biometrics:

Biometrics involves the use of measurable physical characteristics as a means of authentication and authorisation. Fingerprints, retina, iris and voice prints can all be used as a means of biometric identification. Ongoing biometrics and absolute identification are two alternative biometric techniques. Ongoing biometrics involves recording biometric information the first time an individual accesses a system, and on subsequent accesses the new biometric is compared with the stored record. Absolute identification involves the construction of a large database matching names and personal data with biometrics [50]. Although biometrics is a costly and complicated means of authentication and authorisation, they provide a higher level of security than passwords and physical tokens.

3.4.6 Firewalls

A firewall is software or hardware, or a combination of both, configured to control access between the Internet and hosts connected to a private network [49], [51]. Although access to the Internet is crucial for the ASP model, it also constitutes as a security threat to the organisation. The solution is to implement a firewall to filter all traffic to and from the organisation. However, as Wool highlights in [59], *"the protection that firewalls provide is only as good as the policy they are configured to implement."* Figure 3.8 illustrates the use of a firewall through the use of an organisation network diagram.

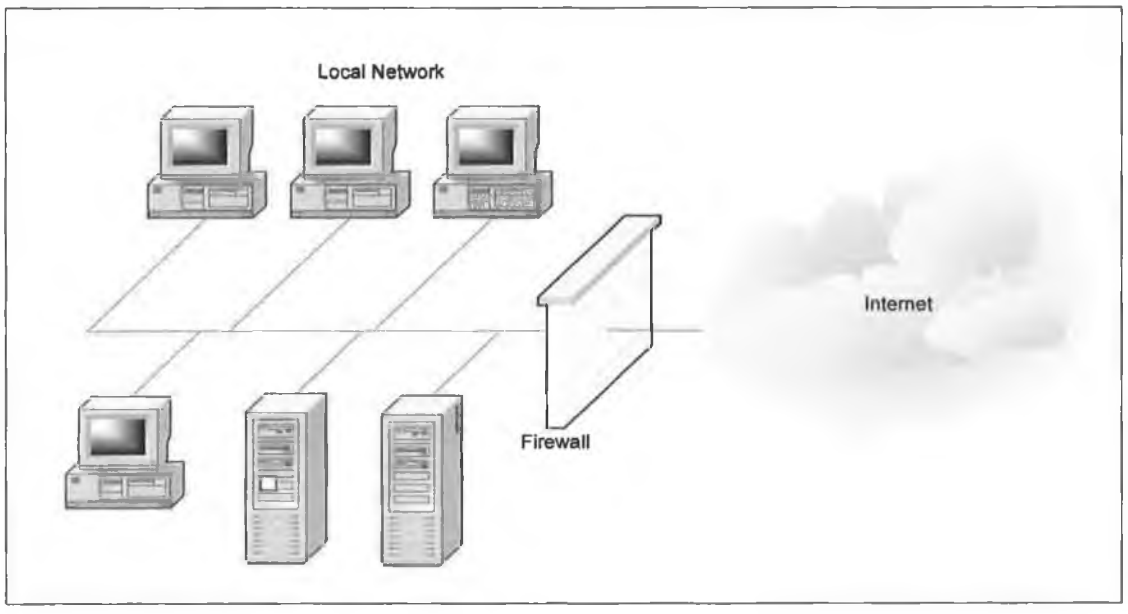


Figure 3.8 – Firewall network diagram

Packet Filtering Firewalls, Circuit-level Proxies and Application-level Proxies are the most common types of firewalls [51]. A proxy prevents direct communication between local node applications and external hosts.

Packet filtering firewalls:

Packet Filters inspect all packets sent to or from the local network and accept or reject the packet based on the source IP address, destination IP address, port number and a pre-defined set of rules. The network administrator is responsible for defining the firewall filtering rules. Organisations may wish to block incoming packets from all IP addresses and port 23 (used by TELNET), or one or more specified IP addresses and port 23, or alternatively allow incoming packets from a combination of one or more specified IP addresses and port 23. A default policy to either accept or reject a packet if no rule exists, should also be configured.

Application-level proxies:

Application-level proxies intercept all traffic and relay data packets back and forth between local node applications and an external host. Application-level proxies inspect

the entire packet and relay or reject the packet based on the header information and the contents of the packet.

Circuit-level proxies:

Circuit-level proxies intercept all traffic and relay data packets back and forth between local nodes and external hosts. However, circuit-level proxies do not examine the contents of the packet once the connection has been established. Circuit-level proxies control the flow of data at the session layer.

3.4.7 Internet Protocol Security (IPsec)

In 1994, the Internet Architecture Board (IAB) issued a report entitled "Security in the Internet Architecture", which stated the need for better Internet security. The report identified the need to secure network traffic from unauthorised monitoring and access through the use of authentication and encryption mechanisms [51]. IP Security (IPsec) is a set of protocols at the network layer, developed by the IETF, to facilitate the transfer of packets securely over the Internet, a Local Area Network (LAN) or a Wide Area Network (WAN) [50], [51]. IPsec works in conjunction with Internet Protocol version 4 (IPv4), the standard version of IP used on today's Internet and has been incorporated into Internet Protocol version 6 (IPv6), the next generation IP [50]. IPsec consists of two alternative protocols: Authentication Header and Internet Protocol (IP) Encapsulating Security Payload (ESP) and supports two encryption modes: transport and tunnel.

Transport mode:

Transport mode encrypts only the data/payload of each packet. However, the header remains unencrypted. In transport mode the IPsec header is inserted just after the IP header and contains security information, primarily the security association identifier (i.e. a simplex connection between two end points and associated security identifier), a new sequence number, and possibly an integrity check of the payload [49].

Tunnel mode:

Tunnel mode is the more secure option as it encrypts both the header and the data/payload and is useful when the tunnel ends at a location other than the final destination e.g. a firewall [49], [47], [51], [48]. In tunnel mode the entire IP packet is encrypted and encapsulated in the body of a new IP packet with a completely new IP header [49].

Figure 3.9, which provides an overview of IPsec protocol has been adapted from [51].

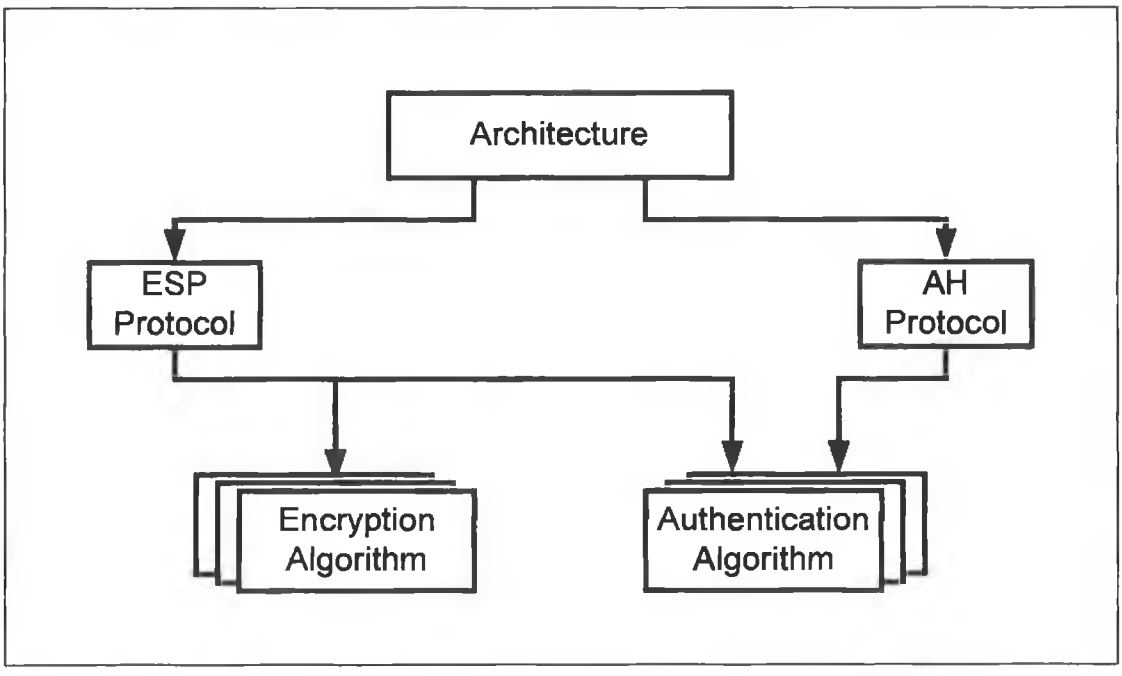


Figure 3.9 – IPsec Document Overview

IP Authentication Header (AH):

The IP Authentication Header (AH) provides authentication and integrity between hosts or gateways [47]. Figure 3.10, which outlines the IP Authentication Header (AH) transport mode, has been adapted from [49].

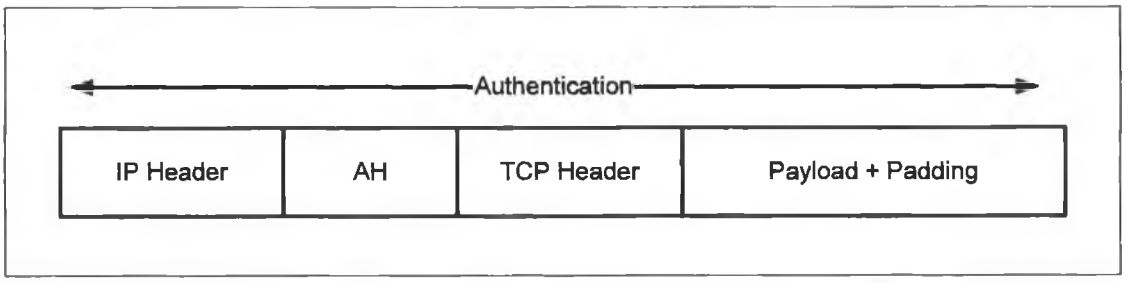


Figure 3.10 – IP AH transport mode

IP Encapsulating Security Payload (ESP):

The IP Encapsulating Security Payload (ESP) encrypts the data and places it as part of the ESP portion of the datagram/packet [47]. Figure 3.11(a) and 3-11(b), which outline the ESP Header in transport and tunnel mode respectively, have been adapted from [49].

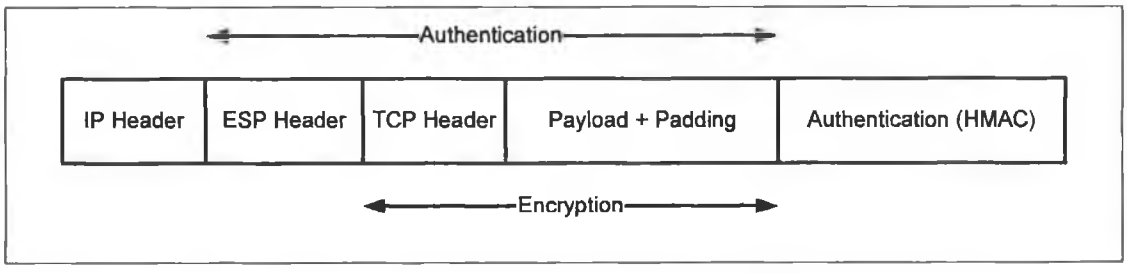


Figure 3.11(a) – ESP Header transport mode

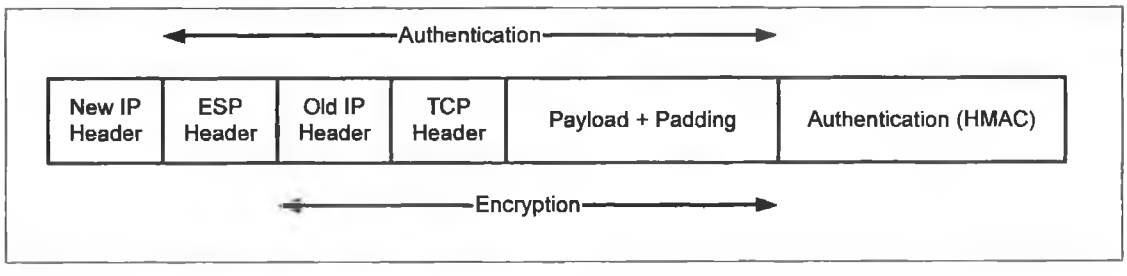


Figure 3.11(b) – ESP Header tunnel mode

IPsec enhances electronic commerce built-in security protocols, secures remote access and branch office connectivity over the Internet or a WAN through the use of a Virtual Private Network (VPN) [51].

3.4.8 Virtual Private Networks (VPNs)

Many national and international organisations lease telephone lines from telecom organisations, in order to link multiple sites. Commonly known as a leased-line private network, it provides a very secure means of communication, as intruders have to physically wiretap the lines in order to compromise data [49]. Although leased-line private networks are very secure they are also a very expensive. Figure 3.12, which outlines a leased line private network, has been adapted from [49].

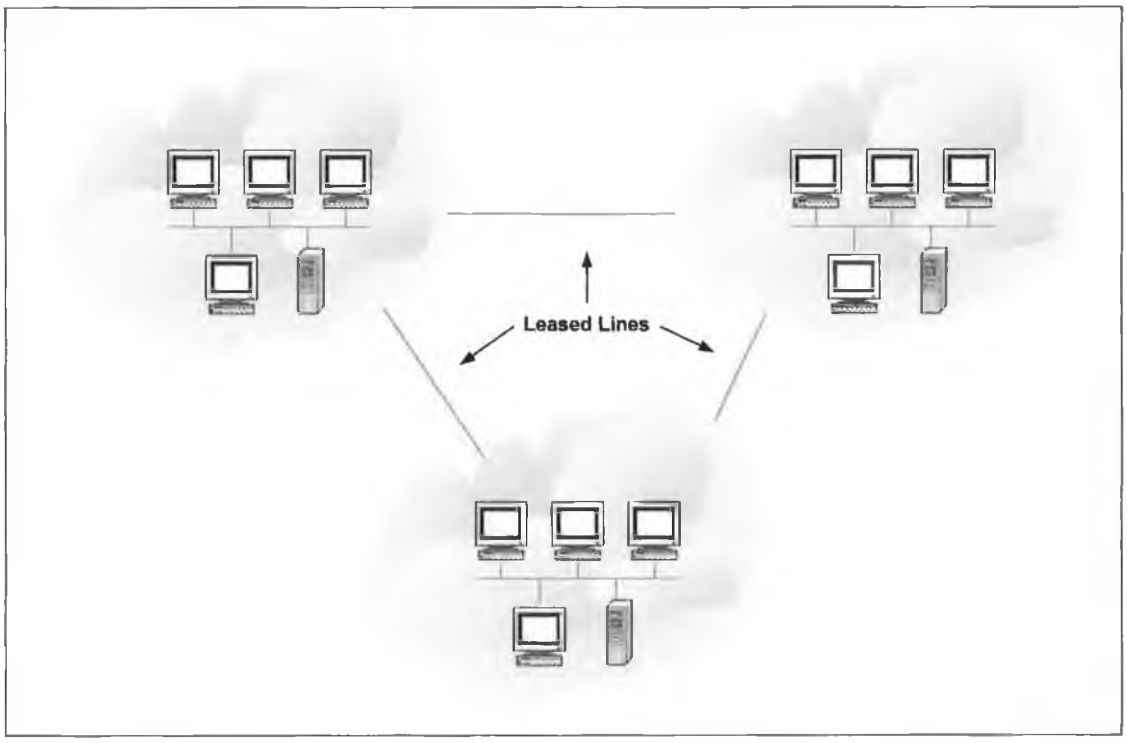


Figure 3.12 – Leased Line Private Network

The growth of the Internet provided a cheaper alternative to a leased line network solution, at the expense of network security. An effort to resolve the security issues, with regard to transferring sensitive data over the Internet, lead to the development of Virtual Private Networks (VPNs). Each site hosts a firewall that creates a tunnel through the Internet with each additional site using IPsec ESP tunnel mode. This tunnel provides a means of secure communication between each of the nodes on each connected site. If

site A and site B wish to communicate securely, their respective firewalls must negotiate the connection and maintain the IPsec tunnel. Once the tunnel has been established, machines on site A will appear as local machines to site B and vice versa. Machines on site A and site B appear to be on the same network, thus the name Virtual Private Network. Figure 3.13, which outlines a VPN, has been adapted from [49].

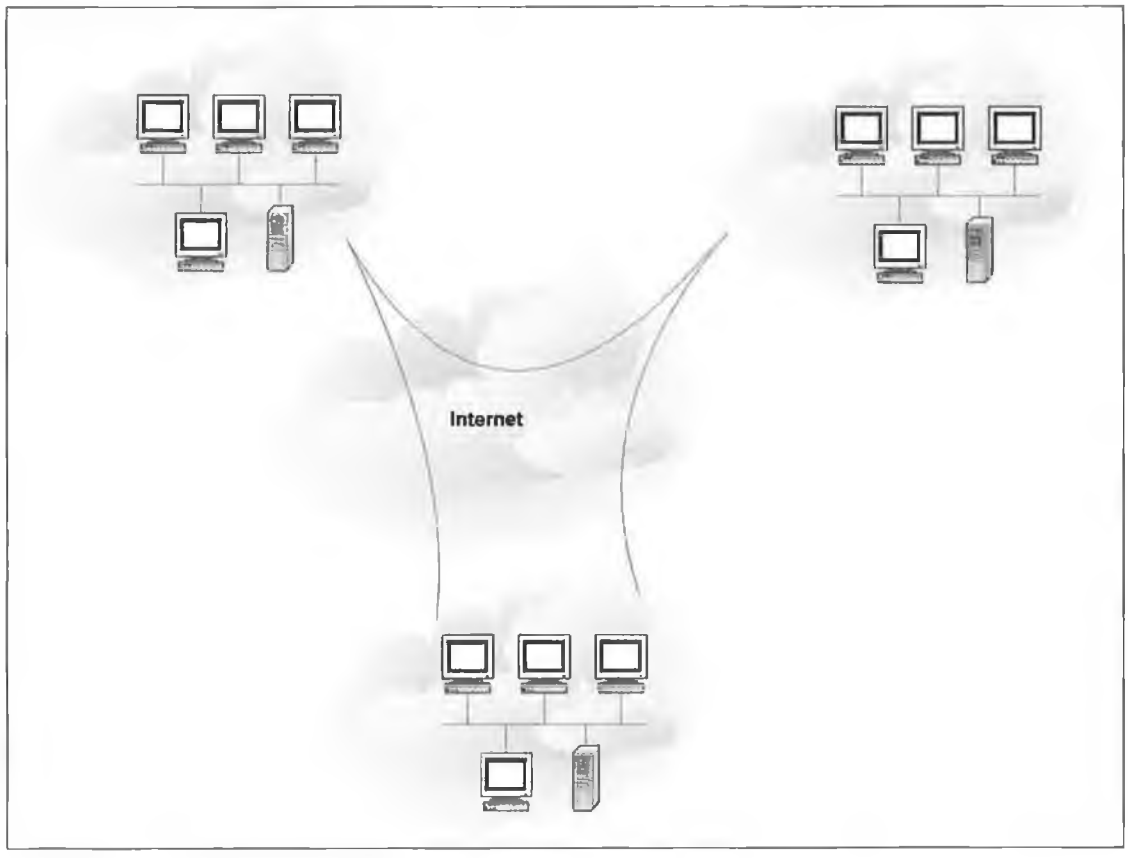


Figure 3.13 – Virtual Private Network

3.5 Middleware Security Technologies

Middleware refers to the software layer between the operating system, including the basic communication protocols, and the distributed applications that interact via the network [60]. This software infrastructure facilitates the interaction among distributed software

modules [60]. With the advent of Web Services, developing seamless interoperability between heterogeneous middleware technologies has become increasingly achievable [61]. However, the diversity and openness of distributed applications systems have given rise to questions of trust and security [62].

This section examines the security of three competing distributed object paradigms: The Object Management Groups (OMG) Common Object Request Broker Architecture (CORBA), Microsoft's Distributed Component Object Model (DCOM) and Sun Microsystems' Remote Method Invocation (RMI).

3.5.1 Common Object Request Broker Architecture (CORBA)

Common Object Request Broker Architecture (CORBA) technology is an open standard facilitating distributed computing in a heterogeneous environment. CORBA enables programmers to design and implement distributed applications in a standardised manner that guarantees portability and interoperability, following the object-oriented paradigm [62]. The OMG devised the CORBA 1.0 specification in 1991 [63]. The OMG is a consortium of more than 700 companies, which focuses on providing a common framework for developing applications using object-oriented programming techniques. The latest version CORBA 3.0 was released in 2002 [64].

The central component in the CORBA architecture, the Object Request Broker (ORB), was designed to provide only core functionality and consequently the CORBA Security Service Specification (CORBAsec), resides outside the ORB [62], [65]. CORBAsec defines a collection of core security interfaces, which are written in a consistent Interface Definition Language (IDL) [65]. The security interfaces that define services for authentication, confidentiality, integrity, access control, auditing and non-repudiation [62], are designed to ensure a reasonable level of security of a CORBA-compliant system as a whole [66]. In addition, CORBAsec describes an array of interfaces that can be used for the crucial task of security management/administration [62].

CORBAsec provides three distinct levels of increasing functionality:

- Level 0, which was not part of the specification in the beginning, integrates SSL into CORBA [62].
- Level 1, provides all the abovementioned security measures except for non-repudiation. This is done in such a way that the applications running on top of the CORBA system are not aware of the security features provided underneath [62].
- Level 2 provides all the abovementioned security measures including non-repudiation. Level 2 deals with those applications that are security aware and that are consequently in a position to interact with CORBAsec in order to specify the exact security features used [62], [67].

3.5.2 Distributed Component Object Model (DCOM)

DCOM is an extension of the Component Object Model (COM). COM objects are discrete components, each with a unique identity, which expose interfaces that allow applications and other components to access their features. DCOM interfaces are written in the Microsoft's interface definition language, the Object Definition Language (ODL) [68]. DCOM makes COM objects location independent and adds security and multithreading to COM [65]. Although DCOM hides many complexities of client server application development it has proven difficult to deploy in corporate environments where communication is performed across firewalls [69].

DCOM is designed with built in security. The NT LAN Manager (NTLM) and the Microsoft Transaction Server (MTS) authenticate users and authorise checking via Access Control Lists (ACL). The Microsoft Crypto Application Programming Interface (MS CryptoAPI) provides both data encryption and integrity, while the Authenticode SDK uses digital signatures to provide non-repudiation [65]. On the downside DCOM uses a binary format for method invocations, and because firewalls don't understand the content of the exchanged binary packages, they view the packages as a potential attack

and block the calls [70]. Though there is a way to overcome this limitation, the solution is regarded as weakening security [69]. These problems are addressed by a new framework, which is known formally as SOAP.

3.5.3 SOAP

SOAP is a lightweight, operating system independent XML-based messaging protocol used to encode the information in Web Service request and response messages before sending them over a network. Initially SOAP was an acronym for 'Simple Object Access Protocol', however in June 2003 the World Wide Web Consortium (W3C) dropped this acronym as it was considered to be misleading. SOAP is based on two common protocols: XML which is used as an encoding scheme for request and response parameters of the method calls and HTTP which defines how messages are formatted and transmitted [71]. While SOAP offers obvious benefits in the world of interoperability, it comes at a price of performance degradation and additional development efforts required for implementation of features missing from SOAP, such as security and state management [69].

The Web Services Security (WSS or WS-Security) standard outlines a standard set of SOAP extensions that facilitates the implementation of security token propagation, message integrity and message confidentiality. WS-Security is highly flexible and facilitates the construction of a wide variety of security models including PKI, kerberos and SSL. Specifically WS-Security provides support for multiple security tokens, multiple trust domains, multiple signature formats and multiple encryption technologies [72].

3.5.4 Remote Method Invocation (RMI)

RMI is a set of protocols developed by Sun Microsystems' JavaSoft division that enables Java objects to communicate remotely with other Java objects. RMI has many of the same features of other remote procedure call (RPC) systems, letting an object running in one Java virtual machine make a method call on an object running in another, perhaps on a different physical machine [73][74]. RMI uses the Java Remote Method Protocol (JRMP) for remote Java object communication. RMI is a relatively simple protocol, however unlike more complex protocols such as CORBA and DCOM, RMI only works with the Java programming language. However, Java RMI over the Internet Inter-Orb Protocol (RMI-IIOP) gives you RMI ease of use coupled with CORBA/IIOP language interoperability. Like RMI, RMI-IIOP provides flexibility by allowing graphs of objects to be serialised and transferred as pass-by-value method parameters. Like CORBA, RMI over IIOP is based on open standards defined with the participation of hundreds of vendors and users in the Object Management Group [75].

As RMI sits on top of the Java Virtual Machine it leverages Java's built-in garbage collection, security, and class-loading mechanisms [74]. The RMISecurityManager is an example security manager, which implements security policies similar to those provided by the security manager for applets, java applications included in a HTML page, which run in a browser [74]. It determines whether methods are invoked locally or remotely and protects against potentially unsafe operations [76], by providing a mechanism to "sandbox" the execution of downloaded code, so that it cannot adversely affect the local machine [74]. Applications must either define their own security manager or use the RMISecurityManager. The Java security manager thus facilitates the definition of a fine-grained security policy for both RMI client and server applications. It is worth noting that Enterprise JavaBeans (EJB) (a component architecture for multi-tier client/server systems) and Jini (an open architecture that enables the connection and sharing of devices, such as printers and disk drives, on a network) are both built on top of RMI.

3.6 Platform Security

The J2EE platform provides a secure development and runtime environment for Java applications. "*Data type checking at compile-time and automatic memory management leads to more robust code and reduces memory corruption and vulnerabilities. Bytecode verification ensures code conforms to the JVM specification and prevents hostile code from corrupting the runtime environment. Class loaders ensure that untrusted code cannot interfere with other Java programs*" [77].

The original Java platform provided two distinct levels of security. Local code was provided full access to vital system resources, while remote code was provided limited access to resources inside the sandbox. The platform was later updated to provide remote code, signed by a trusted entity, the same level of security as local code. Today the Java platform provides a much more fine-grained approach. All code regardless of whether it is local or remote can be subjected to a security policy [78]. The Java platform provides a set of Application Programming Interfaces (APIs) spanning several major security areas including cryptography, public key infrastructure, authentication, secure communication, and access control. This section examines how the aforementioned APIs supplement Java's core security architecture.

3.6.1 Java Authentication and Authorisation Service (JAAS)

Java Authentication and Authorisation Service (JAAS) is composed of a set of APIs that enable applications to both authenticate and enforce access controls upon users or entities such as services. JAAS authentication is performed in a pluggable fashion, which permits Java applications to remain independent from underlying authentication technologies [79]. Java applications interact with JAAS through an object called the LoginContext. The LoginContext, in turn, communicates with a login module, which is specified at run time and is responsible for implementing and performing the authentication. Because the LoginContext never changes and the login module exposes a

standard interface, no code change or recompilation is required to change authentication algorithms [80]. JAAS provides some reference LoginModule implementations, such as the JndiLoginModule, the NTLoginModule and the UnixLoginModule.

Once the user has been authenticated, the JAAS authorisation component works in conjunction with the existing Java access control model to protect access to sensitive resources. JAAS access control decisions are based on who is running the code, in addition to existing code-centric access controls [80] [81]. Once a user or entity, has been authenticated using JAAS, an associated subject is created. A subject is comprised of a set of principals, where each principal represents an identity for that user or entity. Permissions can be granted in the Java security policy for specific principals. For each subsequent security-checked operation, the Java runtime environment will automatically determine if the security policy grants the required permission to a principal associated with the authenticated subject [82].

3.6.2 Java Cryptography Extension (JCE)

Java Cryptography Extension (JCE) provides both a framework and implementations for encryption, key generation and key agreement, and Message Authentication Code (MAC) algorithms. JCE provides support for symmetric, asymmetric, block, and stream cipher encryption [83]. JCE supplements core Java security such as digital signatures or one-way hash functions. The Java Cryptography Architecture (JCA) is included in the Java run-time environment distributed by Sun and includes algorithms to perform message digests, create digital signatures, and generate key pairs. JCE extends the JCA by providing algorithms to generate single keys and secret keys in addition to encryption and decryption cipher algorithms [84].

3.6.3 Java Secure Socket Extension (JSSE)

Java Secure Socket Extension (JSSE) is a set of packages that enable secure Internet communications. JSSE provides both a framework and an implementation for a Java version of the SSL and the TLS protocols and includes functionality for data encryption, server authentication, message integrity, and optional client authentication [85], [86]. JSSE ensures the secure passage of data between a client and a server running any application protocol, including HTTP, Telnet, or FTP, over Transmission Control Protocol over Internet Protocol (TCP/IP). By abstracting the complex underlying security algorithms and "handshaking" mechanisms, JSSE minimises the risk of creating subtle, but dangerous security vulnerabilities. In addition, it simplifies application development by serving as a building block, which developers can integrate directly into their applications [85].

3.7 Database Security

Bankers would be considered negligent if they locked a bank's outer doors and left the vault's doors open at night. Likewise, it doesn't make sense for an organisation to lock down the network and leave databases vulnerable [87].

Database systems are the primary data management technology used today by organisations for both day-to-day operations and decision-making. Damage and misuse of data held in database systems could have disastrous consequences on the entire organisation [88]. It is essential that database systems are protected, not only from external threats, but also from insider threats. An organisations database management system should ensure the confidentiality, integrity and availability of data. In as ASP environment, each client is provided access to the ASP solution software on a one-to-many basis, therefore it is critical that the ASPs ensure each clients data is protected from their competitors. In addition, auditing should be used to ensure that database security policies, procedures, and safeguards are working as intended.

3.7.1 Oracle

As databases contain extremely sensitive information, restriction of access through the enforcement of strong authentication is one of first lines of defence. Oracle provides strong authentication solutions leveraging on existing security frameworks such as Kerberos and Public Key Cryptography [89], [90]. In addition, Oracle provides several features to ensure data integrity whether in the case of system failure, human error, or malicious attacks. These features include redo log files, rollback segments, and LogMiner [91]. Oracle also supports the protection of selected data via encryption within the database. Although encryption is not a substitute for effective access control, one can obtain an additional measure of security by selectively encrypting sensitive data before it is stored in the database [90].

A critical aspect of any security policy is maintaining a record of system activity to ensure that users are held accountable for their actions. Oracle provides three standard types of auditing: Structured Query Language (SQL) statement-level, privilege-level and object-level auditing. Audit records can be written to the standard Oracle audit table, to an operating system audit trail (dependent on operating system used), or to an external file [92].

Based on industry standard best practices Sinha [89] provides the following guidelines with regard to the configuration of Oracle9i in order to maximise its security features:

- Install only what is required.
- Lock and expire default user accounts.
- Change default user passwords.
- Enable data dictionary protection.
- Practice principle of least privilege.
- Enforce access controls effectively.
- Restrict network access.
- Apply all security patches and workarounds.

3.8 Security Standards

The Open Standards Interconnect (OSI) security model 7498-2 and International Standards Organisation (ISO) 17799 security standard are both relevant to this examination of ASP security. OSI 7498-2 deals principally with network security, whereas ISO 17799 is concerned predominately with information security.

3.8.1 OSI Security Model 7498-2

ISO who are credited with the development of the seven layer OSI network model 7498-1, are also responsible for the development of the OSI security model 7498-2. The OSI security model is also composed of seven layers (outlined in Figure 3.14) and provides a high level view of network security. OSI 7498-2 defines a set of security services based on generally agreed objectives and identifies the architectural levels at which they may be provided [93][94].

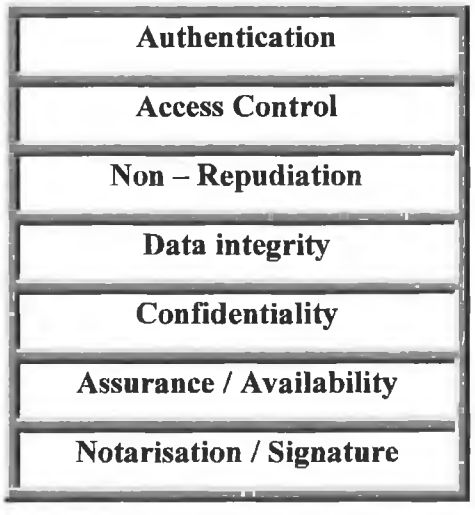


Figure 3.14 – OSI Security Model 7498-2

3.8.1.1 Authentication

Authentication provides a means of verification of claimed identity at a point in time [95]. Username/ID and password combinations are a basic means of authentication. However, weak passwords may still result in unauthorised access. In order to protect against unauthorised access in the form of brute-force attacks, it is recommended that passwords be eight to ten characters and be composed of a combination of alphanumeric, punctuation symbols and upper and lower case [96], [97], [98]. Digital signatures are recommended when a higher degree of system security is required.

3.8.1.2 Access Control

Access control, is a means of authorisation, the process of granting individuals access to system objects, based on their identity. Stergiou [99] defines access control as the restricted admission to network services and resources for the users holding the appropriate privileges. Access control lists are used to specify the users and corresponding access rights e.g. read, write, execute etc., associated with individual system objects. Alternatively, an access control list can be composed of a set of roles and corresponding access rights for system objects, with individuals assigned to relevant roles. Access to the ASP system and systems resources in particular should be restricted to an absolute minimum.

3.8.1.3 Non – Repudiation

Non-repudiation is the concept of ensuring that a transaction cannot later be denied by one of the parties involved i.e. the sender or recipient. Non-repudiation is analogous to signing a letter and sending recorded delivery [95]. It acts as a counter measure against fraudulent claims, denial of contracts and non-acceptance of debt liability. Digital

signatures are a means of assuring non-repudiation, which is a legal requirement of many e-business solutions.

3.8.1.4 Data Integrity

Data integrity is the process of ensuring the consistency and validity of both data stored locally and data transferred over the network. It encompasses the protection of both the user and signalling data from either accidental or malicious modifications [99]. An efficient security policy should ensure the integrity of data by defining effective access control lists and appropriate means of encryption. The use of encryption technologies such as symmetric key encryption and public key encryption ensures the integrity of data transferred over the Internet. Encryption is a form of cryptography with data being encoded prior to being sent over the network.

3.8.1.5 Confidentiality

Confidentiality is the process of protecting the privacy of both data stored locally and data transferred over the network. When data is exchanged over the network, both the connection and data must be protected against unauthorised access and disclosure [99]. An efficient security policy should ensure the confidentiality of data by limiting local access to authorised personnel and adapting the aforementioned encryption techniques to protect the privacy of data transferred over the network.

3.8.1.6 Assurance / Availability

Assurance/availability is the process of ensuring the optimal accessibility and usability of systems and the data stored therein [93], [94]. It is imperative that a breach of security does not hinder system availability. Two different modes of protection are available:

prevention; and detection and recovery using backup facilities [94]. A disaster recovery plan is essential to an optimal speed of recovery in the event a security breach, thus limiting its overall impact.

3.8.1.7 Notarisation / Signature

Notarisation is the process where by a trusted third party acts as a legal representative to guarantee the data integrity, peer authentication and non-repudiation services to the users [99]. Notarisation certifies the validity of both digital data and digital signatures through the use of Public Key Infrastructure (PKI). PKI is the allocation of digital signatures, generally in the form of digital certificates, by trusted third parties known as certification authorities. Notarisation provides a means of assurance of the accuracy of data content, origin, time and delivery.

3.8.2 ISO 17799

ISO 17799 is composed of a comprehensive set of controls, designed by the ISO, outlining best practices in information security and is composed of ten sections. Together, the ten sections provide guidelines to organisations that strive towards optimal information security. Figure 3.15, which outlines ISO 17799, has been adapted from [100], [101].

Figure 3.15 – ISO 17799

3.8.2.1 Security Policy

A security policy provides guidelines to management with regard to the documentation and the implementation a set of rules to ensure the optimal security of information stored locally and transferred over the network [100], [101]. Organisations have the option of constructing a new security policy, altering an existing security policy to meet the requirements outlined in ISO 17799, or alternatively they may purchase a security policy, which has been written by a third party to comply with the requirements of the standard. This activity involves a thorough understanding of the organisation's business goals and its dependence on information security [102].

3.8.2.2 System Access Control

System access control is a means of protecting information by controlling access to information systems. It is achieved through the use of an access control policy, which provides a means of authorisation by granting individuals access to system objects and functionality, based on their identity [100]. The detection of unauthorised activities is achieved through the adoption of monitoring techniques.

3.8.2.3 Computer and Operations Management

Computer and operations management is responsible for ensuring the secure operation of information processing facilities along with the integrity of system software and information, thus preventing the loss, modification or misuse of information stored locally or exchanged between organisations [101]. An organisation should document a set of procedures for day-to-day operations and, in addition, the procedures to be used in the event of a security disaster.

3.8.2.4 System Development and Maintenance

System development and maintenance is responsible for maintaining the security of system software and data by ensuring that security is built into information systems. Systems should be designed and implemented with a view to ensuring the privacy, integrity and validity of information. Any changes to operating system software packages should be strictly controlled [102].

3.8.2.5 Physical and Environmental Security

Physical and environmental security is responsible for the prevention of unauthorised access to computer facilities and the protection of information systems against theft, tampering or damage [100]. An organisation should devise a physical security policy, which controls access to the facility and the computer system.

3.8.2.6 Compliance

Compliance is concerned with avoiding breaches of any criminal or civil law, statutory, regulatory or contractual obligations and of any security requirements and ensuring compliance of the organisation with internal security policies and procedures [101].

3.8.2.7 Personnel Security

Personnel security involves, reducing the risk of human error, theft or fraud by providing appropriate training with regard to security threats and risks, and organisation policies and procedures [100], [101]. In addition, background checks of prospective employees should be carried out, and confidentiality or non-disclosure agreements and employee contracts should be used to protect the privacy of information.

3.8.2.8 Security Organisation

Security organisation involves the overseeing of information security across the organisation and maintaining the security of information when the responsibility for information processing has been outsourced to another organisation [100]. A central body should be responsible for ensuring the accuracy of security policies, monitoring

security activity and for the continuous improvement of information security throughout the organisation.

3.8.2.9 Asset Classification and Control

Asset classification and control is responsible for maintaining the appropriate protection of both corporate assets and information assets [100]. All assets within the organisation should be analysed and classified according to their sensitivity and criticality to ensure the appropriate access procedures are applied [102].

3.8.2.10 Business Continuity Management

Business continuity management counteracts interruptions to business activities and to critical business processes from the effects of major failures or disasters [100]. A business continuity process should identify security risks and threats, analyse the likelihood of the organisation being exposed to these threats, analyse the effects to the business if exposed to these threats and develop and test a business continuity plan to minimise impact.

3.9 Chapter Summary

Security in the context of Application Service Provision (ASP) is multifaceted, requiring an analysis of the security models across all tiers of the service architecture. However, the discrete security components must not be analysed in isolation, but should form part of a holistic security model.

This chapter provided an overview of the ASP model by drawing comparisons with traditional technology models, such as time-sharing and outsourcing. It examined ASP security in general and furthered this analysis by reviewing the application, middleware and data layers of the service architecture. This was achieved through the analysis of application, middleware, platform, database and network security technologies.

Finally it examined security policies, disaster recovery plans, service level agreements and security standards in order to obtain a complete perspective of security in an Application Service Provision (ASP) environment. It is hoped that this literature review illustrates the importance of the privacy, security and integrity of data across all tiers of the service architecture.

Chapter IV : Case Study

4.1 Introduction

This chapter describes aspects of security with regard to the Application Service Provision (ASP) billing prototype case study, formally known as Billing4Rent. The Billing4Rent solution enables tier 3 and 4 network operators, content providers, service aggregators and other genres of service providers to access state-of-the-art billing functionality on a subscription or rental basis. The Billing4Rent solution was implemented as part of a joint venture between The Telecommunications Software & Systems Group (TSSG) at the Waterford Institute of Technology and the Informatics Research Group at the Galway-Mayo Institute of Technology.

The Billing4Rent solution prototype is subdivided into three interconnected applications: the Billing4Rent Ltd web application, the billing solution and an administration tool. In addition to providing information about the ASP service, the web application provides a means of accessing both the billing solution and the administration tool. The billing solution provides Billing4Rent clients with a means to configure and maintain customer and product details and to generate billing information in the form of invoices. The administrator tool provides Billing4Rent employees with the ability to configure and maintain Billing4Rent client details and to monitor both the security and the performance of the billing solution.

4.2 Architecture

The ASP Billing4Rent solution prototype is composed of a number of JavaServer Pages (JSP), Java servlets, enterprise beans and libraries, interfacing with an Oracle database. The solution architecture was designed based on the Model-View-Controller (MVC)

design pattern and the solution was deployed in a Java Platform Enterprise Edition (J2EE) environment. The MVC architecture separates an application into three distinct components so that modifications to one component can be made with minimal impact to the others. The model encapsulates the application data, the view is responsible for displaying the data stored in the model and the controller facilitates the exchange of data between the model and the view.

4.2.1 Security Architecture

Figure 4.1 details the network architecture of the Billing4Rent ASP prototype. A demilitarised zone (DMZ) segregates servers requiring external access from other machines on the network. A DMZ is defined as a network that sits between the trusted internal network and the untrusted external network [105]. External access to Hyper Text Transport Protocol (HTTP), Hyper Text Transport Protocol/Secure Sockets Layer (HTTPS), File Transfer Protocol (FTP), File Transfer Protocol /Secure Sockets Layer (FTPS) and Simple Mail Transfer Protocol (SMTP) services on the DMZ network is permitted through the outer firewall. Access to machines on the internal network is only allowed from machines in the DMZ through the inner firewall. All Billing4Rent hardware and communications equipment is located in a lockable room within the main monitored, alarmed and secured campus building. The Billing4Rent ASP infrastructure is further secured through the use of security cables, padlocks and other such devices.

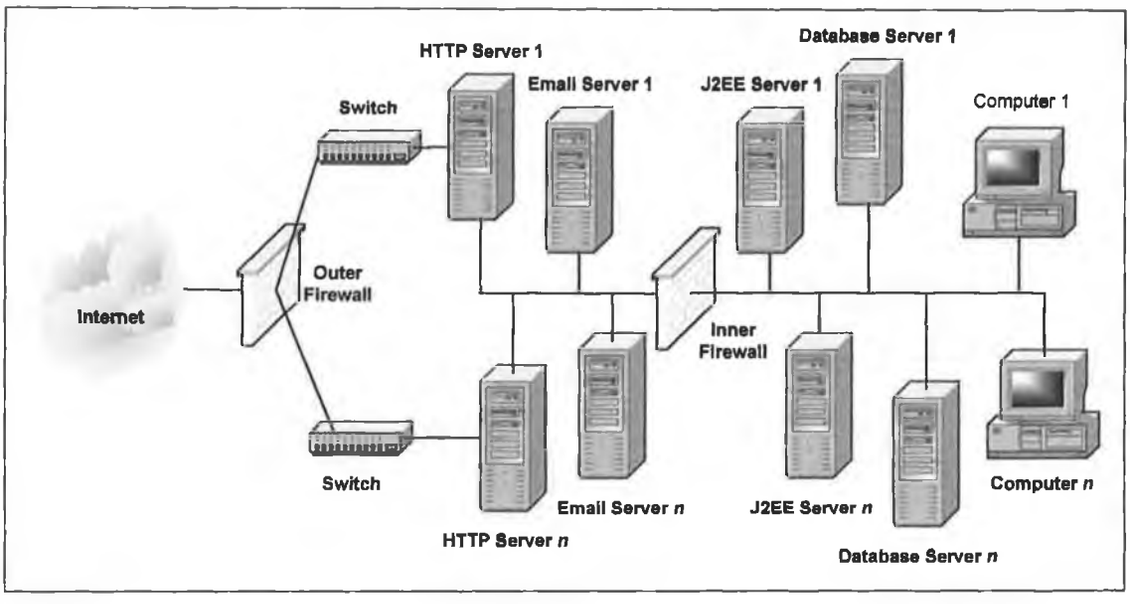


Figure 4.1 – Biling4Rent network architecture

As an additional measure of security, the network administrator ensures all machines on the Billing4Rent network are protected against network attacks. All unused ports and daemons are shutdown. Appropriate service packs and patches are applied, as soon as they are made available. Finally, anti-virus and anti-spyware software are installed and are updated periodically by the network administrator.

4.2.2 Solution Architecture

Figure 4.2 depicts the Billing4Rent ASP solution architecture, which was deployed in a J2EE environment. Apache Tomcat accepts both HTTP and HTTPS requests. Tomcat uses a set of Java packages known as, Java Secure Socket Extension (JSSE), to enable secure Internet communications via HTTPS. The Billing4Rent application enforces strict access control to both the billing solution and the administration tool through the use of the dynamic Java Authentication and Authorisation Service (JAAS) framework. The Billing4Rent application uses the *java.SQL.DriverManager* class, to open a Java Database Connectivity (JDBC) connection to the tier 3 database. Jakarta FileUpload,

JAMon and Tag libraries are used to support functionality required by the billing solution and the administration tool.

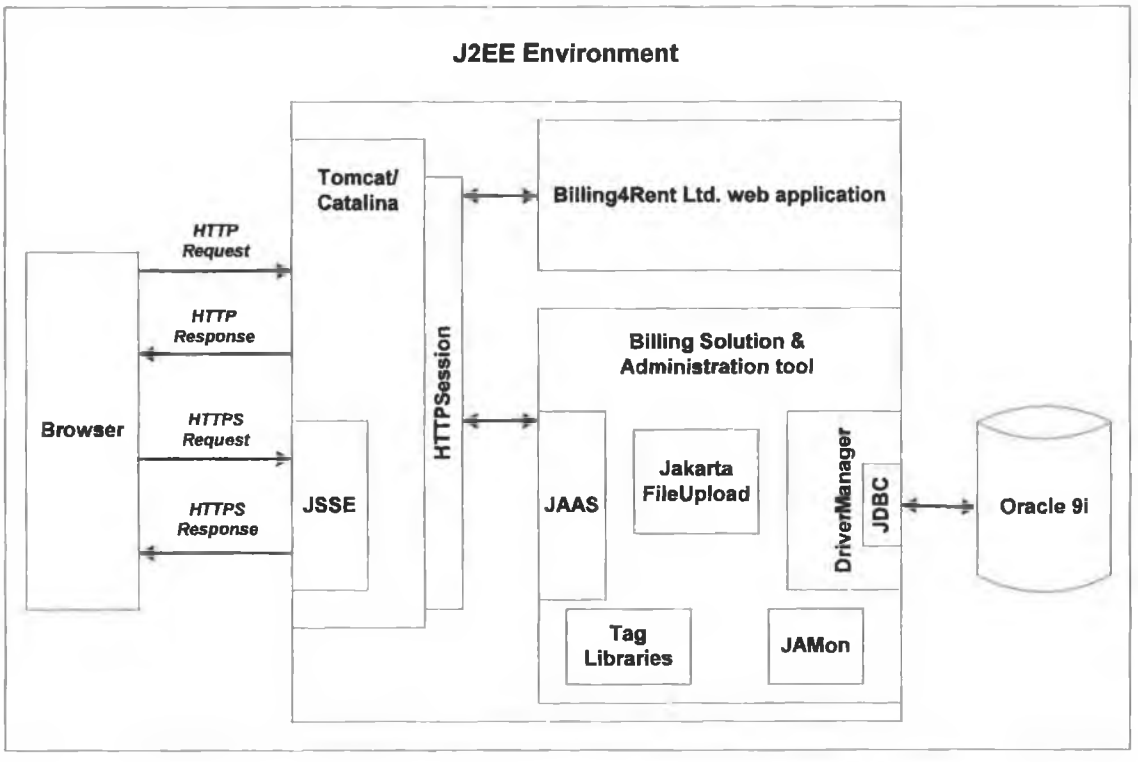


Figure 4.2 – Biling4Rent solution architecture

4.3 Authentication and Authorisation

Authentication is defined as the verification of identity [81] i.e. you are who you say you are. Authorisation is the process of granting or denying access to system resources based on identity. It is essential that access to the billing solution, the administration tool and associated functionality is restricted to authorised personnel. Due to the unpredictable nature of the Billing4Rent project, the project team deemed the dynamic Java Authentication and Authorisation Service (JAAS) framework the most suitable means of performing authentication and authorisation. The JAAS framework simplifies Java security development by adding a layer of abstraction between the Billing4Rent solution

and the underlying authentication and authorisation mechanisms. JAAS achieves implementation independence through an extensible framework of pluggable abstract classes and interfaces to which specific implementations can be developed. The billing solution and administration tool servlets and JSP pages have been designed to ensure that the user has been authenticated and has the required permissions prior to rendering the page on their browser. If an unauthenticated user tries to access a page, by typing the path directly into address bar or using the back button, they will automatically be redirected to the login page. If an unauthorised user tries to access a page, they will be presented with a message indicating they do not have the privileges required to access the page.

4.3.1 Authentication

JAAS authentication is used in order to determine who is currently executing Java code, regardless of whether the code is running as an application, an applet, a bean or a servlet [80] [81]. Figure 4.3, which provides a high-level overview of the JAAS architecture has been adapted from [103]. The application layer code instantiates both the *CallbackHandler* and the *LoginContext* objects. The *CallbackHandler* interacts with the application to retrieve the data required to perform the authentication. The *LoginContext* subsequently communicates with one or more, dynamically configured login modules, which handle the actual authentication using the appropriate security infrastructure. Sun Microsystems supplies a number of *LoginModule* implementations in the com.sun.security.auth package i.e. *JndiLoginModule*, *KeyStoreLoginModule*, *Krb5LoginModule*, *NTLoginModule* and *UnixLoginModule*. Alternatively, it is possible to develop custom login modules that implements the *LoginModule* interface e.g. a database login module.

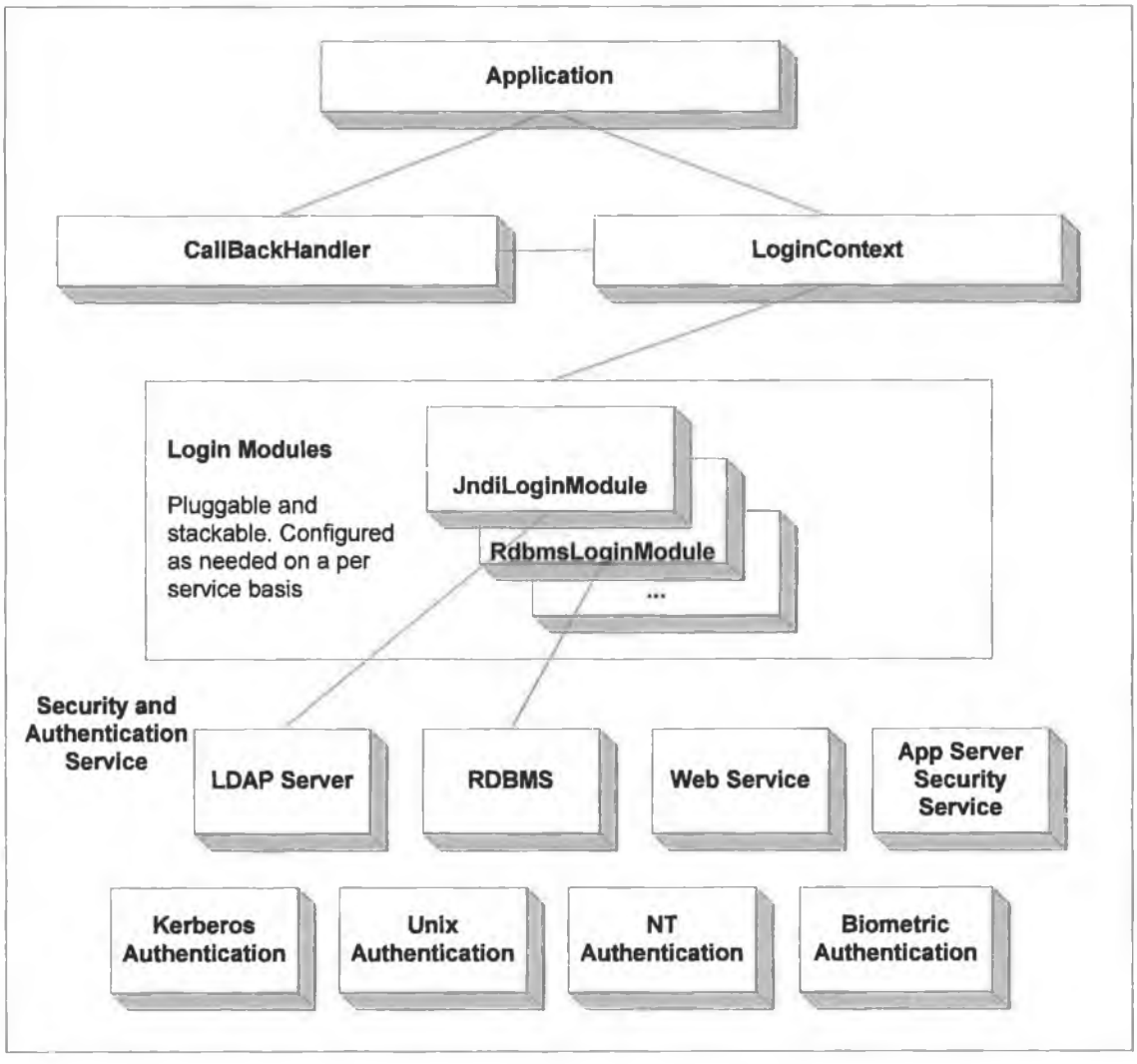


Figure 4.3 – JAAS high-level architecture

The following steps are required to implement authentication using the JAAS framework:

- If required, implement custom *LoginModule* and *CallbackHandler* classes.
- If an implementation exists, instantiate the appropriate *CallbackHandler* object.
- Instantiate a *LoginContext* object passing in the *CallbackHandler* object, where applicable, and the name of the index to the JAAS configuration file, in order to determine which *LoginModule* object(s) should be used. The *LoginContext* is responsible for instantiating the appropriate *LoginModule* object(s).

- Perform authentication by calling the *LoginContext* object's login() method. The *LoginContext* subsequently calls each *LoginModule* object's login() method. Each login() method performs the authentication or enlists the help of a *CallbackHandler*.
- If the login is successful, a new *Subject* object is instantiated and populated with authentication details such as principals and credentials.

The Billing4Rent project team chose to authenticate user login details based on data stored in an Oracle 9i database. The team implemented two login modules: a *RdbmsClientLoginModule* and a *RdbmsAdminLoginModule* in the com.B4R.library.security.JAAS package, to authenticate Billing4Rent clients and administrators respectively. Figure 4.4 provides a high level overview of Billing4Rent client authentication, constructed using Unified Modeling Language (UML) sequence diagram components.

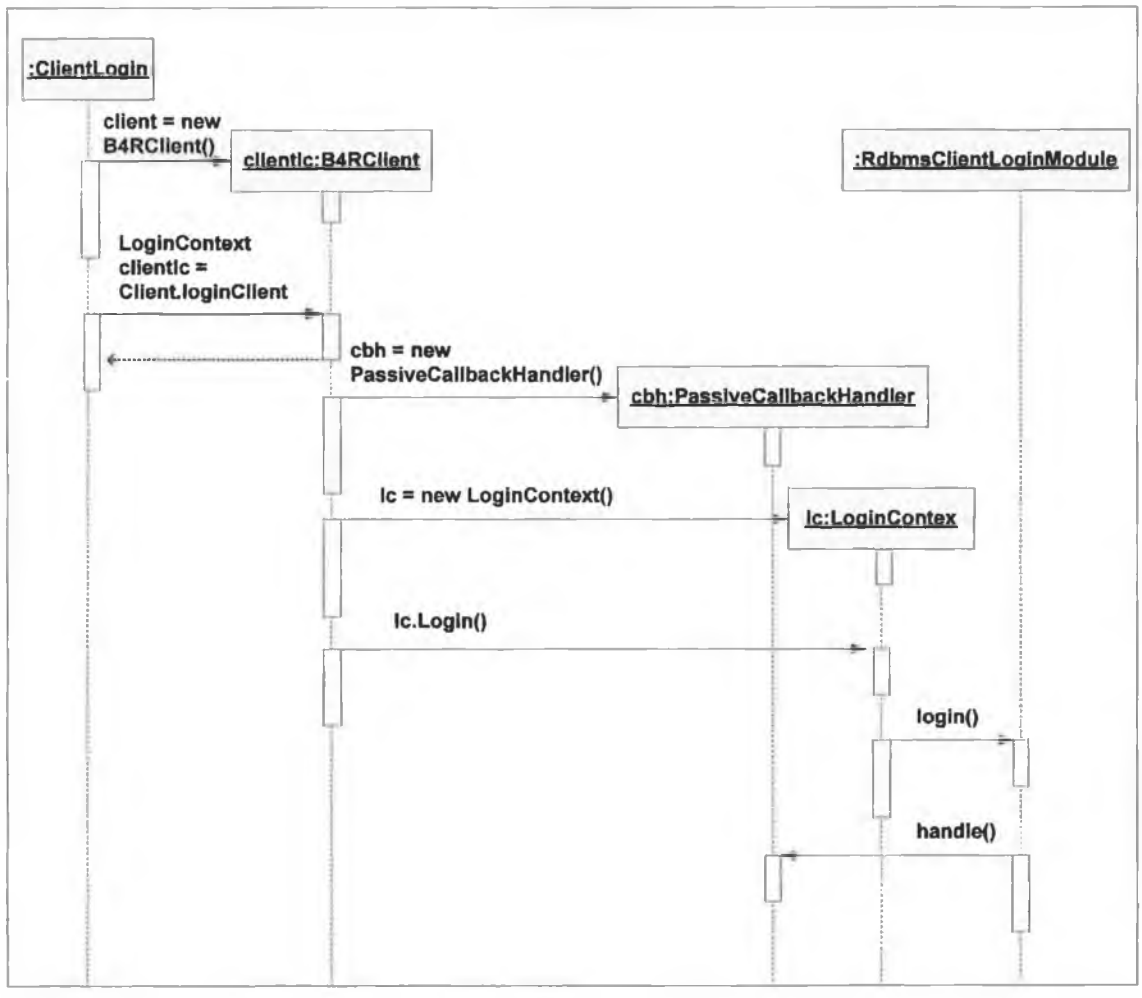


Figure 4.4 – Authentication Sequence Diagram

Figure 4.5 contains a snippet of code extracted from the *B4RClient* business class. The *B4RClient* object is envoked from the *ClientLogin* servlet, when a client attempts to login to the billing solution via the Billing4Rent client login JSP page. The clientID and password, input by the client, and the login time, generated by the system at runtime, are used to instantiate the *passiveCallbackHandler*. On instantiation the *LoginContext* searches for the CLIENTORACLE configuration in a JAAS configuration file, named jaas.config (Figure 4.6), to determine which login module(s) to load. The JAAS configuration file name and path are configured in the Java.security file (Figure 4.7). The *B4RClient* calls the *LoginContext* object's login() method. The *LoginContext* subsequently calls the *RdbmsClientLoginModule* object's login() method. The login()

method performs the authentication by enlisting the help of the *PassiveCallbackHandler* (Figure 4.8, Figure 4.9). If the login is successful, a new *Subject* object is instantiated and populated with authentication details such as the username and password (Figure 4.10), otherwise an exception is thrown.

```
try {
        // Use the clientid/password to initialise the callback handler
        PassiveCallbackHandler cbh = new PassiveCallbackHandler(clientId,
        password, loginTime);

        // Create a new loginContext
        LoginContext lc = new LoginContext("CLIENTORACLE", cbh);

        // Attempt authentication
        lc.login();
} catch (LoginException ex) {

        throw new LoginException(new StringBuffer("[B4RClient] ")
                .append(ex.getMessage())
                .toString());
} catch (Exception e) {

        throw new LoginException(new StringBuffer("[B4RClient] ")
                .append(e.getMessage())
                .toString());
}
```

Figure 4.5 – Extract from the B4RClient business object

4.3.1.1 Login configuration file

The login configuration file contains one or more <LoginModule> specification entries. Figure 4.6, an extract from the jass.config login configuration file, contains an Oracle (CLIENTORACLE) <LoginModule> specification entry. The structure of each entry in the login configuration file takes the following format:

```
<name> {
    <LoginModule> <flag> <options>;
    <LoginModule> <flag> <options>;
    ...
};
```

The <name> property is an index into the JAAS configuration and is used by the *LoginContext* to retrieve authentication details. The <LoginModule> property is the full path of a class that implements the *LoginModule* interface. The <flag> property indicates whether success of the *LoginModule* is required, requisite, sufficient or optional. If only one *LoginModule* is configured, the <flag> property must be set to required. The <options> property specifies a list of name-value-pairs, which are passed to the corresponding *LoginModule*. Each login configuration file entry facilitates the use of multiple entries, e.g. optional additional <LoginModules>, <flag> and <option> properties. The login configuration file can be specified either on the command line or in the Java security properties file. As the ASP billing solution was designed around a web interface, the Billing4Rent project team specified the login configuration file in the Java security properties file. Figure 4.6 defines the *RdbmsClientLoginModule* in the com.B4R.library.security.JAAS package and a number of configuration options. The debug flag, oracle database URL, oracle driver, oracle username and oracle password properties are all defined as name-value-pairs required by the *RdbmsClientLoginModule* to perform authentication.

```
CLIENTORACLE {
com.B4R.library.security.JAAS.RdbmsClientLoginModule required debug="false"
url="jdbc:oracle:thin:@localhost:1521:B4R"
user="SYSTEM"
pass="research"
driver="oracle.jdbc.driver.OracleDriver";
};
```

Figure 4.6 – Extract from jaas.config

4.3.1.2 Java security file

The Java security manager uses the master security properties file, Java.security, to enforce security based on what code is running and, more importantly, who is running the code. Figure 4.7, an extract from the java.security file, specifies the name and the location of the default login configuration file. It is worth noting that it is possible to specify several login configuration files by incrementing the numeric value of the property e.g. login.config.url.1, login.config.url.2, login.config.url.3 etc. If more that one login configuration file is specified, the security manager concatenates the files into a single login configuration file.

```
#
# Default login configuration file
#
login.config.url.1=file:${java.home}/lib/security/jaas.config
```

Figure 4.7 – Extract from java.security

4.3.1.3 PassiveCallbackHandler

Callbacks provide the JAAS framework with the ability to interact with the calling application to retrieve specific authentication data, such as usernames and passwords, or to display certain information, such as error and warning messages. Figure 4.8 contains a snippet of code extracted from the *RdbmsClientLoginModule* class library, which initialises a Callback array with a *NameCallback*, a *PasswordCallback* and a *TextInputCallback*, and passes this array as a parameter to the handle() method of the *CallbackHandler*.

```
// Setup default callback handlers.
Callback[] callbacks = new Callback[] {
      new NameCallback("Username: "),
      new PasswordCallback("Password: ", false),
      new TextInputCallback("LoginTime: ")
};
callbackHandler.handle(callbacks);

String username = ((NameCallback)callbacks[0]).getName();
String password = new String(((PasswordCallback)callbacks[1]).getPassword());
      ((PasswordCallback)callbacks[1]).clearPassword();
String loginTime = ((TextInputCallback)callbacks[2]).getText();
```

Figure 4.8 – Extract from the RdbmsClientLoginModule class library

From a software development perspective, there are various means of communicating with a user, e.g. command-line, Graphical User Interface (GUI), web interface. In order to provide independence from the different means of user interaction, the *LoginModule* invokes a *CallbackHandler* in the javax.security.auth.callback package to perform the user interaction and obtain the requested information. As the Billing4Rent solution was built around a web interface, the Billing4Rent project team implemented a *PassiveCallbackHandler*. Figure 4.9 contains a snippet of code extracted from the

PassiveCallbackHandler class library. The constructor takes a username/userID and password, retrieved from the user via the JSP login page, and a system generated login time, so its handle() method does not have to prompt the user for input. The *PassiveCallbackHandlers* handle() method uses the username/userID, password and login time, that were supplied to the constructor, to popluate the *NameCallback*, *PasswordCallback* and the *TextInputCallback* respectively.

```
/**
* Assigns the username and password input via login page
* to local variables.
*
* @param                user the username to be authenticated.
* @param                pass the password to be authenticated.
* @param                loginTime System generated login time
*/
public PassiveCallbackHandler(String user, String pass, String loginTime) {
      this.username = user;
      this.password = pass.toCharArray();
      this.loginTime = loginTime;
}

/**
* Handles the specified set of Callbacks. Uses the
* username and password that were supplied to our
* constructor to popluate the NameCallback and
* PasswordCallback.
*
* @param   callbacks array of callbacks to handle.
* @throws  IOException if an input or output error occurs.
* @throws  UnsupportedCallbackException if the callback
*          is not an instance of NameCallback or PasswordCallback.
```

```
*/
public void handle(Callback[] callbacks)
   throws java.io.IOException, UnsupportedCallbackException {
      for (int i = 0; i < callbacks.length; i++) {
            // Check if this is a NameCallback
            if (callbacks[i] instanceof NameCallback) {
                  ((NameCallback)callbacks[i]).setName(username);
            // Check if this is a Password Callback
            } else if (callbacks[i] instanceof PasswordCallback) {
                  ((PasswordCallback)callbacks[i]).setPassword(password);
            // Check if this is a Password Callback
            } else if (callbacks[i] instanceof TextInputCallback) {
                  ((TextInputCallback)callbacks[i]).setText(loginTime);
            // Otherwise throw an unsupported callback exception
            } else {
                  throw new UnsupportedCallbackException(
                     callbacks[i],
                     "[PassiveCallbackHandler] Callback class not supported");
            }
      }
}
```

Figure 4.9 – Extract from the PassiveCallbackHandler class library

4.3.1.4 Subject, principals and credentials

If the login is successful, a new *Subject* object is instantiated and populated with authentication details such as the username and password. A *Subject* object represents a group of related information for a single entity such as a person or an organisation. The *Subject* is composed of principals, one or more identities, and corresponding credentials, one or more security-related attributes, e.g. passwords and cryptographic keys.

Credentials are classed as either private containing sensitive information, or public containing information intended to be shared. Figure 4.10 contains a snippet of code extracted from the *RdbmsClientLoginModule* class library, which stores the principals and credentials in temporary variables. Two *RdbmsAdminPrincipal* objects are created for the client, based on the client's full name and role identities, both of which are extracted from the database. In addition, a *RdbmsPrivateCredential* object and a *RdbmsPublicCredential* object are created to store the user's password and public key certificate respectively. Both the public and the private credentials are also extracted from the database. The *RdbmsAdminPrincipal*, *RdbmsPublicCredential* and *RdbmsPrivateCredential* objects are stored temporarily in private vectors.

```
p = new RdbmsAdminPrincipal(new StringBuffer(dbFname)
      .append(" ")
      .append(dbLname)
      .toString());
this.tempPrincipals.add(p);

p = new RdbmsAdminPrincipal(dbRole);
this.tempPrincipals.add(p);

publicCredential = new RdbmsPublicCredential();
publicCredential.setProperty("credential", dbCredential);
this.tempPublicCredentials.add(publicCredential );

privateCredential = new RdbmsPrivateCredential();
privateCredential.setProperty("password", dbPassword);
this.tempPrivateCredentials.add(privateCredential);
```

Figure 4.10 – Extract from the RdbmsClientLoginModule class library

Finally, the data stored in the temporary vectors is added to the *Subject* objects principals and credentials members (Figure 4.11).

```
subject.getPrincipals().addAll(tempPrincipals);
subject.getPublicCredentials().addAll(tempPublicCredentials);
subject.getPrivateCredentials().addAll(tempPrivateCredentials);
```

Figure 4.11 – Extract from the RdbmsClientLoginModule class library

4.3.2 Authorisation

JAAS authorisation enables system administrators to ensure that users have the appropriate permissions to perform required actions and access requested resources. JAAS extends the existing Java security architecture that uses a security policy to specify what access rights are granted to executing code. JAAS enables the administrator to grant access rights based on who is running the code, in addition to what code is running [80], [81]. After the client has been authenticated, a *Subject* object is instantiated and populated with authentication details such as principals and credentials. A *Subject* is comprised of one or more principals, where each principal represents an identity for that user or entity and a set of credentials. A credential is taken to represent any object used to perform authorisation, e.g. a certificate, password, or kerberos ticket. The Java security manager uses a policy file to associate authenticated principals with permissions. If a principal associated with a particular *Subject* object has the required permission, then the action is permitted, otherwise the action is denied and a *SecurityException* is thrown.

4.3.2.1 Policy file

The security policy, used by tomcat and implemented by the Java security manager, is configured in the $CATALINA_HOME/conf/catalina.policy file. The catalina.policy file

is used instead of the java.policy file, located in the Java development kit (JDK) systems directory, for web applications. Figure 4.12 contains an extract from the catalina.policy file used by the Billing4Rent case study. The entries in the catalina.policy file adhere to the following format:

```
grant <signedBy "signer_names">, <codeBase "URL">, <principal "principal name"> {
        permission permission_class_name "target_name", <"action">,
        <signedBy "signer_names">;
        permission permission_class_name "target_name", <"action">,
        <signedBy "signer_names">;
        ...
};
```

Each 'grant' entry contains one or more permission sub entries. The <signedBy>, <codeBase> and <principal> properties are optional. The <permission> property is composed of the full path of a *Permission* class, the actual permission specified by the target name, and where required, the action, e.g. type of file access permitted. The Java API provides several subclasses of the *java.security.permission* class. For example, *AllPermission*, *BasicPermission* and *FilePermission*. Alternatively, it is possible to extend the *java.security.permission* class to handle the type of authorisation required.

As the ASP billing solution was designed around a web interface, the Billing4Rent project team implemented a *java.security.permission* class, *URLPermission*, in order to permit or deny access to URLs. The *URLPermission* class, which resides in the com.B4R.library.security.JAAS package, extends the *java.security.permission* class. The *URLPermission* class is capable of validating both absolute paths and wild cards, i.e. paths that ends in '/*'.

```
grant  Principal com.B4R.library.security.JAAS.RdbmsAdminPrincipal "admin" {
  permission com.B4R.library.security.JAAS.URLPermission "/b4r/admin/*";
};
```

Figure 4.12 – Extract from catalina.policy

4.3.2.2 Privileged actions and the security manager

After the client has successfully logged into Billing4Rent, a *Subject* object is instantiated and populated with authentication details such as principals and credentials. For each subsequent page request, the application calls the *Subject* classes static doAsPrivileged() method, in order to determine if the client has the privileges required to access the resource. Figure 4.13 provides a high level overview of Billing4Rent client authorisation, constructed using Unified Modeling Language (UML) sequence diagram components.

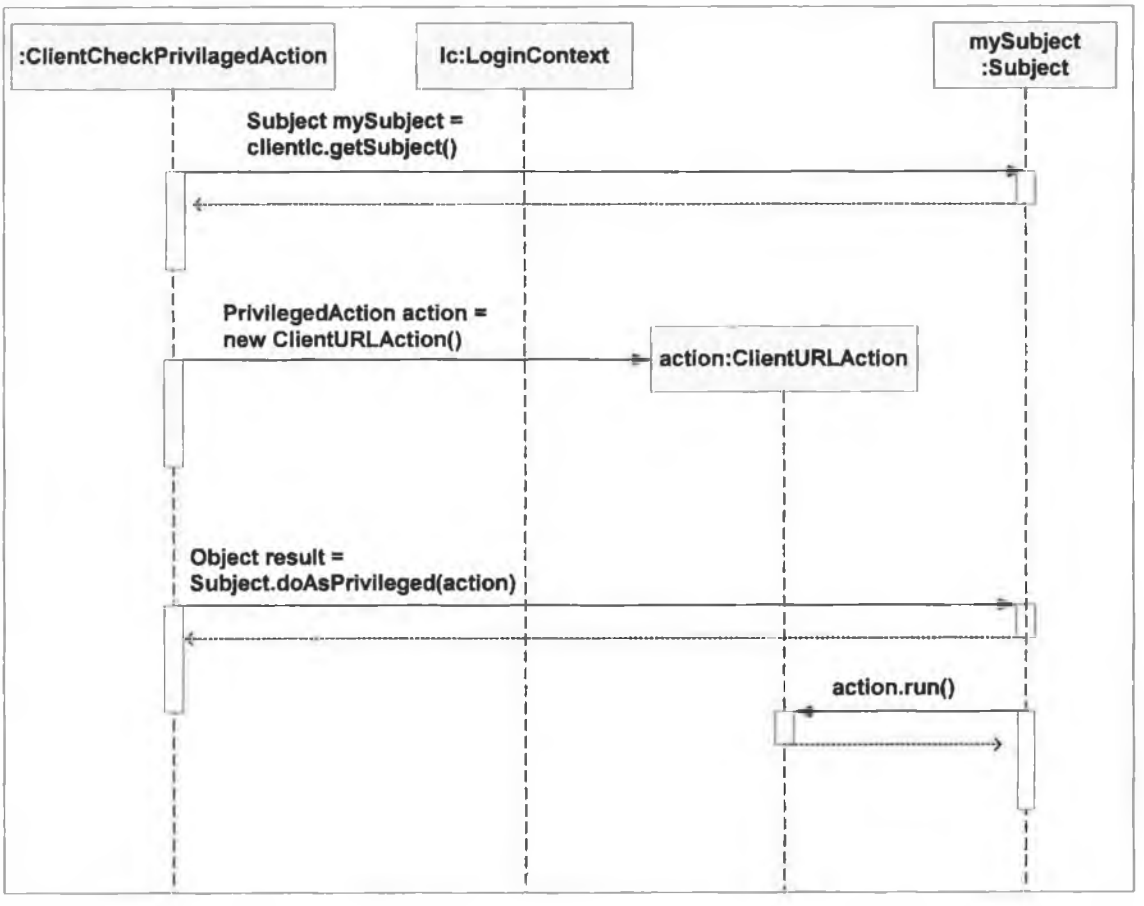


Figure 4.13 – Authorisation Sequence Diagram

Figure 4.14 contains a snippet of code extracted from the *ClientCheckPrivilagedAction* servlet, which retrieves a *Subject* object from a *LoginContext*, instantiates a *ClientURLAction* object and passes both as parameters to the *Subject* class' static doAsPrivileged() method. The doAsPrivileged() method invokes the run() method of the *ClientURLAction* object, which uses the Java security manager and the catalina.policy file to verify the client has the required permissions. Figure 4.4 provides a high level overview of client authentication, using Unified Modeling Language (UML) sequence diagram components.

```
// Retrieve the LoginContext from the session object
LoginContext clientlc = (LoginContext) session.getValue("clientlc");

// Retrieve LoginContext subject and store in subject object
Subject mySubject = clientlc.getSubject();

// Execute URLAction as an authenticated Subject
PrivilegedAction action = new ClientURLAction();
Object result = Subject.doAsPrivileged(mySubject, action, null);

// Convert result from object to boolean
boolean success = ((Boolean)result).booleanValue();
```

Figure 4.14 – Extract from the ClientCheckPrivilagedAction servlet

Figure 4.15 contains a snippet of code extracted from the *ClientURLAction* class library. *ClientURLAction* is an implementation of the PrivilegedAction interface, which determines if the user has the required permission to access the requested webpage. The *ClientURLAction* object's run() method retrieves a *SecurityManager* object if one exists or alternatively instantiates a new *SecurityManager* object. It instantiates a *URLPermission* object and passes it as a parameter to the *SystemManager* object's checkPermission() method. The *SystemManager* object's checkPermission() method in turn calls the *AccessController* object's checkPermission() method which determines whether access to the requested resource should be allowed or denied. This decision is based on the permission to be validated and the security policy currently in effect. If access is allowed, checkPermission() returns quietly, i.e. returns boolean true. Alternatively an exception is thrown. Finally, the *ClientURLAction* object's run() method returns true if the client has the required permissions. Otherwise it returns false. It is worth noting that the *URLPermission* class is capable of validating both absolute paths and wild cards, i.e. paths that ends in '/*'.

```
// If a security manager does not exist create a new security manager object
final SecurityManager sm;

if (System.getSecurityManager() == null)
      sm = new SecurityManager();
else
     sm = System.getSecurityManager();

// Create a new URLPermission object in order to permit/deny
// access to URL pages based on user permissions.
final Permission urlPerm = new URLPermission("/b4r/client/*");

// Throws a SecurityException if the requested access, specified
// by the given permission, is not permitted based on the security
// policy currently in effect. This method calls
// AccessController.checkPermission with the given permission.
sm.checkPermission(urlPerm);
```

Figure 4.15 – Extract from the ClientURLAction class library

4.4 Secure Login and navigation

In an ASP environment, both the security of the client login details and the security and the integrity of the client data transmitted over the network are imperative. In order to ensure the confidentiality and integrity of the solution data, the Billing4Rent project team designed the billing solution to use HTTPS. HTTPS is a secure version of the HTTP where requests and responses are encrypted using Secure Sockets Layer (SSL) or Transport Layer Security (TLS) prior to being sent over the network. In the Billing4Rent solution, the HTTPS protocol is used for the login to (Figure 4.16, Figure 4.17) and navigation of both the billing solution and the administration tool. It is worth noting that

as the Billing4Rent prototype was also designed to ensure optimal performance, access to general information that does not warrant security is permitted via HTTP.

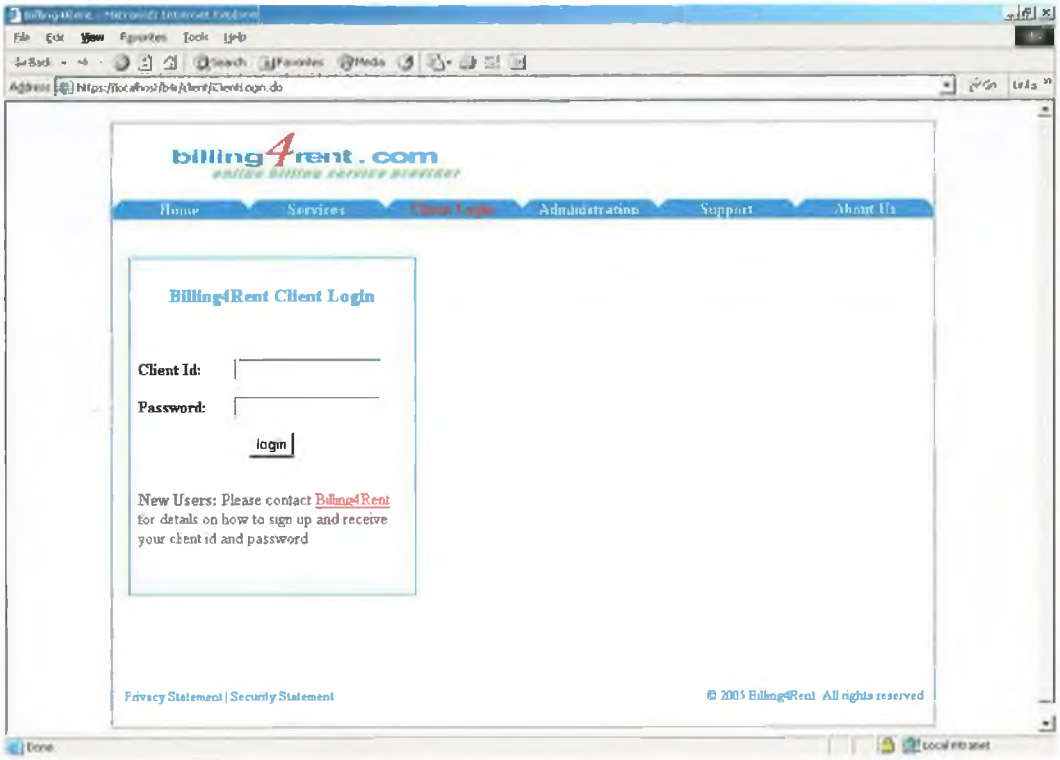


Figure 4.16 – Billing4Rent Client Login

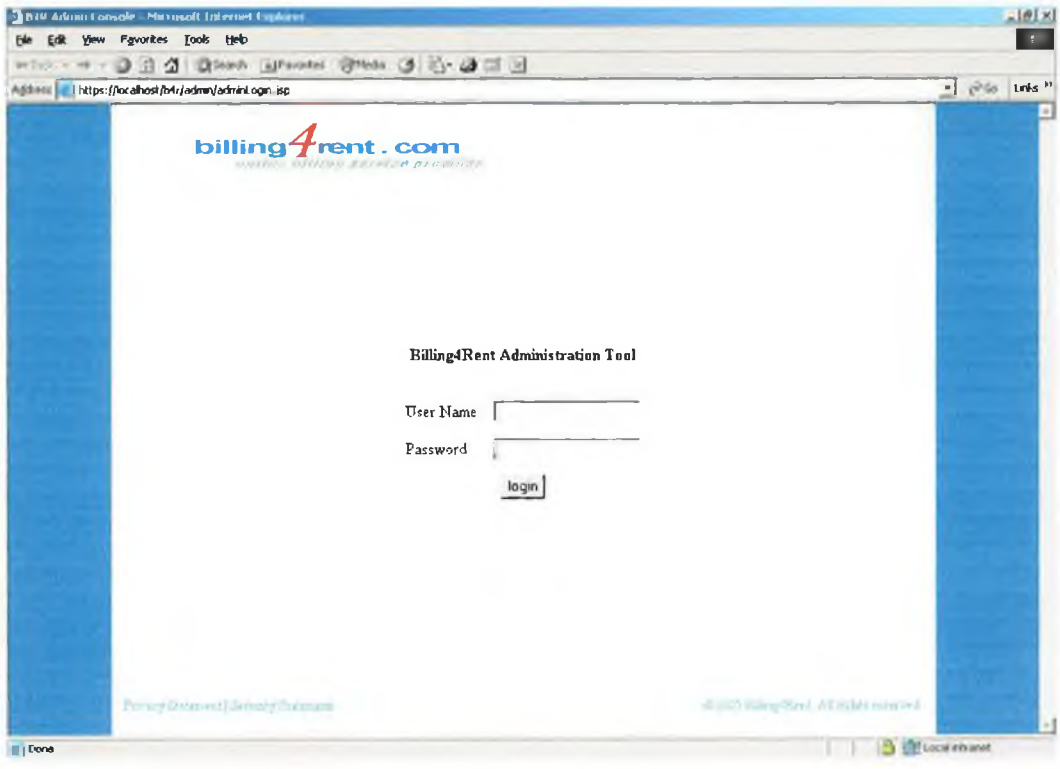


Figure 4.17 – Billing4Rent Client Login

The Billing4Rent solution was deployed in an Apache Tomcat environment, which uses a set of Java packages known as Java Secure Socket Extension (JSSE), to enable secure Internet communications. HTTPS is configured in Tomcat by creating a certificate keystore (Figure 4.18) and defining a SSL HTTP connector in the server.xml configuration file (Figure 4.19).

Keytool is Sun Microsystem's command-line, key and certificate management utility. It facilitates the administration of public/private key pairs and associated certificates for use in self-authentication or data integrity and authentication services, using digital signatures. A keytool is used to store the keys and certificates in a keystore - a special file type that can hold keys and certificates. The keystore is subsequently encrypted with a password for security.

The following steps are required to create a new certificate keystore:

- Ensure the JAVA_HOME environment variable is configured to point to the Java Development Kit (JDK).
- Open a command prompt and type the command in Figure 4.18. This command will create a new file, in the home directory of the user under which it is run, named '.keystore'. Alternative, it is possible to use the keystore parameter to specify the complete pathname to the keystore file.
- Enter the keystore password when prompted. The default password used by Tomcat is 'changeit', although it is possible to specify a custom password.
- Complete the general information with regard to this certificate, such as company, contact name etc.
- Finally, enter the key password, which is the password specific to this certificate.

```
%JAVA_HOME%\bin\keytool -genkey -alias tomcat -keyalg RSA
```

Figure 4.18 – Call Keytool from the command line

To complete the Tomcat HTTPS configuration, uncomment the SSL HTTP connector element in the server.xml configuration file and update the connector attributes. Change the default port number from 8443, the port on which Tomcat listens for secure TCP/IP connection, to 443 the default port for HTTPS communication. Figure 4.20, adapted from [104] details several attributes, which may need to be updated based on the options specified during creation of the keystore.

```
<!-- Define a SSL HTTP/1.1 Connector on port 443 -->
<Connector port="443" maxHttpHeaderSize="8192"
        maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
        enableLookups="false" disableUploadTimeout="true"
        acceptCount="100" scheme="https" secure="true"
        clientAuth="false" sslProtocol="TLS" />
```

Figure 4.19 – Extract from server.xml

Attribute	Details
ClassName	The fully qualified class name of the Java class that implements this socket factory. Do not change the default value.
ClientAuth	Set this value to true if you want Tomcat to require all SSL clients to present a client Certificate in order to use this socket.
KeystoreFile	Add this attribute if the keystore file you created is not in the default place that Tomcat expects (a file named .keystore in the user home directory under which Tomcat is running). You can specify an absolute pathname, or a relative pathname that is resolved against the $CATALINA_BASE environment variable.
KeystorePass	Add this element if you used a different keystore (and Certificate) password than the one Tomcat expects (changeit).
Protocol	The encryption/decryption protocol to be used on this socket. Do not change the default value.

Figure 4.20 – Connector properties adapted from [104]

As the Billing4Rent billing solution and administration tool both use HTTPS, the first time a user attempts to access a secured page on the Billing4Rent site, they will be presented with a dialog containing the details of the certificate and asked if they wish to proceed with the transaction (Figure 4.21). If the user chooses to proceed, the server

renders the appropriate login page on the browser. Otherwise the current page will continue to be displayed.

Figure 4.21 – Accept/reject certificate dialog

4.5 Secure Logout

After the user has logged out of a password-protected system, they should be redirected to the login page. Under no circumstances should a user be granted access to the system without re-logging into the system. In order to achieve this level of security, it is imperative that all pages of a password-protected site ensure the user has been authenticated prior to the page being rendered on the browser. Secondly, pages must not be cached on the client i.e. HTML documents should not be stored temporarily for ease of retrieval later.

4.5.1 Logout

When the user chooses to logout of either the billing solution or the administration tool, the appropriate *Subject* object is destroyed and the user's session is invalidated. Figure 4.22 contains a snippet of code extracted from the *ClientLogout* servlet, which retrieves the *LoginContext* object from the session and calls the *LoginContext* object's logout() method. This method in turn invokes the logout() method of the *RdbmsClientLoginModule* object (Figure 4.23). If logout is successful, the method returns quietly, i.e. returns boolean true. Alternatively an exception is thrown. Finally, the servlet removes both the *LoginContext* and the clientID from the *HttpSession* object, invalidates the session and unbinds any objects coupled to it.

```
// Retrieve the LoginContext object from the session object
LoginContext clientlc = (LoginContext) session.getValue("clientlc");

client = (String) session.getValue("clientId");

// Call the LoginContext logout method
clientlc.logout();

// Remove LoginContext session attribute
session.removeAttribute("clientlc");

session.removeAttribute(client);

// Invalidate the session
session.invalidate();
```

Figure 4.22 – Extract from the ClientLogout servlet

Figure 4.23 contains a snippet of code extracted from the *RdbmsClientLoginModule* class library, which uses an Iterator to remove all of the principals and the credentials from the *Subject* object.

```
// remove the principals the login module added
Iterator it = subject.getPrincipals(RdbmsAdminPrincipal.class).iterator();
while (it.hasNext()) {
        RdbmsAdminPrincipal p = (RdbmsAdminPrincipal)it.next();
        subject.getPrincipals().remove(p);
}

// remove the public credentials the login module added
it = subject.getPublicCredentials(RdbmsPublicCredential.class).iterator();
while (it.hasNext()) {
        RdbmsPublicCredential c = (RdbmsPublicCredential)it.next();
        subject.getPublicCredentials().remove(c);
}

// remove the private credentials the login module added
it = subject.getPrivateCredentials(RdbmsPrivateCredential.class).iterator();
while (it.hasNext()) {
        RdbmsPrivateCredential c = (RdbmsPrivateCredential)it.next();
        subject.getPrivateCredentials().remove(c);
}
```

Figure 4.23 – Extract from the RdbmsClientLoginModule class library

A primary requirement of the Billing4Rent solution is to ensure that sensitive data remains confidential. After the user has logged out of either the billing solution or the administration tool and the session has been invalidated, the user is redirected to the appropriate login page. The billing solution and administration tool servlets and JSP pages have been designed to ensure that the user has been authenticated prior to rendering

the page on the user's browser. If an unauthenticated user tries to access a page by typing the path directly into address bar, they will automatically be redirected to the login page. However, further action was required to ensure that the browsers back button could not be used in order to access cached pages.

4.5.2 Disable Back Button

When the Back button is clicked, by default the browser does not request the page from the web server. Instead, the browser simply reloads the page from its cache. The solution to this problem was to disable caching on all billing solution and administration tool servlets and JSP pages. This is achieved by appending headers to the HTTP response using the *HTTPServletResponse* interface. Figure 4.24 contains a snippet of code included at the top of all billing solution and administration tool servlets and JSP pages.

```
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Pragma", "no-cache");
response.setDateHeader("Expires", -1);
```

Figure 4.24 – Extract from Billing4Rent solution servlets

The Billing4Rent solution has been designed to ensure that the user must be authenticated in order to access the billing solution or the administration tool and, in addition, the pages are not cached. Therefore, attempts to use the back button after the user has logged out of the system will not result in the page being rendered. If the user attempts to access a page which was created using a POST request, a page expired message will be displayed (Figure 4.25). If the user elects to refresh the page, they will be redirected to the appropriate login page.

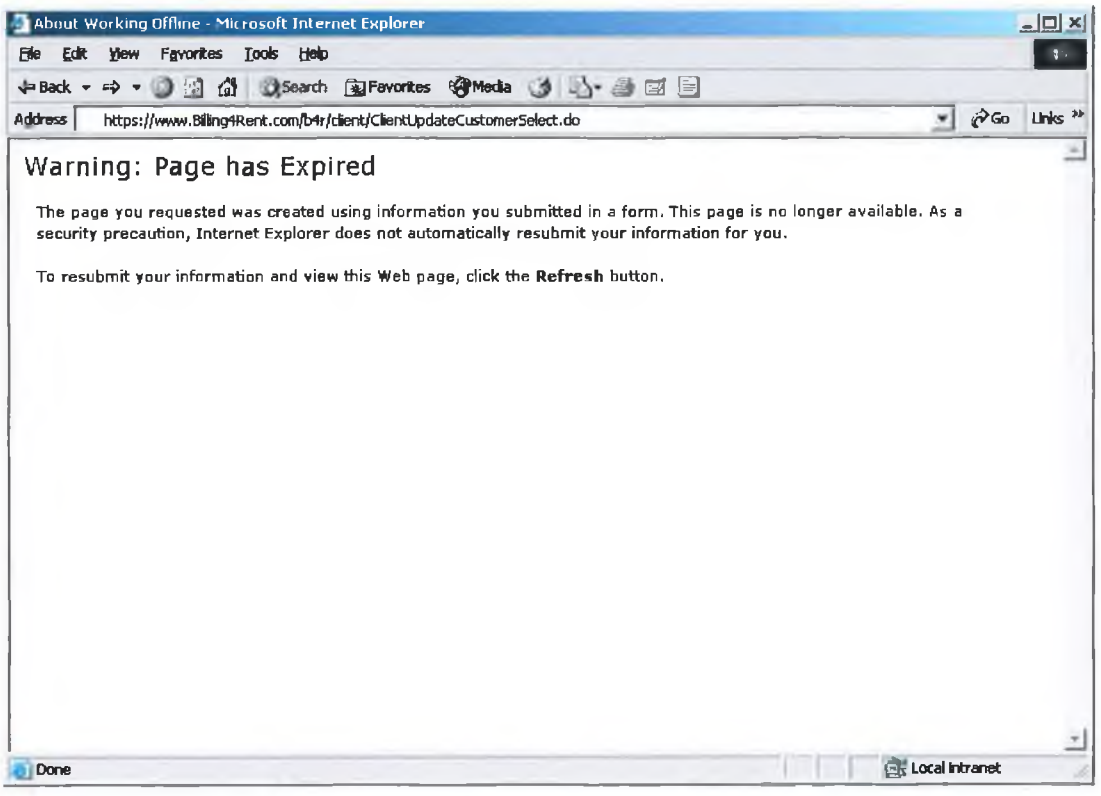


Figure 4.25 – Page has Expired

Unfortunately, if the user elects to refresh the login page the username and password parameters are resubmitted from cache by default and the browser has no means of knowing that this is in fact a security breach. In order to resolve this issue, the login page was updated to include a hidden field that is dynamically initialised at runtime with the current system time. Each time a user attempts to login to the system, the current login time is compared to the last successful login time for the corresponding user, which is stored in the USER table in the database. If the current login time is greater than the last successful login time, the user is permitted access to the system. Otherwise they are redirected to the login page. Figure 4.26 contains a snippet of code extracted from the RdbmsClientLoginModule class library, which stores the last successful login time in the B4RCLIENT table in the oracle database.

```
currentLoginTime = Long.parseLong(loginTime.trim());

if(dbLoginTime != null)
    previousLoginTime = Long.parseLong(dbLoginTime.trim());

if ((currentLoginTime > previousLoginTime) || (dbLoginTime == null)) {
    SQLUpdate = new StringBuffer("UPDATE B4RCLIENT set last_login_time = '')
        .append(loginTime)
        .append(" where CLIENTID = '")
        .append(user)
        .append("'")
        .toString();

    // Execute the sql insert statement
    stmt.executeUpdate(SQLUpdate);
}
```

Figure 4.26 – Extract from the RdbmsClientLoginModule class library

4.6 Encryption

Encryption is the translation of data into an unintelligible form through the use of a secret code, in order to ensure confidentiality. Encrypted data is referred to as 'cipher' or 'cipher text', whereas unencrypted data is commonly known as 'plain text'. Decryption is the process of restoring data to its original form through the use of a secret code. Encryption is an effective means of ensuring the privacy and integrity of information stored locally or transferred over a network. In order to ensure the confidentiality and integrity of data transferred over public networks, the Billing4Rent project team designed the billing solution and the administration tool to use HTTPS. For an overview of the use of HTTPS in order to facilitate secure login and navigation see section 4.4. It is equally

important that sensitive data stored in secondary storage is protected from both external and internal threats. Although encryption is not a substitute for effective access control, one can obtain an additional measure of security by selectively encrypting sensitive data before it is stored in the database [90]. The Billing4Rent solution has been designed to encrypt solution passwords, using a one-way hash function, prior to inserting a password into the Oracle database.

Figure 4.27 contains a snippet of code extracted from the *Password* class library, which uses Message Digest algorithm 5 (MD5) encryption and BASE64 encoding to encrypt passwords. An encrypt() method generates a *MessageDigest* object that implements MD5. A message digest is the representation of text in the form of a single string of digits. MD5 is a widely used cryptographic hash function with a 128-bit hash value. The password is encoded into a sequence of bytes using the 8-bit Unicode Transformation Format (UTF-8) character set. The result is subsequently used as a parameter into the *MessageDigest* update() method. UTF-8 encodes each unicode character as a variable number of 1 to 4 octets, where the number of octets depends on the integer value assigned to the unicode character. The update() method fills the digest's buffer with data. Once the buffer contains all the necessary data the *MessageDigest* digest() method completes the hash computation by performing final operations such as padding. Finally the *BASE64Encoder* encode() method encodes the byte array into Base64 format. Base64 encoding takes three bytes, each consisting of eight bits, and represents them as four printable characters in the American Standard Code for Information Interchange (ASCII) standard.

```
String ALGORITHM  = "MD5";
MessageDigest md = null;

try {
        // Generate a MessageDigest object that implements
        // the specified digest algorithm.
        md = MessageDigest.getInstance(ALGORITHM);

        // Fill the digest's buffer with data to compute a
        // message digest from
        md.update(password.getBytes("UTF-8"));

        // Generate the digest. This does any necessary padding
        // required by the algorithm
        byte raw[] = md.digest();

        // Undocumented java class to Encode bytes to base64 to
        // get a string
        String hash = (new BASE64Encoder()).encode(raw);

        return hash;
} catch (Exception e) {
        throw new Exception(new StringBuffer("[Password] ")
                .append(e.getMessage())
                .toString());
}
```

Figure 4.27 – Extract from the Password class library

Subsequently when the user attempts to login to either the Billing4Rent billing solution or the administration tool, the login module uses MD5 encryption and BASE64 encoding to encrypt the password input by the user. Thus the username and password input by the

user can be compared to the username and the encrypted password stored in the Oracle database.

4.7 Strong Passwords

Access to both the Billing4Rent web application and computer systems is restricted to authorised personnel through the enforcement of strong passwords. The stronger the password is the more difficult it is for both humans and machines to figure it out. Best practices denote that a strong password should be at least eight characters long and should be composed of a combination of uppercase, lowercase, numeric and punctuation characters [96], [97], [98].

Figure 4.28 contains a snippet of code extracted from the Password class library, which ensures that all billing solution and administration tool passwords adhere to the above guidelines. Firstly, a regular expression representing a strong-password is initialised (Figure 4.28, Figure 4.29). A regular expression is a means of describing a set of strings based on common characteristics shared by each string in the set. It is worth noting that there are several different types of regular expression syntax e.g. Perl, Python and PHP. The regular expressions in the Java java.util.regex package are very similar to Perl syntax. Java regular expressions are based on perl regurlar expressions as a result of their popularity and the power of the perl approach.

```
^(?=.*\\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[^\\w])(?!.*\\s).*$
```

Figure 4.28 – Strong Password regular expression

Construct	Matches
^	The beginning of a line
(?=.*\\d)	Must contain at least one digit
(?=.*[a-z])	Must contain at least one lower case character
(?=.*[A-Z])	Must contain at least one upper case character
(?=.*[^\\w])	Must contain at least one non word character
(?!.*\\s)	Allow whitespace characters
.* $	The end of a line

Figure 4.29 – Strong Password regular expression construct

Secondly, the password is validated to ensure that it is greater than eight characters long. If the password is not greater than eight characters, the method returns false and an appropriate message is displayed to the user (Figure 4.30). Thirdly, a *Pattern* object is created and the regular expression is compiled into a pattern. The *Pattern* object is used to create a *Matcher* object by calling the *Pattern* object's matcher() method. Finally, the matches() method attempts to match the entire input sequence against the pattern. If the password does not match the strong password pattern, the method returns false and an appropriate message is displayed to the user (Figure 4.31).

```
// Pattern to ensure a string is composed of a numeric ,
// an uppercase, a lowercase and a punctuation char
String pattern = "^(?=.*\\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[^\\w])(?!.*\\s).*$";
try {
        // Passwords must be greater than 8 characters in length
        if(password.length() > 8) {
                success = true;
        }
        if(success) {
                // Create a pattern object and compile the given regular
                // expression into a pattern
```

```
                Pattern p = Pattern.compile(pattern);

                // Create a matcher object that will match the given input
                // against this pattern
                Matcher m = p.matcher(password);

                // Attempt to match the entire input sequence against the
                // pattern on success return true otherwise false
                success = m.matches();
        }

        // return true/false
        return success;

} catch (Exception e) {
        throw new Exception(new StringBuffer("[Password] ")
                .append(e.getMessage())
                .toString());
}
```

Figure 4.30 – Extract from the Password class library

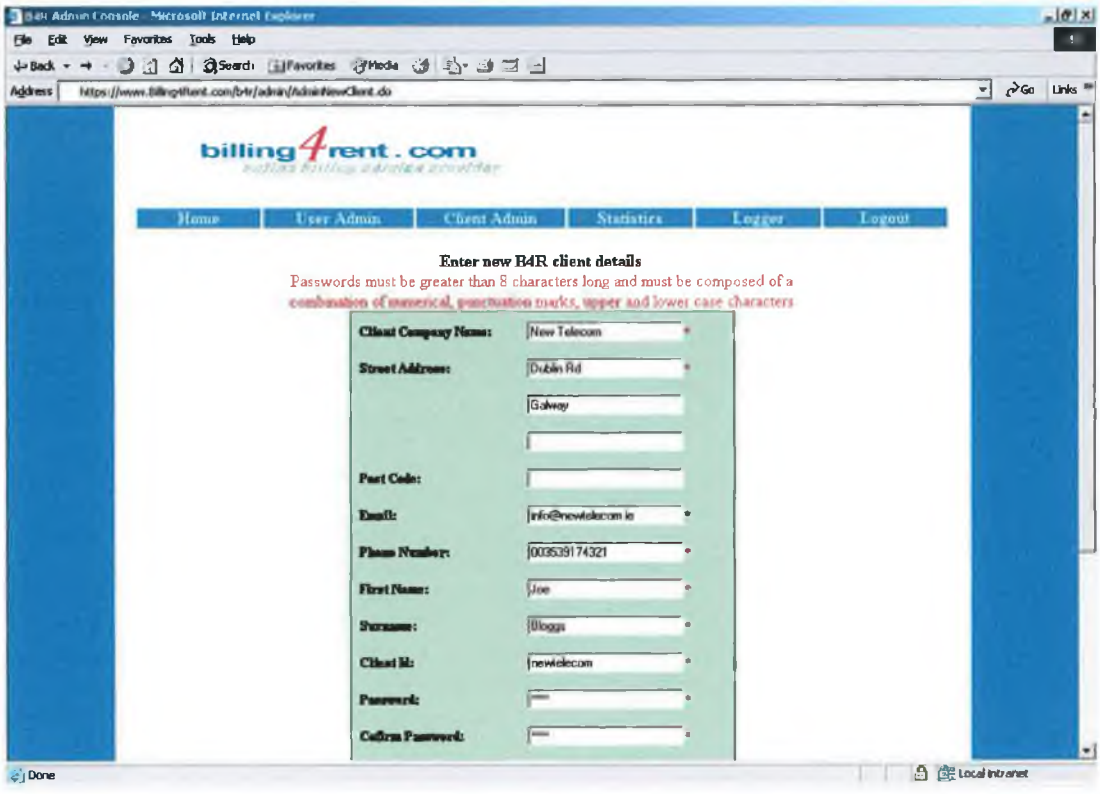


Figure 4.31 – Billing4Rent administration tool add new client

4.8 Timeout

As a security measure, the Billing4Rent billing solution and the administration tool timeout after a specified period of inactivity. In Apache Tomcat the session timeout is configured in web.xml, the deployment descriptor of the web application. Alternatively, the timeout can be specified using the *HTTPSession* object's setMaxInactiveInterval() method. The timeout internal, the time period a session can remain idle for before the server terminates it is specified in minutes. Both the billing solution and the administration tool have been configured to timeout after ten minutes of inactivity (Figure 4.32). When either the billing solution or the administration tool timeout due to inactivity the *Subject* object is destroyed, the session is invalidated and the user is

presented with the login page. If the user tries to use the back button or type the path directly into the browser they will be redirected to the login page.

```
<session-config>
        <session-timeout>10</session-timeout>
</session-config>
```

Figure 4.32 – Extract from Billing4Rent web.xml

4.9 File Upload

The Billing4Rent team elected to use a combination of Apache Commons FileUpload and HTTPS in order to facilitate the secure uploading of client collateral. Figure 4.33 contains a snippet of code extracted from the *ClientFileUpload* servlet, which uses the Apache Commons FileUpload package in order to provide the client with a means of securely uploading electronic resources. Firstly, a DiskFileUpload object is instantiated and used to parse the request, in order to obtain a list of items to stream to the web application. Secondly, an *Iterator* object is used to loop through the list. If the list item is an uploaded file, the list getName() method, which returns the full path of the file on the client machine, is used to create a *File* object. The *File* object's getName() method, which returns only the file name, is subsequently used to create a new *File* object by concatenating the full destination path with the filename. Finally, the uploaded file is written to disk. Figure 4.34 and Figure 4.35 contain screenshots from the Billing4Rent billing solution depicting the selection of the file to be uploaded and display of the said file respectively. For security reasons, the uploaded file should be checked for viruses prior to being written to disk. Symantec AntiVirus Corporate Edition can be configured for real-time scanning and on-demand scans.

```
// Create a new DiskFileUpload object
DiskFileUpload upload = new DiskFileUpload();

// Use the handler to parse this request
// Giving us a list of items from the request
List items = upload.parseRequest(request);

// Create an iterator object
Iterator itr = items.iterator();

while(itr.hasNext()) {

    FileItem item = (FileItem) itr.next();

    // Check if the current item is a form field or an uploaded file
    if(!item.isFormField()) {

        if(item.getName().equals("")){

            String msg = "File name required!";

            // Store error msg in in a request object
            request.setAttribute("msg", msg);

            // Forward to uploadFile jsp page
            dispatcher = getServletContext()
               .getRequestDispatcher("/client/clientFileUpload.jsp");

            dispatcher.forward(request,response);
        } else {
```

```
                File fullFile  = new File(item.getName());

                // Construct destination path for the file to be uploaded
                File savedFile = new File((new
                    StringBuffer(getServletContext().getRealPath("/"))
                    .append("images\\")
                    .toString()), fullFile.getName());

                // Write the file to disk
                item.write(savedFile);
            }
        }
    }
```

Figure 4.33 – Extract from the ClientFileUpload servlet

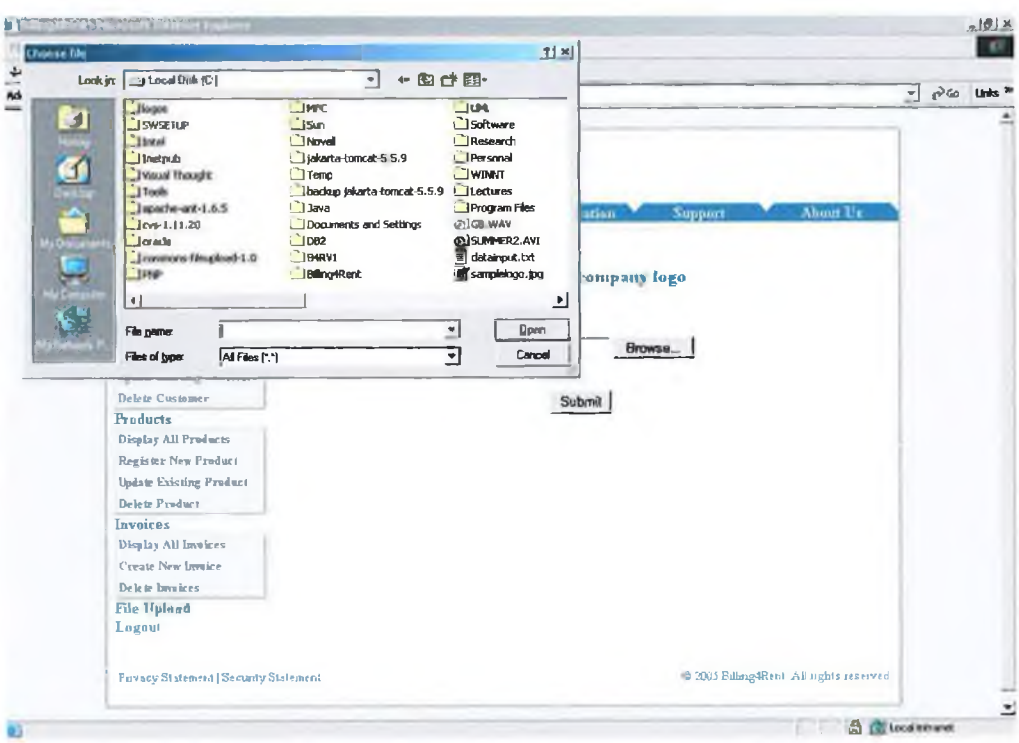


Figure 4.34 – Billing4Rent billing solution upload file

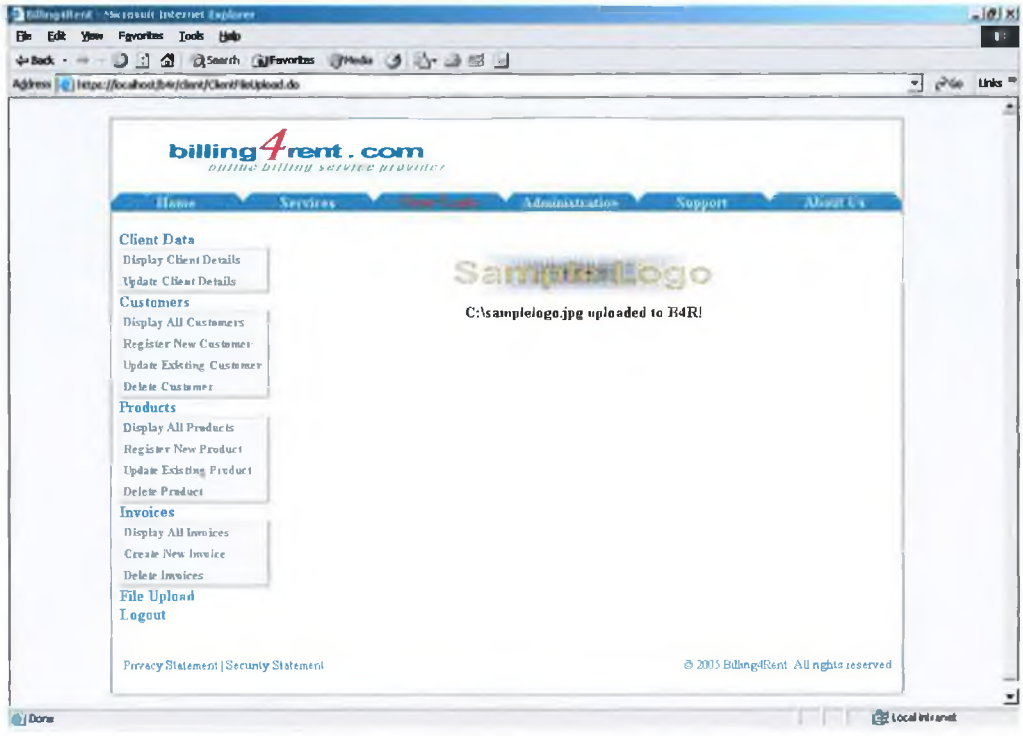


Figure 4.35 – Billing4Rent billing solution file uploaded

4.10 Logging

In the event of a security breach, logging helps determine the extent of the damage caused. Both the Billing4Rent billing solution and the administration tool have been developed to support audit, error and security logging. After examining a number of logging applications, the Billing4Rent development team elected to implement a custom logging system, due to the unique nature of the Billing4Rent ASP logging requirements. Figure 4.36 contains a snippet of code extracted from the *ClientLogin* servlet, which concatenates a string using a *StringBuffer* object and uses a *B4Rlogger* object to add data to the B4RSECURITYLOG table in an oracle database. The B4RSECURITYLOG table is composed of the following fields: LOG_ENTRY, USER_ID, DATE_LOGGED, STATUS, MODULE and LOG_MSG. The B4Rlogger object is used to manipulate the error, audit and security logs. The error log details any errors that occur throughout the Billing4Rent web application. The audit trail provides an overview of client activity, e.g.

when they logged in and what actions they performed. Finally, the security log outlines access to secure resources and the performance of operations deemed to be of a sensitive nature.

```
securitymsg = new StringBuffer("Successfully logged in B4RClient [")
   .append(client)
   .append("] into b4r application")
   .toString();
log = new B4Rlogger(debug);
log.addToLogger("Sys Admin", "1", MODULE, securitymsg,
"B4RSECURITYLOG");
```

Figure 4.36 – Extract from the ClientLogin servlet

The administration tool provides Billing4Rent administrators with the ability to view and delete logs, which are stored in audit, error and security tables in the oracle database. Figure 4.37 provides a snap shot of the *B4RsecurityLog*, which can be filtered based on the UserID, Date Logged or Status fields.

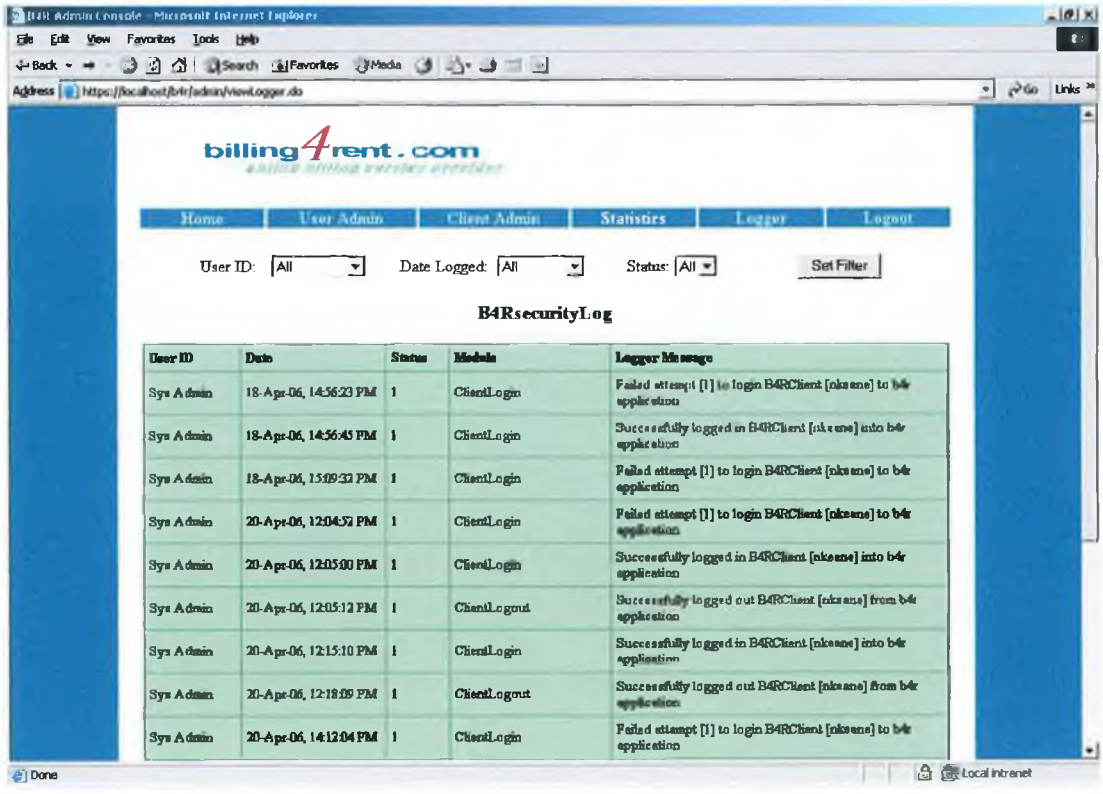


Figure 4.37 – Security log

4.11 Prototype

The Billing4Rent solution prototype is subdivided into three interconnected applications: the Billing4Rent Ltd. web application, the billing solution and the administration tool. In this section, the author provides an overview of the technologies used and outlines the functionality of each of the applications. Relevant Billing4Rent prototype screenshots are included as *appendix II.*

The ASP Billing4Rent solution prototype is composed of a number of JSPs, Java servlets, enterprise beans and libraries, interfacing with an Oracle database. The solution architecture was designed based on the Model-View-Controller (MVC) design pattern and the prototype was deployed in a J2EE environment. The prototype was developed in order to facilitate the testing of security libraries and to optimise configuration. Due to

the unpredictable nature of the Billing4Rent project, the prototype architecture was designed to enable loosely coupled components to interact by using high-level abstractions and established object oriented design principles and patterns. MVC is a composite design pattern that can be used at both an architectural macro level and at a finer grained micro level. The MVC architecture provides loose coupling by dividing the application into three distinct components. Modifications to one component can be made with minimal impact to the others. The model component encapsulates the application data, the view component is responsible for displaying the data stored in the model and the controller component facilitates the exchange of data between the model and the view (Figure 4.38). A high level of abstraction was achieved by designing library classes to implement interfaces and abstract classes and by developing polymorphism into the design to facilitate loose coupling and change.

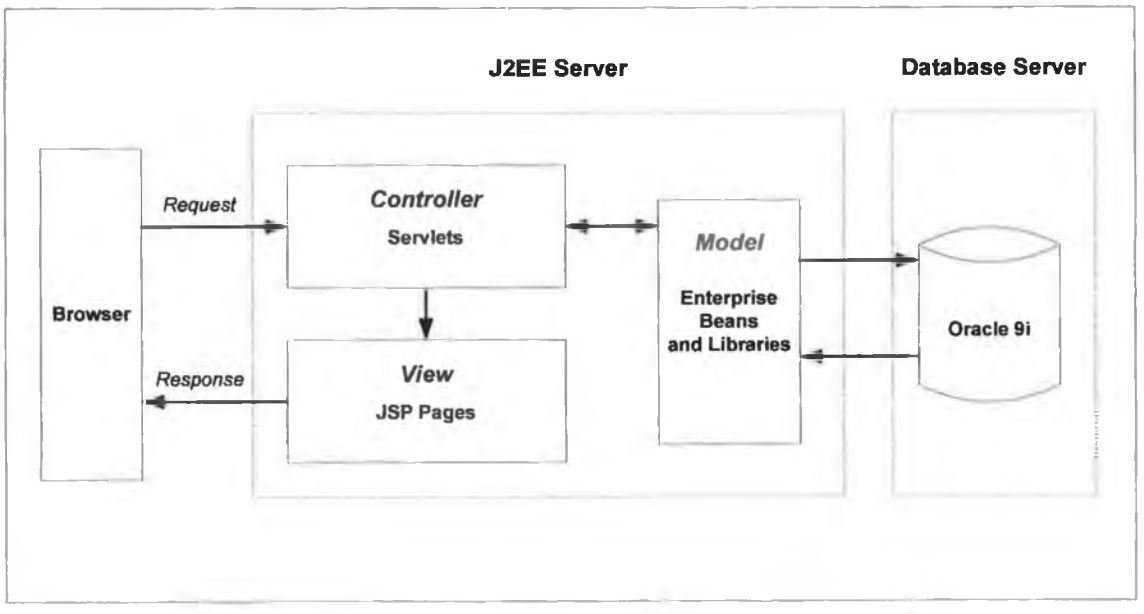


Figure 4.38 – Model-View-Controller architecture

4.11.1 The Billing4Rent Ltd. Web Application

In addition to providing information about the ASP service, the Billing4Rent Ltd. web application provides a means of accessing both the billing solution and the administration tool. As the Billing4Rent prototype was also designed to ensure optimal performance in addition to security, access to the Billing4Rent Ltd., is permitted via HTTP. As the Billing4Rent billing solution and administration tool both use HTTPS, the first time a user attempts to access a secured page on the Billing4Rent site, they will be presented with a dialog containing the details of the certificate and asked if they wish to proceed with the security transaction (Figure 4.39).

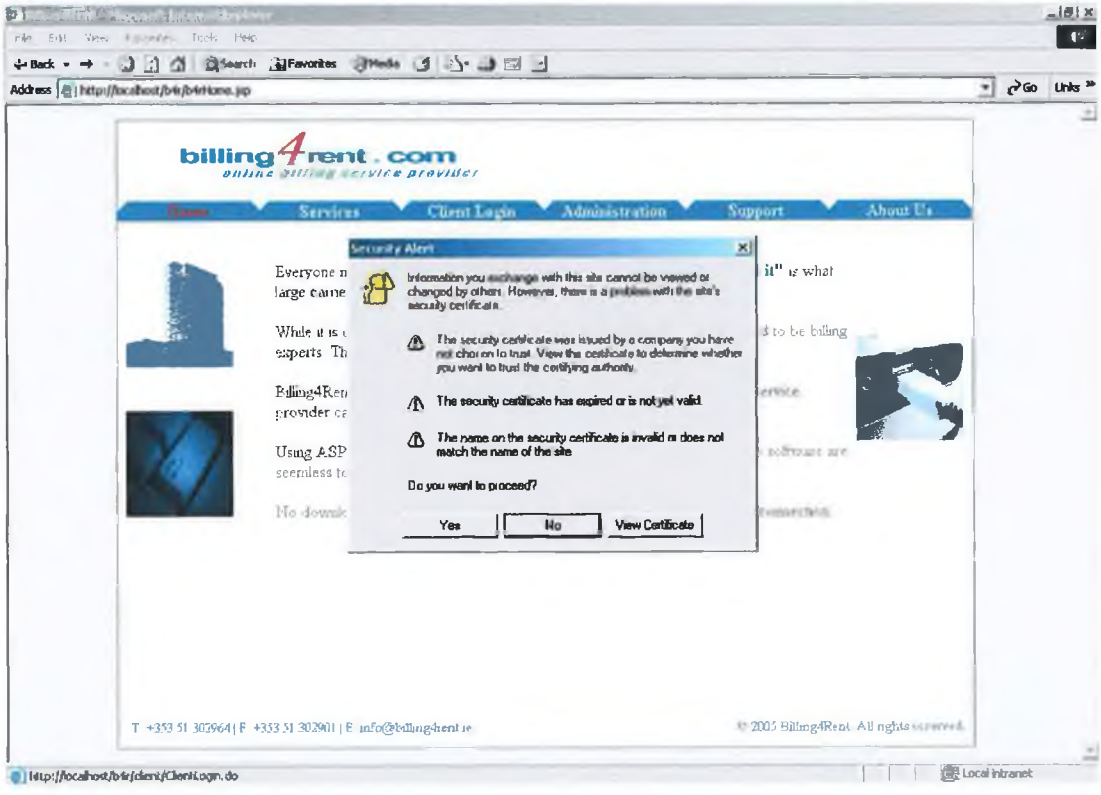


Figure 4.39 – Billing4Rent billing solution security alert

4.11.2 The Billing Solution

The primary functionality of the Billing4Rent billing solution is to provide billing functionality on a subscription or rental basis. The billing solution provides Billing4Rent clients with a means to configure and maintain their customer and product details and to generate billing information in the form of invoices (Figure 4.40).

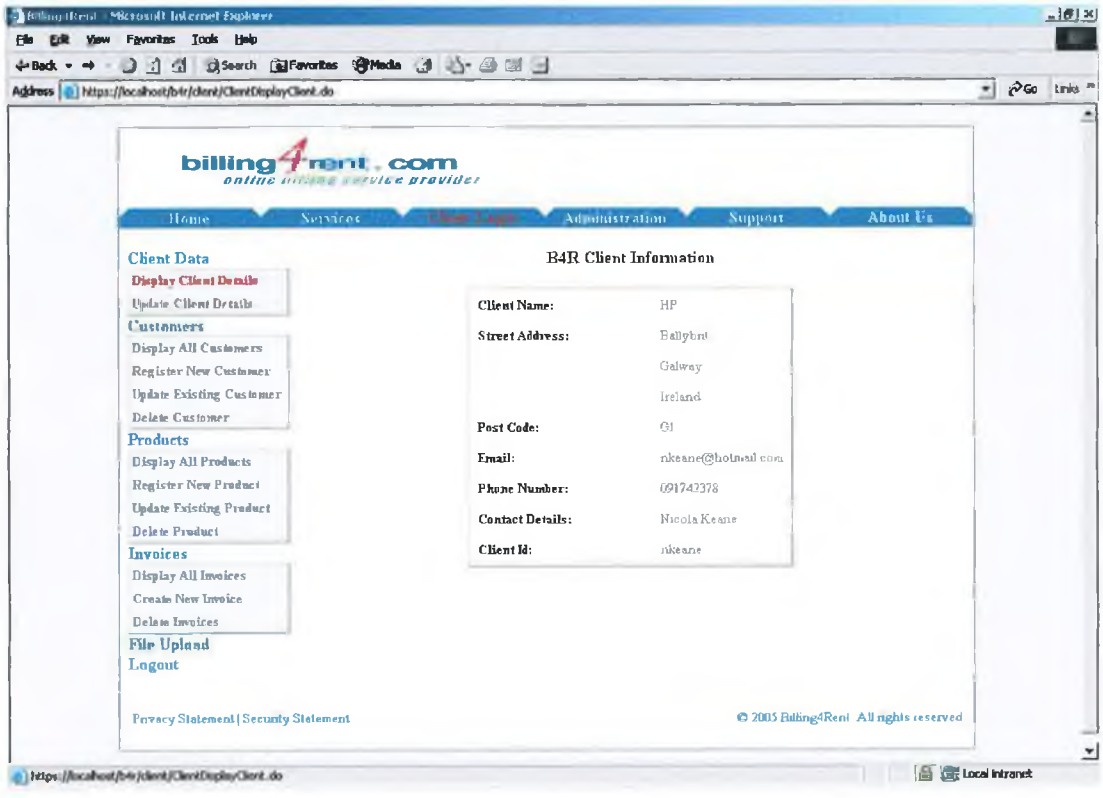


Figure 4.40 – Billing4Rent billing solution

Figure 4.41 provides Billing4Rent clients with a means to view and update the data held by Billing4Rent with regard to their organisation.

Figure 4.41 – Client Data menu

Figure 4.42 allows Billing4Rent clients to manage their customer base. The 'Display All Customers' menu option details all customers the client has configured to date. Customer ID, company name, contact name and a field to indicate if this profile is enabled or disabled are displayed for each customer. The company name links to a page containing additional customer contact data such as address, email and phone number. The 'Register New Customer' menu option allows the client to configure additional customers. The 'Update Existing Customer' menu option enables the client to select an existing customer from a dropdown list. The client can thus update the corresponding customer details displayed on the update customer form. The 'Delete Customer' menu option provides the client with a means to delete one or more customers configured to date. However, invoices associated with deleted customers are not removed from the system by default.

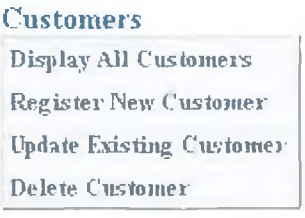


Figure 4.42 – Customers menu

Figure 4.43 facilitates the configuration of client products. The 'Display All Products' menu option displays a list of client products under the headings product ID, name, unit charge and enabled flag. The flag is used to indicate which products are currently enabled. Each product name can be used as a link to additional product data, such as description, invoice string, tax percentage etc. The 'Register New Product' menu option displays a form which can be used, by the client, to input new product details. The 'Update Existing Product' menu option allows the client to select a product, to be updated, from a dropdown list. Subsequently, the selected product details are displayed and the application provides the client with the ability to update one of more properties, with the exception of the product ID. The 'Delete Product' menu option allows the client to delete one or more products. As with the deletion of customers, associated invoices are not deleted by default.

Figure 4.43 – Products menu

Figure 4.44 provides Billing4Rent clients with a means to generate billing information in the form of invoices. The 'Display All Invoices' menu option provides the client with the ability to select from a drop down list of customers and displays a summary of invoices generated to date for that customer. The invoice number provides a link to the corresponding invoice header and line item details. The 'Create New Invoice' menu option provides the client with the ability to select from a dropdown list of customers and subsequently create an invoice for the selected customer. The invoice number and date are generated automatically. The client must supply a corresponding purchase order and must add one or more line items to the invoice. The 'Delete Invoices' menu option provides the client with the ability to select from a dropdown list of customers and subsequently facilitates the deletion of one or more invoices created for that customer.

Figure 4.44 – Invoices menu

The 'File Upload' menu option permits the secure uploading of client collateral. The 'Logout' menu option invalidates the session and redirects the client to the client login page. For an in-depth look at the functionality provided by the Logout and File Upload menu options see section 4.5 and 4.9 respectively.

4.11.2.1 Privacy statement

In an ASP environment, the privacy protection of client information is critical. Figure 4.45 details the privacy practices of Billing4Rent Ltd. Users can access the privacy statement by clicking the link at the bottom left hand corner of the client login tab.

The following provides and overview of Billing4Rents privacy statement:

- Billing4Rent respects the client's right to privacy and upholds both the Privacy and Electronic Communications Directive (Directive 2002/58/EC) and the Data Protection Directive (Directive 95/46/EC).
- Billing4Rent takes all reasonable security steps to protect the information provided by their clients.
- Billing4Rent will only process personal data with clear permission from the client. Any data submitted to Billing4Rent will not be disclosed to a third party without explicit and unambiguous consent.
- Registration is an entirely voluntary process.
- All registered clients have secure direct access to their own set of data. Clients have the right to amend their own data if it contains errors, or if some data requires updating.
- All registered clients have the right to de-register from Billing4Rent at any time.

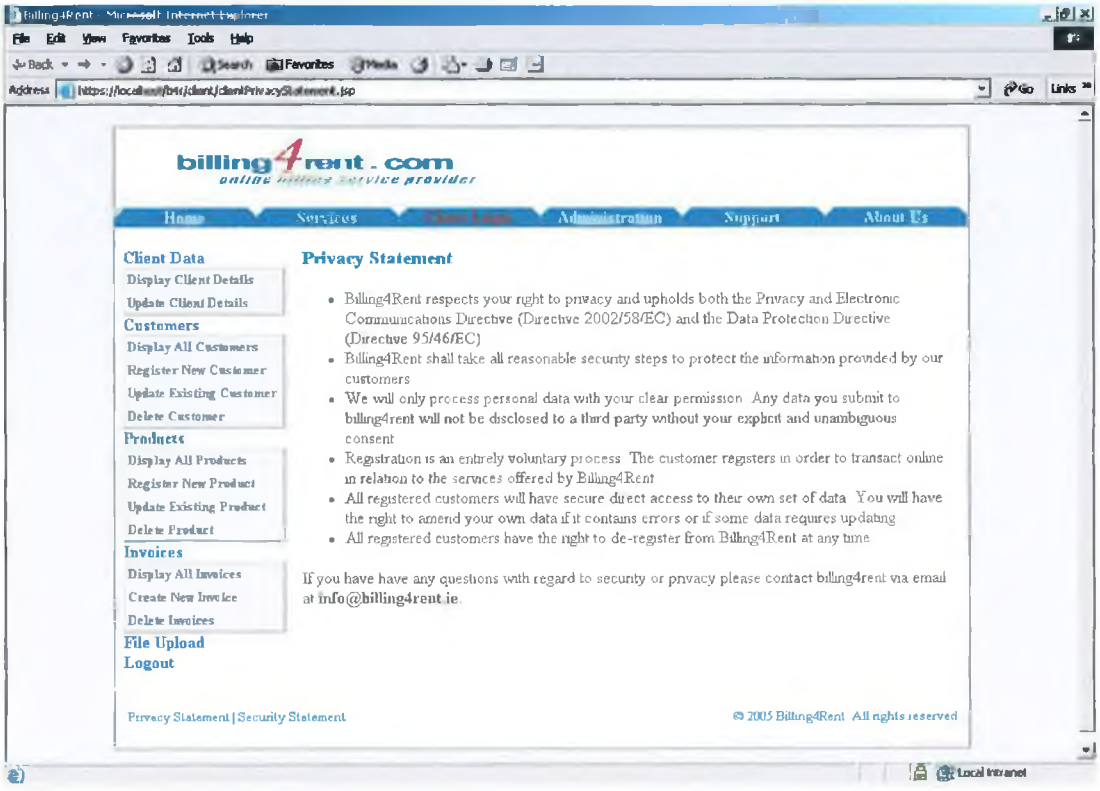


Figure 4.45 – Privacy Statement

4.11.2.2 Security statement

Figure 4.46 details the security practices of Billing4Rent Ltd. Users can access the security statement by clicking the link at the bottom left hard corner of the Client Login tab.

The following provides and overview of Billing4Rents security statement:

- Security and your privacy are of the utmost importance to Billing4Rent.
- Billing4Rent web services are only accessible through secured networks designed to prevent unauthorised activity.
- Client data is stored on separate database machines that are further protected and have no direct public access.

- Clients must login using their username and password in order to gain access their personal information. Provided clients keep their password secret their personal information will be protected.
- The Billing4Rent web application uses encryption technology to ensure that the personal information being passed between the client and Billing4Rent is scrambled and therefore cannot be read or understood by third parties.
- Billing4Rent uses cookies to help strengthen security. A cookie is a piece of information saved on a client machine by a web application. The cookies used on the Billing4Rent web application allows the web-server to identify which page they should send to the user next and to verify that the person who is asking for the next page in the secure area is actually the same person who passed the customer login.

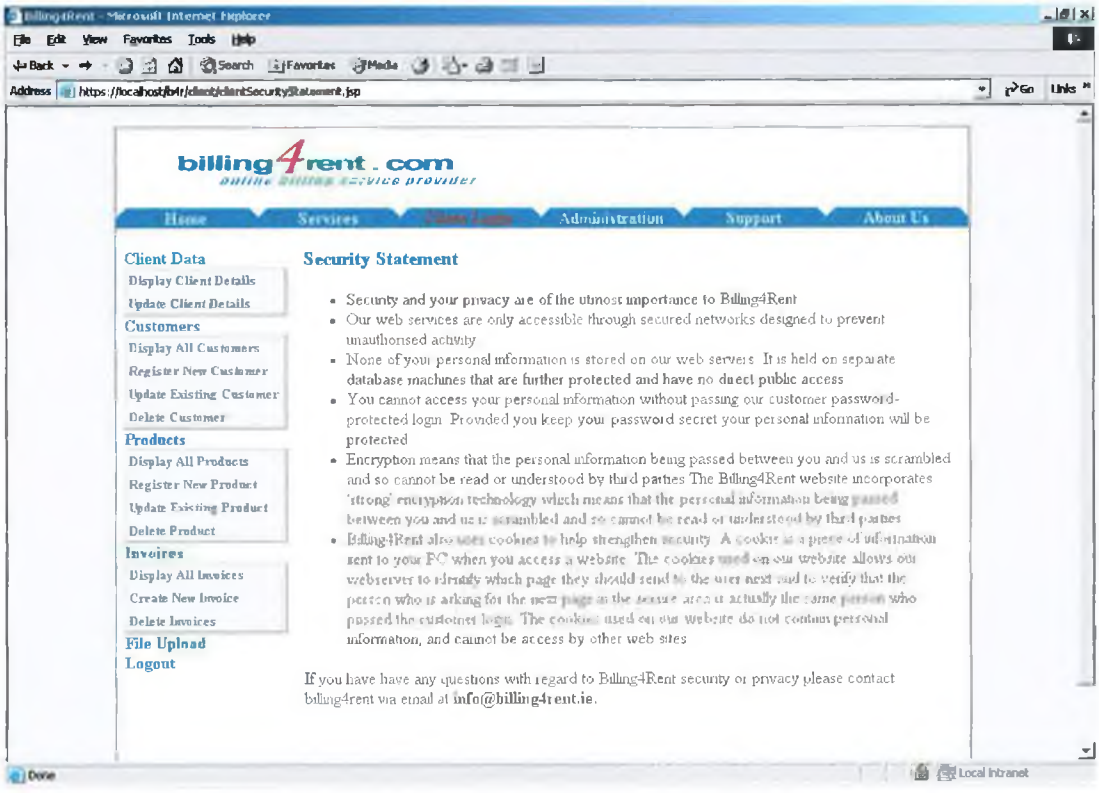


Figure 4.46 – Security Statement

4.11.3 The Administration Tool

The administration tool provides Billing4Rent employees with the ability to configure and maintain Billing4Rent client details and to monitor both the security and the performance of the billing solution (Figure 4.47).

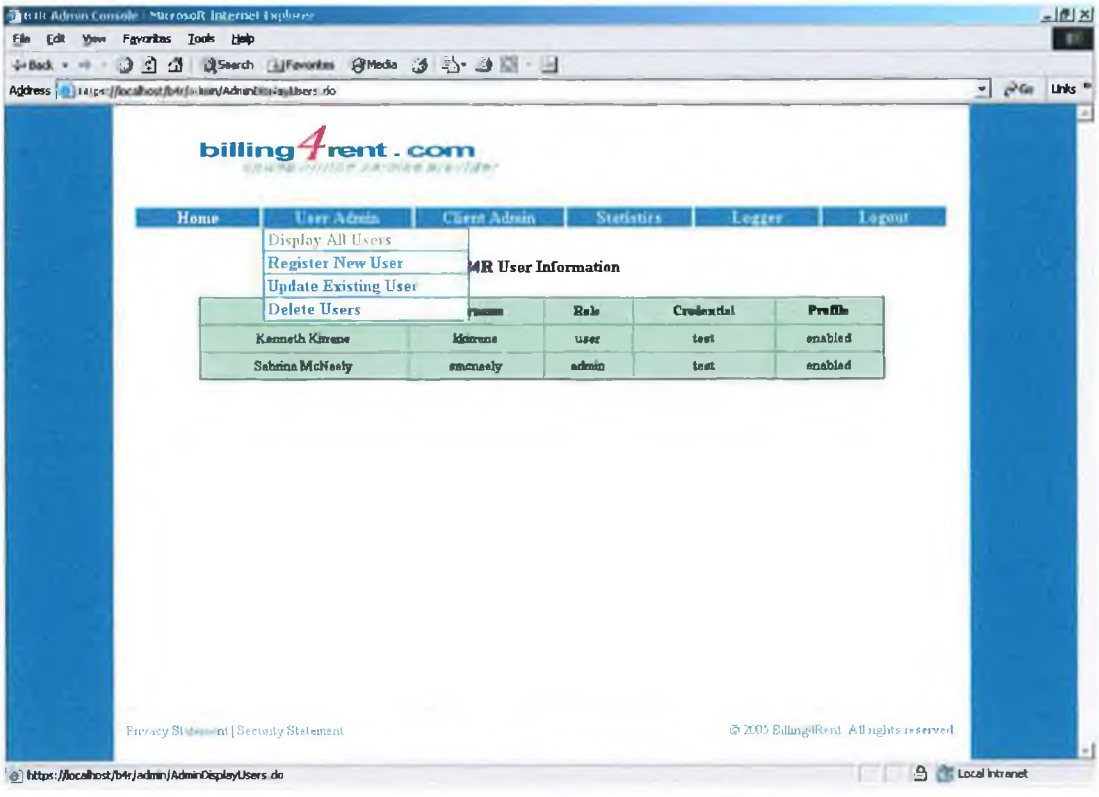


Figure 4.47 – Billing4Rent Administration tool

Figure 4.48 provides Billing4Rent administrators with the ability to manage the administration tool users. The 'Display All Users' menu option displays the name, username, role and public credentials of all users configured to date. The profile field indicates whether a profile is currently enabled or disabled. The 'Register New User' menu option allows the administrator to configure addition administration tool users. The 'Update Existing User' and 'Delete Users' menu options enable the administrator to update or delete existing users respectively.

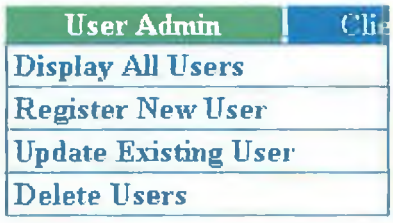


Figure 4.48 – User administration menu

Figure 4.49 facilitates the configuration of Billing4Rent clients. The 'Display All Clients' menu option displays a list all clients configured to date. The 'Register New Client' menu option allows the administrator to configure new clients. The password configured by the administrator should be updated by the client the first time they access the billing solution. The 'Update Existing Client' menu option enables the administrator to update existing client details. Alternatively the client can update their details using the billing solution 'Update Client Data' menu option. The 'Delete Clients' menu option provides the administrator with a means to delete Billing4Rent clients. As a result all invoices and products associated with this client are deleted by default.

Figure 4.49 – Client administration menu

Figure 4.50 provides Billing4Rent administrators with the ability to view statistical data with regard to the billing solution.

B4R Tables Sizes

B4R User	Number of Rows in Table: 2
B4R Client	Number of Rows in Table: 1
B4R Customer	Number of Rows in Table: 2
B4R Product	Number of Rows in Table: 3
B4R Invoice Header	Number of Rows in Table: 4
B4R Invoice Details	Number of Rows in Table: 6
B4R Error Log	Number of Rows in Table: 4
B4R Audit Log	Number of Rows in Table: 201
B4R Security Log	Number of Rows in Table: 13
B4R Admin Error Log	Number of Rows in Table: 1
B4R Admin Audit Log	Number of Rows in Table: 112

Figure 4.53 – B4R Table Sizes

The 'Launch JAMon' menu option opens a new window and displays performance data with regard to the Billing4Rent billing solution e.g. page hits, access times and module completion times (Figure 4.54). The performance monitor can be configured to monitor servlets, JSP pages and specific sections of code.

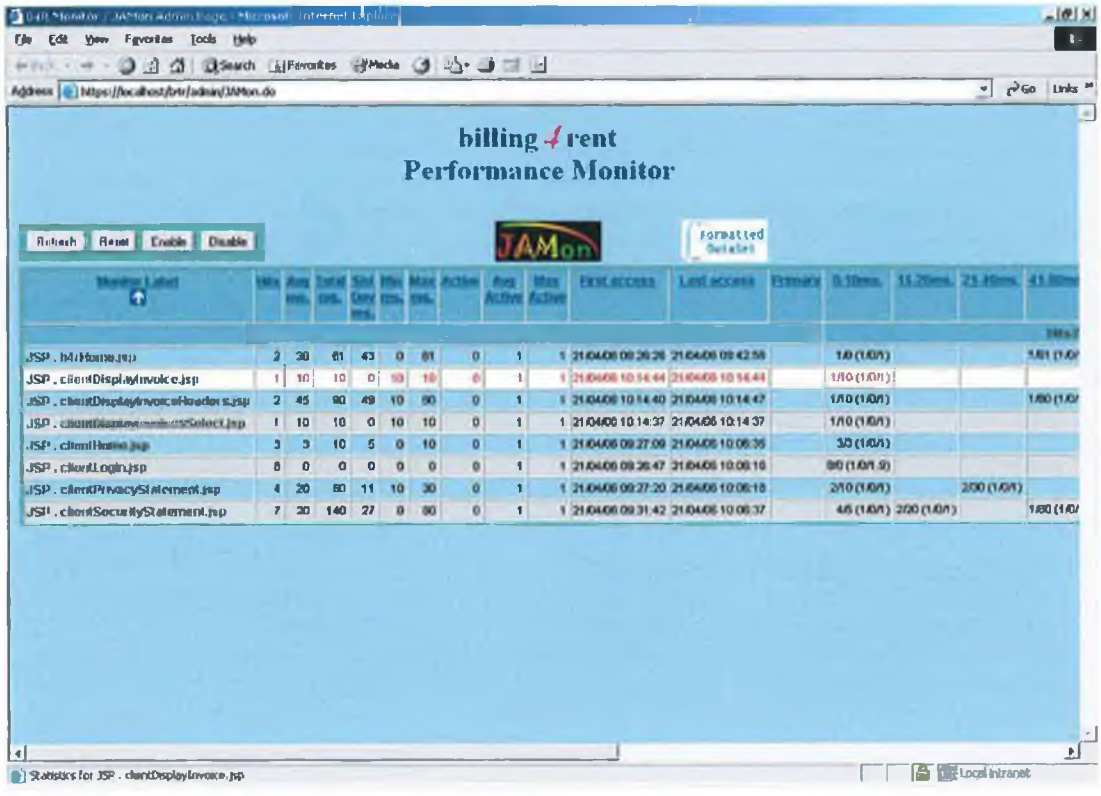


Figure 4.54 – JAMon performance monitor

Figure 4.55, the JAMon performance monitor provides the ability to view and delete the audit, error and security logs for both the billing solution and the administration tool. For an in-depth look at logging functionality provided see section 4.10.

Logger	Logou
Audit: View All	
Error: View All	
Security: View All	
Admin Audit: View All	
Admin Error: View All	
Audit: Delete Single	
Error: Delete Single	
Security: Delete Single	
Admin Audit: Delete Single	
Admin Error: Delete Single	
Audit: Delete All	
Error: Delete All	
Security: Delete All	
Admin Audit: Delete All	
Admin Error: Delete All	

Figure 4.55 – JAMon performance monitor

Finally, the 'Home' menu option provides links to all pages in the Billing4Rent administration tool and the 'Logout' menu option invalidates the session and redirects the client to the administration tool login page. For an in-depth look at the functionality provided by the Logout menu option see section 4.5.

4.12 Security Policy

The primary function of the Billing4Rent security policy is to outline guidelines for ensuring optimal security of the ASP solution. The Billing4Rent security policy is an ever-changing document, constantly amended to cater for updates to the organisation's systems and procedures. The Billing4Rent security policy is composed of a combination of guidelines with regard to general security, physical security, host security, network

security and data security. This section provides a high level overview of the Billing4Rent security policy. For further information please see *appendix III*.

4.12.1 General Security

The general security section of the Billing4Rent security policy provides security guidelines for the organisation. It describes the nature of the security policy and the actions required in order to ensure the Billing4Rent solution adheres to security best practices.

The general security section outlines the need to:

- Periodically audit the Billing4Rent infrastructure to ensure compliance with the security policy for the Billing4Rent ASP solution.
- Maintain a complete architecture document that includes a full network diagram of the Billing4Rent ASP environment, illustrating the relationship between the ASP environment and any other relevant networks. The document should include a full data flowchart detailing where client data resides, the applications that manipulate it, and the security thereof.
- Provide the ability to disable all or part of the functionality of the ASP solution should a security issue be identified.
- Perform non-intrusive network audits (basic portscans, etc.) randomly, without prior notice, and both intrusive network and physical audits, with 24 hours notice.

4.12.2 Physical Security

The physical security section details the protection of both ASP infrastructure and its location against theft, tampering and damage, either intentional or not. It recommends that Billing4Rent ASP infrastructure should be located in a physically secure data center.

Access to the facility should be restricted to authorised personnel, through the use of biometric scanners plus user pins or passwords. Closed Circuit Television (CCTV) should be configured in order to monitor activities throughout the data center. The infrastructure should be further secured physically, through the use of security cables, padlocks and other such devises. In order to achieve optimal security, a CCTV policy and an access control Policy should be devised and maintained. The CCTV policy should detail the type of system used and security footage storage methods. The access control policy should document stringent access control policies and procedures, outlining who is authorised to enter the data center, in addition to who is authorised to access the Billing4Rent ASP infrastructure.

4.12.3 Host Security

Host security is concerned with restricting access to the ASP solution to authorised personal. Appropriate service packs and patches should be applied to the Billing4Rent ASP operating systems and applications as soon as they are made available. The corporate standard anti-virus/anti-spyware software should be installed on all Billing4Rent machines. The anti-virus/anti-spyware site should be checked periodically for a list of updates, which should be installed as they become available. Access to Billing4Rent ASP solution should be restricted to authorised personnel by enforcing strong passwords and ensuring the implementing of an account maintenance policy. In order to facilitate the above Billing4Rent should document the following policies:

- Service packs/Patches policy
- Anti-virus/Anti-spyware policy
- Password policy
- Account maintenance policy

4.12.4 Network Security

The network security section provides guidelines for both the protection of hosts on the Billing4Rent network and the data transmitted over the Internet. The Billing4Rent ASP solution firewall should be configured to filter undesired traffic between the Internet and Billing4Rent ASP infrastructure. All unnecessary services running on the ASP infrastructure should be disabled, by shutting down unused ports. Remote access to Billing4Rent ASP solution hardware should be limited to an absolute minimum. Billing4Rent should document in the Billing4Rent ASP Remote and Dial-in Access Policies, standards for connecting to Billing4Rent's network from remote hosts. Data sent over the Internet should be encrypted to ensure security and integrity of data being transmitted as outlined in the Billing4Rent ASP Cryptography Policy.

4.12.5 Data Security

Data security is also concerned with ensuring the privacy and integrity of client data in an ASP environment. Redundant Billing4Rent ASP hardware should be disposed of in an appropriate manner. Software and data should be uninstalled and erased to guarantee that sensitive data it is not accessible to unauthorised individuals. Billing4Rent ASP solution software and the configuration of the Billing4Rent ASP environment should be tested rigorously to ensure the security and the integrity of client data. ASP solution security threats should be identified and the appropriate action should be taken to combat unauthorised access to sensitive data.

4.13 Chapter Summary

The ASP Billing4Rent solution prototype is composed of a number of JSPs, Java servlets and Java libraries, interfacing with an Oracle database. The solution architecture was designed based on the MVC design pattern and the solution was deployed in a J2EE environment. The Billing4Rent solution tackles security on a number of fronts (Figure 4.56).

Solution Security	**Description**
Architecture	Solution configured to use a demilitarised zone (DMZ). The DMZ segregates servers requiring external access from other machines on the network.
Hardware	Secure premises and the use of security cables and padlocks to further secure Billing4Rent infrastructure.
Operating System	Service packs, patches, anti-virus and anti-spyware installed and kept up to date.
Authentication	Implemented using the Java Authentication and Authorisation Service (JAAS) framework and custom pluggable login module.
Authorisation	Access control verified using Java Authentication and Authorisation Service (JAAS) framework security policy and *URLPermission* class.
Login and navigation	Hyper Text Transport Protocol/Secure Sockets Layer (HTTPS) encrypts network traffic.
Logout	*Subject* object is destroyed and the user's session is invalidated.
Data encryption	Uses MD5 encryption and BASE64 encoding to encrypt passwords.
Auditing	Due to the unique nature of the Billing4Rent ASP logging requirements, the Billing4Rent development team implemented a custom logging system.

Timeouts	The web applications timeout after ten minutes of inactivity. The timeout is configured in web.xml, the deployment descriptor of the web application.
Strong passwords	All passwords must be at least eight characters long and should be composed of a combination of uppercase, lowercase, numeric and punctuation characters.
File upload	Apache Commons FileUpload and HTTPS facilitate the secure uploading of client collateral.

Figure 4.56 – Solution security overview

This chapter details all aspects of security with regard to the ASP Billing4Rent prototype case study, through a combination of code snippets, configuration details, diagrams, tables and screen shots. In order to demonstrate how the security is incorporated into the larger solution we provided an overview of the core functionality of the Billing4Rent prototype case study.

Finally it examines the Billing4Rent security policy, which is composed of a combination of guidelines with regard to general security, physical security, host security, network security and data security

Chapter V : Research Evaluation

5.1 Introduction

Chapter three detailed the results of a comprehensive literature and technology review of Application Service Provision (ASP) with a particular emphasis on system security. Chapter four introduced the Billing4Rent solution prototype. It examined the implementation of system security in the Billing4Rent case study and illustrated the range and types of technologies that encompass the security framework. This chapter draws on the knowledge gained from both the literature and technology review and the Billing4Rent case study to provide an objective comparison and analysis of the practical implementation of system security in an ASP environment.

Tao [14] defines ASP security as a combination of both client data and server availability. Linthicum [21] and Anderson [22] delve a little deeper, breaking down server availability into hardware and software security breaches. In order to provide an in-depth evaluation of security in an ASP environment, the author further subdivides security as follows:

- Software security, including application, middleware, platform, data and network security.
- Hardware security, in particular, physical and network security.
- The security and the integrity of client data.

Although the Billing4Rent team incorporated a high level of security into the development of the Billing4Rent solution, best practice dictates that security should be continually monitored and improved on an ongoing basis [34], [43]. This chapter examines the Billing4Rent case study with regard to each of the aforementioned software components and the requirement for ongoing security enhancements.

5.2 Security Evaluation Matrix

The literature and technology review identified two security standards relevant to the examination of ASP security: the Open Standards Interconnect (OSI) security model 7498-2 and International Standards Organisation (ISO) 17799 security standard. OSI 7498-2 deals principally with solution security, whereas ISO 17799 is a generic information security standard predominately concerned with information security within an organisation. As this thesis is primarily concerned with system security requirements and implementation in an ASP environment, the author elected to evaluate the Billing4Rent case study based on the former.

The security evaluation matrix, depicted by Figure 5.1, lists the seven security services identified by OSI 7498-2, based on generally agreed security objectives. It indicates the method of protection required, i.e. software or hardware and the security technologies used. It identifies the need to supplement hardware and software security through the documentation and enforcement of a security policy. The Billing4Rent security policy is composed of a number of guidelines for ensuring optimal security of the ASP solution. It is an ever-changing document, constantly amended to cater for updates to the organisation's systems and procedures. Finally, the matrix provides an overall picture of the security of the Billing4Rent case study by rating the level of security provided by the solution as strong, medium, weak or not applicable. In the remaining sections of this chapter, the author examines the software and hardware technologies used to secure the Billing4Rent case study and the reasons pertaining to the security ratings outlined in the security evaluation matrix.

Security Service	**Method**	**Technologies**	**Security Policy**	**Security Rating**
Authentication	Software	JAAS	No	Strong
		Password validation	No	Strong
Access Control	Software	JAAS	Yes	Strong
		Java compiler and runtime	No	Strong
		Strong passwords Service packs Patches Anti-virus Anti-spyware	Yes	Strong
	Hardware	Firewall	Yes	Strong
		Physical security	Yes	Medium
Non - Repudiation	Software	Digital signature	No	Not applicable
Data integrity	Software	JSSE/ HTTPS	No	Strong
Confidentiality	Software	JSSE/ HTTPS	No	Strong
		Message digest	No	Strong
		Data encryption	Yes	Weak
Assurance / Availability	Software	Logging	Yes	Medium
		System monitoring	Yes	Medium
Notarisation / Signature	Software	PKI	No	Medium

Figure 5.1 – Security Evaluation matrix

5.3 Software Security

In this context, software security is a generalised expression used to describe the security of all software components that encompass an ASP solution. Software security incorporates the operating system software, the development environment software, the ASP application software, the middleware in a distributed environment, the database management system and the network communication software. This section examines the security of each of these discrete software applications in order to provide a complete appraisal of the overall solution.

5.3.1 Platform Security

The term platform can be taken to mean either solution hardware or solution software or a combination of both. The physical security section focuses primarily on hardware security; therefore this section examines platform security purely from a software perspective. Platform security, in a software context, can be used to denote both the operating system and the development environment.

Howard and LeBlanc [32] highlight the fact that several of the well-publicised computer security and virus problems relate to bugs in software, due to badly written software and poorly configured solutions. The Billing4Rent project team address this issue in their security policy. The security policy recommends that appropriate service packs, patches, anti-virus and anti-spyware software are installed on all Billing4Rent machines and are continually kept up to date by the system administrator. It proposes the adoption of authentication and authorisation mechanisms and provides guidelines for the enforcement of strong operating system passwords. Although the security policy is geared primarily towards the commercial stage of the project, where feasible the Billing4Rent team adhered to the recommendations therein throughout the implementation stage. By following best practices with regard to host security, Billing4Rent minimises the security risks attributed to poorly configured solutions.

The Billing4Rent ASP solution was developed and deployed in a Java 2 Platform Enterprise Edition (J2EE) environment. J2EE is composed of a set of standards for developing and deploying enterprise Java applications [105]. J2EE provides a multi-tier distributed application model, which can be divided into four main architectural tiers (Figure 5.2):

- Client tier
- Presentation tier
- Business logic tier
- Enterprise Information Systems (EIS) tier

The client tier incorporates various client application e.g. browsers, applets or standalone application clients. The presentation tier composes JSP and servlet web components, deployed in web containers. Business logic components implement business rules and access enterprise data. The Enterprise Information Systems (EIS) tier, commonly referred to as the back-end, is composed of database management systems and other legacy systems.

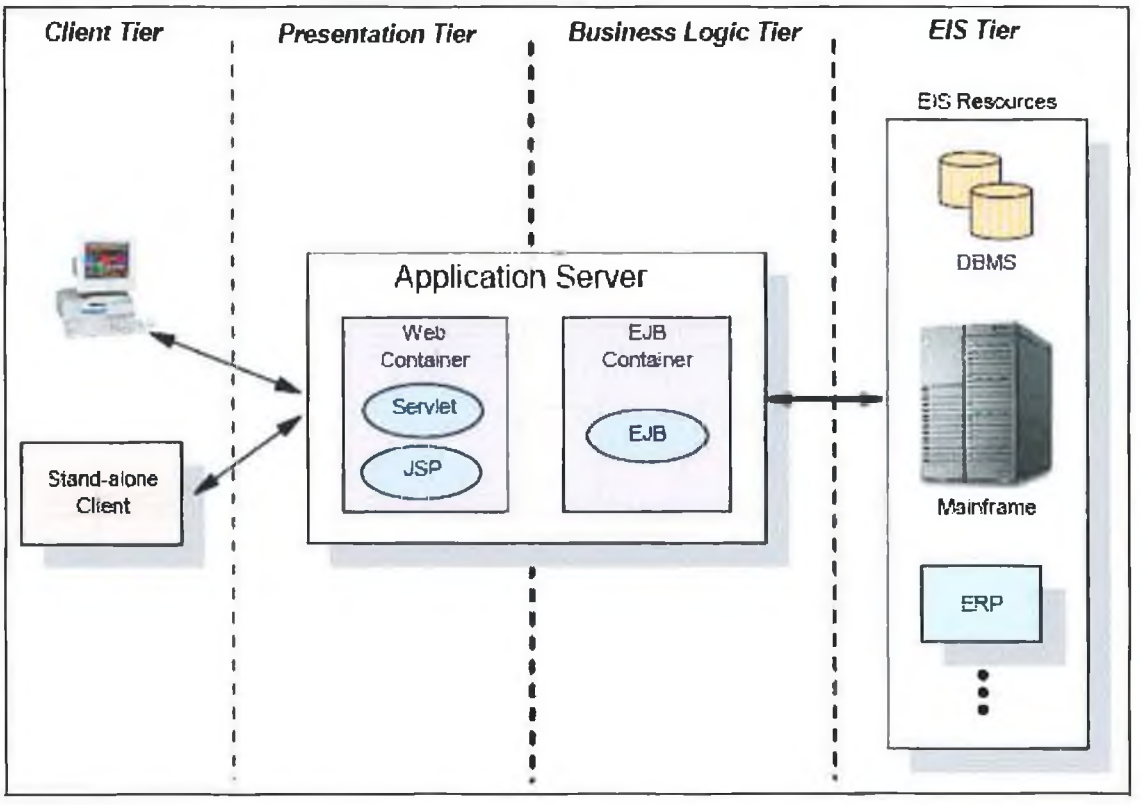


Figure 5.2 – Multi-tier application model adapted from [105]

The J2EE platform provides a secure development and runtime environment for Java applications. *"Data type checking at compile-time and automatic memory management leads to more robust code and reduces memory corruption and vulnerabilities. Bytecode verification ensures code conforms to the JVM specification and prevents hostile code from corrupting the runtime environment. Class loaders ensure that untrusted code cannot interfere with other Java programs"* [77]. In addition, the J2EE platform provides mechanisms for enforcing security both declaratively and programmatically. Declarative security is specified external to the application whereas programmatic security is embedded in an application. Declarative security is achieved through the specification of a security policy using the Extensible Markup Language (XML) within J2EE deployment descriptors. J2EE provides a number of extensible Application Programming Interfaces (APIs) to facilitate the development of secure communications, cryptography, authentication and access control security features. The Billing4Rent

solution was developed to make use of Java APIs where feasible, as they represent best practice and it would be unlikely that they could be improved upon. The Billing4Rent solution benefits from the code validation and memory management provided by the Java platform. The solution further benefits from a high level of abstraction provided through the use of declarative security, APIs and configuration files.

5.3.2 Application Security

Application security is concerned with restricting access to the ASP solution to authorised personal and minimising the security risks associated with badly written software. According to Morris and Thompson [31], an ASP solution is susceptible to threats at the remote terminal, along the communications link and at the physical machine. The ASP Billing4Rent solution prototype is composed of a number of JSPs, Java Servlets, enterprise beans and libraries, deployed in a J2EE environment. This section details a number of measures undertaken by the Billing4Rent team to safeguard the Billing4Rent solution against unauthorised access via the remote client.

5.3.2.1 Client Authentication

Joyce [95] defines authentication as a means of verification of claimed identity at a point in time. Access to the Billing4Rent solution, via a browser is restricted through the use of the dynamic Java Authentication and Authorisation Service (JAAS) framework. JAAS is composed of a set of APIs that facilitate the authentication and authorisation of users or entities such as services. Although the JAAS framework can be used for both authentication and authorisation, this section focuses exclusively on client authentication. The JAAS API supplies a number of dynamically configured *LoginModules*, which perform the authentication using several disparate security infrastructures. Alternatively, it is possible to develop custom *LoginModules* that implement the *LoginModule* interface. The Billing4Rent project team chose to implement a custom *LoginModule*, responsible

for the authentication of login details against data stored in a tier 3 database. The primary benefits of the implementation of authentication using JAAS are the ease of development through the reuse of existing code libraries, the level of abstraction provided by the dynamically configured *LoginModules* and the ability to use multiple independent L*oginModules* simultaneously.

In order to gain access to the Billing4Rent solution, the client must first successfully log-in to the web application. The Billing4Rent team designed the solution to use form-based login to source a username and password from the user. The login details are subsequently verified, by the *LoginModule,* against validation data stored in the tier 3 database. Despite well-known security weaknesses, username and password combinations remain the most common means of authentication [55]. Passwords are subject to security threats such as brute force attacks and network eavesdropping [31], [55], [58]. In addition, passwords written down, stored online as cleartext or divulged to unauthorised individuals may result in compromised security [52]. In an attempt to minimise the aforementioned security risks, Billing4Rent adhered to best practices with regard to the creation of passwords [96], [97], [98]. The Billing4Rent solution enforces strict password selection guidelines. For example, all Billing4Rent passwords must contain more than eight characters and must be composed of a combination of alphanumeric characters, punctuation symbols and upper and lower case characters. In order to prevent attackers from checking several passwords, Pinkas and Sander [55] recommend that accounts should be locked after a predefined number of unsuccessful login attempts. As an additional security measure, the Billing4Rent solution has been designed to disable an account after three unsuccessful login attempts. The Billing4Rent administrator is responsible for re-enabling the account at the client's request. As such the Billing4Rent solution provides adequate protection against unauthorised access in the form of brute force attacks. A brute force attack is an attempt by a user or a program to try multiple combinations of characters, words or other commonly used strings to gain unauthorised access to a system. The author would like to highlight that the Billing4Rent solution uses the Hyper Text Transport Protocol over Secure Sockets Layer (HTTPS)

protocol to protect the privacy of login data transferred over the network. The protection of data security and integrity shall be reviewed in a section dedicated to network security.

5.3.2.2 Server Authentication

In an ASP environment, authentication is intrinsically twofold and as such includes the authentication of the application service provider in addition to user authentication as outlined above. Secure Sockets Layer (SSL) is a network protocol, which is used to ensure secure communication over the Internet. SSL provides authentication of the server and, where required, the client. HTTP is a request/response protocol used to retrieve web resources from a server and render them in a web browser. HTTPS is a secure version of HTTP where requests and responses are encrypted using SSL prior to being sent over the network. An X.509 certificate is used in order to facilitate server authentication. A server certificate contains information about the server that allows a client to identify the server prior to sharing sensitive information. The Billing4Rent project team generated a public/private key pair and corresponding X.509 certificate using Sun Microsystems' Keytool, a key and certificate management command-line utility. This self-signed certificate is used to verify the identity of the Billing4Rent organisation. Although self-authentication is acceptable as a means of proof of identity, in a prototype environment, the author recommends that this certificate should be replaced with a digital certificate sourced from a trusted third party, such as a certification authority, prior to deploying the service in a production environment. A digital certificate generated by a certificate authority it is deemed more secure than self-signed certificates, as certificate authorities confirm the identity of the applicant prior to issuing the certificate [106].

5.3.2.3 Authorisation

Once the user has logged into the Billing4Rent solution, access to resources are managed through the adoption of a stringent authorisation and access control policy. Stergiou [99] defines access control as the restriction of admission to resources to the users holding the appropriate privileges. Access to sensitive data displayed on Billing4Rent solution web pages is restricted through the use of the JAAS framework. JAAS enables the administrator to grant access rights based on who is running the code, in addition to what code is running [80], [81]. After the Billing4Rent user has been authenticated, a *Subject* object is instantiated and populated with one or more principals, where each principal represents an identity for that user. The Java security manager uses a security policy to grant or deny principals access to privileged resources. The Billing4Rent project team chose to implement a custom *URLPermission* class, which implements the *Permission* interface. The *URLPermission* class is used in conjunction with the security policy to permit or deny access to URL resources. Again, the JAAS framework facilitates ease of development through the reuse of existing libraries and provides a high level of abstraction through the use of a security policy. Access policies are declared in a single file that can be easily updated by the system administrator, without the need for code changes.

5.3.2.4 Errors in the Code or Application Logic

Software bugs such as errors in the code or application logic leave the computer system vulnerable to external attacks [32], [33]. McGrath [34] furthers this hypothesis, stating that "*vulnerabilities typically fall into two categories: bugs at the implementation level and flaws at the design level*". Ideally, applications should be rigorously tested to minimise the associated security risks. Ravi *et al* [35] emphasise the fact that best practice recommends considering security throughout the entire Software Design Life Cycle (SDLC). Potter and McGraw [33] in turn reveal the benefits that can be achieved by identifying security risks in the system and creating tests driven by those risks.

Unfortunately, there is no such thing as complete security in a usable system and consequently it is important to concentrate on reducing risk as opposed to wasting resources trying to eliminate it completely [40]. The Billing4Rent solution was developed using an iterative waterfall model, with security being a foremost consideration throughout the project life-cycle. The Billing4Rent team identified potential security risks from the offset, incorporated security into both the design and the development phases and created a number of corresponding test cases, with a view to at best eliminating or at worst minimising the risk associated with errors in the code or application logic. The author would like to highlight that security is also of utmost importance in the integration phase and throughout the entire life-time of the solution.

5.3.2.5 Logging and System Monitoring

Best practice dictates that security should be continually monitored and improved on an ongoing basis [34], [43]. Software logs help determine the extent of the damage caused by security breaches and assist in meeting the aforementioned objective. In addition, logging provides the system administrator with the data required to proactively monitor the system for security anomalies [41]. System monitoring provides a successful means of highlighting unforeseen threats in a timely manner and thus limits their impact [43]. The Billing4Rent solution has been designed to generate audit, error and security logs. The administration tool provides Billing4Rent administrators with the ability to access logs, which are stored in audit, error and security tables in an Oracle database. Although the current Billing4Rent system monitoring process is quite effective, the author believes there is scope here for improvement through automation. The author recommends further research into the feasibility of the Billing4Rent solution automatically generating a notification, such as an email, when it uncovers a suspected security breach.

5.3.2.6 Additional Recommendations

As discussed previously, the Billing4Rent solution uses the dynamic JAAS framework to handle both authentication and authorisation. JAAS uses the login configuration file to retrieve the name of a class which implements the *LoginModule* interface, the associated Java package and optional configuration options. The debug flag, Oracle database URL, Oracle driver, Oracle username and Oracle password properties are all defined, in the Billing4Rent login configuration file, as name-value-pairs. As both the Oracle user name and password are stored as cleartext in an unencrypted configuration file, the database is susceptible to attack should an unauthorised individual gain access to the host and subsequently, the configuration file. The author highlights the need for further development to enhance the overall security of the solution. All sensitive configuration options should be stored as encrypted data in the login configuration file and subsequently decrypted by the associated *LoginModule*.

5.3.3 Middleware Security

Middleware refers to the software layer between the operating system, including the basic communication protocols, and the distributed applications that interact via the network. This software infrastructure facilitates the interaction among distributed software modules [60]. In general, ASP refers to the supply of online software functionality to multiple clients on a subscription or rental basis, remotely via the Internet or a private network. However, a thin client could indeed be used instead of or in combination with, a web browser. Therefore, for completeness, the author elected to perform a review of middleware and corresponding technologies. The ASP Billing4Rent solution prototype was developed using web technologies and was deployed entirely in a J2EE environment. As such the solution benefited from the middleware provided by the web server.

5.3.4 Data Security

Bertino and Sandhu [88] emphasise the fact that damage and misuse of data held in database systems could result in disastrous consequences. The Billing4Rent solution prototype interacts with an Oracle 9i Relational Database Management System (RDBMS). Although Oracle 9i provides a high level of out-of-the-box security, the author believes it would be prudent to follow Oracle's security checklist [89] with regard to the configuration of Oracle9i, in order to maximise its security features.

As an additional means of security, the Billing4Rent solution has been designed to encrypt solution passwords, using a one-way hash function, prior to storing the password in the Oracle database. Subsequently, when the user attempts to login to Billing4Rent, the password they enter is encrypted, using the aforementioned hash function and compared against the password stored in the database for the specified user. In [90] Oracle suggest that encryption technologies could be used to obtain an additional measure of security, by selectively encrypting sensitive data prior to storing it in a database. The author suggests that the use of encryption to enhance the security of sensitive data warrants further research, especially with regard to the anticipated performance overhead.

5.3.5 Network Security

The Billing4Rent solution was deployed on a J2EE server, which uses a set of Java packages known as Java Secure Socket Extension (JSSE), to enable secure Internet communication. Best practice dictates that sensitive data transmitted over a network should be encrypted to ensure that data cannot be interpreted, should it be intercepted by a third party [46]. The Billing4Rent solution uses the HTTPS protocol for both login to and navigation of the Billign4Rent web application once the user has been authenticated. HTTPS is a secure version of the HTTP where requests and responses are encrypted using SSL or TLS prior to being sent over the network. SSL is the *de-facto* standard

[49], [47], [50], [51] used to secure communication over the Internet. As discussed earlier, the Billing4Rent project team used Sun Microsystems' Keytool utility to generate a public/private key-pair and associated X.509 certificate. The public/private key-pair is used to initiate secure communication between a server and a browser. Subsequent transmissions are encrypted with a random symmetric key generated by the browser. As the Billing4Rent prototype was also designed to ensure optimal performance, access to general information that does not warrant security is permitted via HTTP. The section dedicated to software security highlighted the fact that HTTPS also facilitates the authentication of the server, and where required, the client, through the use of digital certificates.

5.4 Hardware Security

As discussed previously, the term platform is often attributed to system hardware, system software or a combination of both. Since this thesis has already examined platform security from a software context, this section focuses on the evaluation of the Billing4Rent case study from a hardware security perspective. Wang [26] defines hardware security as the physical protection of devices through the provision of a secure environment. This section examines the physical security of computer systems and communications equipment and the use of networking devices to protect all machines on the network.

5.4.1 Physical Security

Physical security focuses on the security of all hardware components that encompass the ASP solution. Bhagyavati and Hicks [25] define physical security as the locking up of assets such as networking infrastructure, computer systems and data storage, in order to provide protection from unauthorised monitoring, theft, corruption, and natural disasters. In the literature and technology review, the author identified four distinct hardware

threats. Firstly, ASP solution hardware and communications equipment could be damaged or destroyed by natural disasters, electrical surges, fire or water. Secondly, unauthorised access to the facility could result in ASP solution hardware being damaged, tampered with or stolen. Thirdly, the ASP solution hardware could be damaged, tampered with or stolen by disgruntled or former employees. Finally, authorised personnel could inadvertently damage ASP solution hardware.

The Billing4Rent security policy provides guidelines with regard to the physical protection of hardware devices and network media against the aforementioned threats, through the provision of a secure environment. The author recommends that the housing of hardware and communications equipment should be outsourced to a third party. The major advantage of adopting an outsourcing strategy is that organisations are free to concentrate on their core business [27], [28]. In addition, Billing4Rent can expect to benefit from economies of scale and a high degree of security, as data centers specialise in both the security and the availability of hardware and communications equipment. As outlined in the solution security section 3.3.2, although the security policy is more relevant to the commercial stage of the project, where feasible the Billing4Rent team adhered to the recommendations therein throughout the implementation stage. As such, the Billing4Rent hardware and communications equipment was stored, throughout the implementation stage of the project, in a lockable room within the main monitored, alarmed and secured campus building. Where feasible, the Billing4Rent ASP infrastructure was further secured through the use of security cables, padlocks and other such devices.

5.4.2 Network Security

The section dedicated to solution security examined network security from a software perspective. This section deals exclusively with the use of networking devices to protect network resources. A firewall is undoubtedly the most important security device on the organisation's network, as it creates a secure barrier between the organisation's internal

network and the outside world. A firewall can be composed of software or hardware, or a combination of both, configured to control access between the Internet and hosts connected to a private network [49], [51]. A firewall is responsible for filtering incoming and outgoing IP packets, depending on the rules configured by the administrator. These rules can be configured based on different network protocols, the network address of the destination or source, the port number, IP packet headers etc. [107]. Although access to the Internet is crucial for the ASP model, it can also constitute as a security threat to the organisation. The optimal solution is to use one or more firewalls to filter all traffic to and from the organisation.

Based on best practice outlined in [108], the author recommends the use of a demilitarised zone (DMZ) to protect devices on the Billing4Rent network. A DMZ is defined as a network that sits between the trusted internal network and the untrusted external network. All servers requiring external access are placed in the DMZ. External access to HTTP, HTTPS, FTP, FTPS and SMTP services on the DMZ network should be permitted through the outer firewall. Access to machines on the internal network should be permitted solely from machines in the DMZ through the inner firewall. Sun Microsystems, Microsoft and IBM all propose two-tier DMZ network architectures. Figure 5.3 adapted from [105] provides a high-level overview of the two-tier DMZ network architecture.

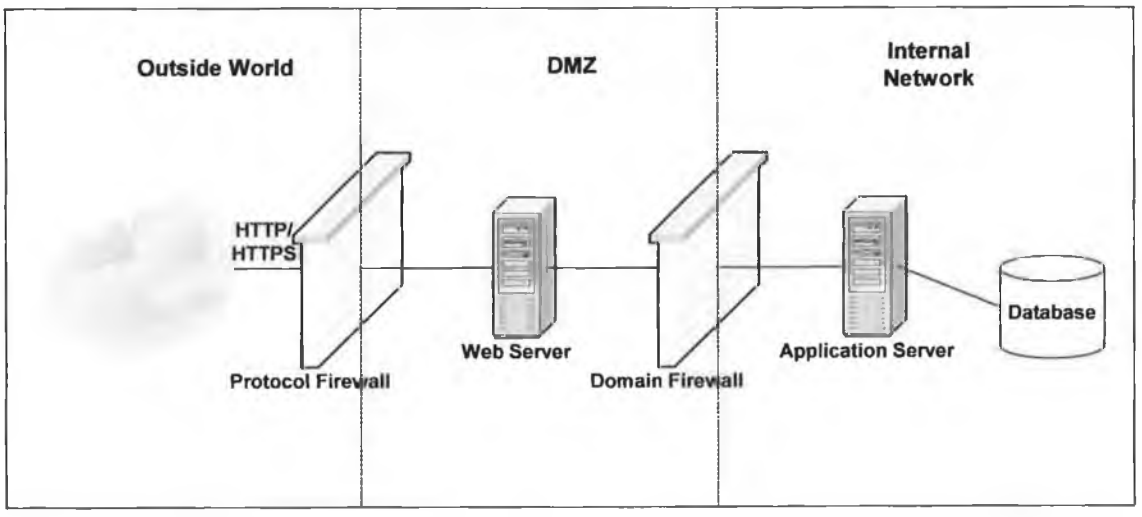


Figure 5.3 – Two-tier DMZ design adapted from [105]

Although a two-tier architecture provides adequate protection of the internal network, a three-tier architecture would provide an additional layer of abstraction and could thus be deemed more secure. META Group [109], which was acquired by Gartner in April 2005, outlines best practices pertaining to the design of secure network architectures. The three-tier architecture has been a popular choice for many organisations as it affords the opportunity to apply different security filters and services between each tier of common three-tier web applications, thereby facilitating a defense-in-depth strategy (Figure 5.4).

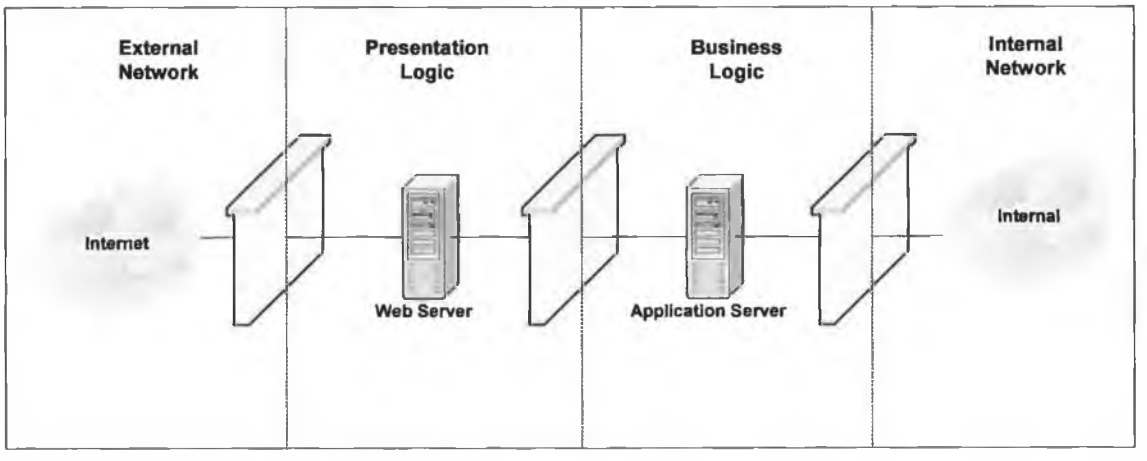


Figure 5.4 – Three-tier DMZ design adapted from [109]

META group [109] highlight the fact that although the three-tier DMZ design is still adequate, it is not ideally suited to today's objectives, such as minimising the amount of exposed infrastructure and economically accommodating access by both internal and external users to the same set of applications. Meta Group [109] expects to see many organisations migrate to the two-tier proxy-enabled design depicted in Figure 5.5. This design provides a greater level of security than both the two-tier DMZ and the three-tier DMZ designs as both web servers and application servers are no longer visible, as they reside on the internal network.

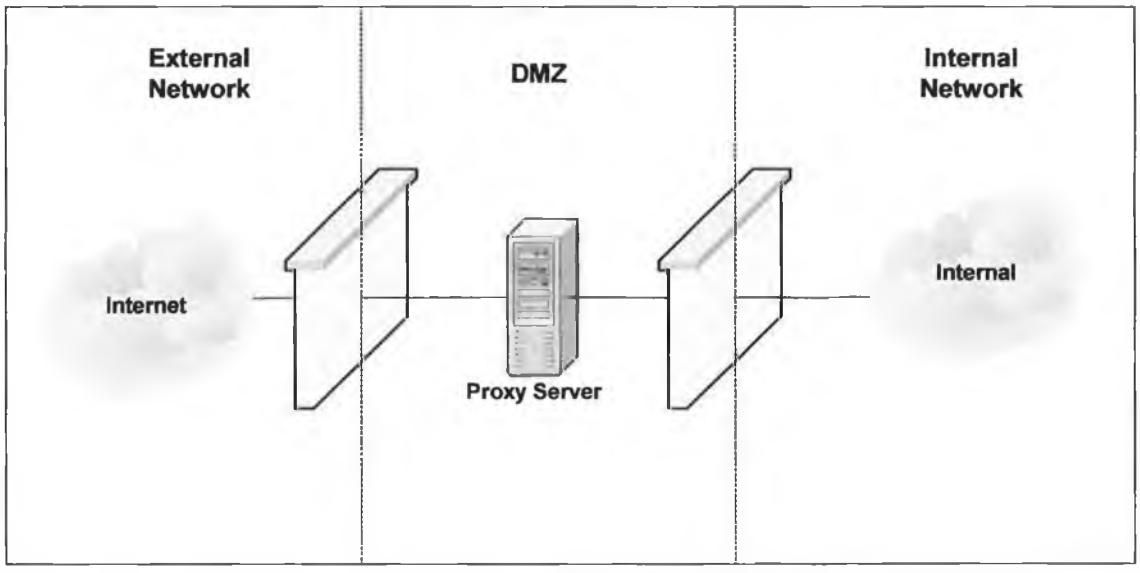


Figure 5.5 – Two-tier, proxy-enabled DMZ design adapted from [109]

As an additional measure of security, many firewalls incorporate a Virtual Private Network (VPN) server. VPNs provide secure remote access to network resources through the encryption of traffic between the firewall and remote users.

5.5 Client Data Security

Data security is intrinsically twofold: it includes the privacy protection and the soundness of the stored information [36]. Tao [14] outlines the need for ASPs to establish stringent procedures to ensure the security and integrity of customer data while it is under their care. As illustrated in the literature and technology review, there is a distinct overlap between both ASP physical and solution security, and the security and integrity of customer data. In addition, ASPs need to be aware of their legal obligations with regard to the protection of client data. In Europe, this legislation comes in the form of the Privacy and Electronic Communications Directive 2002/58/EC and the Data Protection Directive 95/46/EC. These regulations outline conditions for processing personal data and provide legislation to permit individuals and organisations to inspect personal data and to ensure this data is valid. The author suggests that by following best practices with

regard to the security of both hardware and software, the organisation ensures the security and integrity of data transmitted over the Internet. The Information Sensitivity Policy in turn provides guidelines with regard to what information can be disclosed to non-employees, as well as the relative sensitivity of information that should not be disclosed outside of the Billing4Rent organisation without proper authorisation. Although the Information Sensitivity Policy details the protection required for information at varying sensitivity levels, the author suggests that the Billing4Rent team should relate the levels of sensitivity specifically to the client data.

5.6 Chapter Summary

In this chapter the author uses the knowledge gained from both the literature and technology review and the Billing4Rent case study to provide an in-depth evaluation of the practical implementation of system security in an ASP environment. The author evaluated the security of the Billing4Rent case study from a hardware, software and data perspective, across all tiers of the service architecture. The author paid particular attention to best practices with regard to the consideration of security throughout the entire Software Design Life Cycle (SDLC) and the need to monitor and improve security throughout the lifetime of the solution.

The author devised a security evaluation matrix based on the Open Standards Interconnect (OSI) security model 7498-2, which in turn is based on generally agreed security objectives. It provided a detailed picture of the security of the Billing4Rent case study and enabled the reader see at a glance the level of security provided by the solution.

Finally, the author examined the security and integrity of client data based on legal obligations with regard to the protection of data outlined in the Privacy and Electronic Communications Directive 2002/58/EC and the Data Protection Directive 95/46/EC.

Chapter VI : Conclusion

This thesis focused on 'An Investigation into System Security Requirements and Implementation in an Application Service Provision (ASP) Environment'. The research conducted yielded a comprehensive framework of best practice with respect to system security within the ASP domain.

The literature and technology review identified security as one of the key factors influencing the uptake of the ASP model. It highlighted the fact that fears of inadequate security and privacy have prevented many firms from fully investigating and integrating the ASP business model. In the case study, the author examined system security in detail, through the research and development of all aspects of security with regard to the ASP model. In the evaluation chapter, the author used the knowledge gained from both the literature and technology review and the case study to provide an in-depth evaluation of the practical implementation of system security in an ASP environment. This chapter concludes the thesis with a summary of the research outputs, the conclusions drawn and provides a list of recommendations for future work.

6.1 Research Outputs

The research results can be divided into a number of distinct outputs:

1) The ASP solution prototype, a sample implementation of security in an ASP environment, based on proven infrastructure and best practices with respect to security.

2) A number of security libraries, designed to facilitate both change and reuse, achieved through the development of loosely-coupled components and the implementation of a high degree of abstraction.

3) A security policy, which outlines best practice with regard to general security, physical security, host security, network security and data security in an ASP environment.

4) A security evaluation matrix, which provides a high-level view of system security requirements in an ASP environment. It is envisaged that the matrix could be used to evaluate not only the security of ASP applications, but the security of any n-tier application.

6.2 Conclusions Drawn

The author concludes that security considerations in an ASP solution are the same as the security considerations of any other n-tier application such as banking or e-business.

1) Perceptions with regard to fears of inadequate security of ASP solutions and solution data are misguided and unfounded.

2) The technologies and mechanisms used to secure an ASP solution are in fact the tried and tested security technologies and mechanisms used to secure other n-tier applications.

3) ASP solution hardware and communications equipment, like other n-tier applications, can be outsourced to a third party such as a data center, as data centers specialise in both the security and the availability of such equipment.

4) Access to the ASP solution can be restricted through the adoption of strong authentication and stringent access control policies. Applications should be rigorously tested to minimise security risks attributed to badly written and configured solutions.

5) Standard HTTP security protocols should be used to secure any sensitive data transmitted from an ASP client to the hosted application. In the context of ASP, sensitive data is semantically broader than the conventional use of the term.

6.3 Recommendations

The author provides the following recommendations with respect to the research and development of system security in an ASP environment:

1) ASP solutions should be developed and deployed on tried, tested and trusted infrastructure. Proven infrastructure performance and security improves the overall likelihood of a successful solution deployment.

2) Existing Application Programming Interfaces (APIs) should be used where possible. Java provides a number of APIs, which facilitate the development of secure communications, cryptography, authentication and access control security features.

3) Security best practices should be adhered to, where feasible. Best practices represent techniques or methodologies, based on experience and research, that have resulted in optimal solution security practices.

4) Careful consideration should be given to how customer data is secured in an ASP environment. Such consideration should encompass how customer information is manifested in the middleware and how such data is secured in tier three database applications.
5) An effective security policy should be designed, implemented and maintained. The security policy should result in the protection of the organisation's assets from both external and internal threats, without impeding the organisation from achieving its objectives.

6.4 Future Research Potential

The author identifies significant further research potential, with regard to ASP system security:

1) The examination of the feasibility of using digital certificates, in an ASP environment, for client authentication, in addition to server authentication. Digital certificates could be used in combination with other authentication mechanisms to verify that the client is in fact who they claim to be. Special consideration should be given to the difficulty managing client certificates.

2) An investigation into the necessity of using digital signatures, in an ASP environment, to confirm the claimed identity of the originator and guarantee the validity of messages. Digital signatures enable security, and more importantly, accountability for electronic transactions.

3) An exploration into the use of encryption techniques to enhance the security of sensitive data stored in file systems and databases. Particular consideration must be given to what data should be encoded and the expected performance overhead. Database lookups are designed to facilitate searching through millions of rows for specific items in seconds. A database cannot afford to encrypt and decrypt each piece of data it must search.

In summary the author concludes that by following best practices with respect to security an ASP application can provide the same level of security one would expect from any other application. ASPs may in-fact provide greater levels of security than that which could be provided by a customer organisation.

References

[1] Exploratory research
http://en.wikipedia.org/wiki/Exploratory_research

[2] I.M. Crawford
Marketing Research and Information Systems
http://www.fao.org/docrep/W3241E/w3241e00.htm - Contents

[3] Agile software development
http://www.1stcustomsoftware.com/Agile_software_development-44.html

[4] Iterative and Incremental Development
http://www.liberty.edu/informationservices/development/index.cfm?pid=6355

[5] Winston W. Royce
Managing the development of large software systems: concepts and techniques
Proceedings of the 9th International Conference on Software Engineering

[6] Manifesto for Agile Software Development
http://agilemanifesto.org/

[7] Bhavini Desai, Vishanth Weerakkody, Wendy Currie, D. E. Sofiane Tebboune, Naureen Khan
Market Entry Strategies of Application Service Providers: Identify Strategic Differentiation
Proceedings of the 36th Annual Hawaii International Conference on Systems Sciences - 2003

[8] Nozar Daylami, Terry Ryan, Lorne Olfman, Conrad Shayo
Determinants of Application Service Provider (ASP) Adoption as an Innovation
Proceedings of the 38th Annual Hawaii International Conference on Systems Sciences - 2005

[9] Anjana Susarla, Anitesh Barua, and. Andrew B. Whinston
Making the Most Out of an ASP Relationship
IEEE IT Professional - Volume 3, Issue 6, Pages 63-67, November-December 2001

[10] Kenneth R. Walsh
Analysing the application ASP concept: technologies, economies, and strategies
Communications of the ACM – Volume 46, Issue 8, August 2003

[11] A. Bernstein
The State of ASPs
Robert Frances Group - 1999
http://www.cio.com/analyst/122799_rfgonline.html

[12] Thomas Kern and Jeroen Kreijger
An Exploration of the Application Service Provision Outsourcing Option
Proceedings of the 34th Annual Hawaii International Conference on Systems Sciences - 2001

[13] Anjana Susarla, Anitesh Barua, and Andrew B. Whinston
Understanding the Service Component of Application Service Provision: An Empirical Analysis of Satisfaction with ASP Services
MIS Quarterly – Volume 27, Issue 1, pages 91–123, March 2003

[14] Lixin Tao
Shifting Paradigms with the Application Service Provider Model
IEEE Computer – Volume 34, Issue 10, pages 32-39, October 2001

[15] Greg Goth
The Next Gold Rush: Application Service Providers Stake Their Claims in a Red-Hot Market
IEEE Software – Volume 17, Issue 2, Pages 96-99, March/April 2000

[16] Roland Klueber
ASP Strategies and Solutions for eProcurement Process Offered by an eMarket
Proceedings of the 35th Annual Hawaii International Conference on Systems Sciences - 2002

[17] Dr Philip Seltsikas and Prof. Wendy L. Currie
Evaluating The Application Service Provider (ASP) Business Model: The Challenge of Integration.
Proceedings of the 35th Annual Hawaii International Conference on Systems Sciences - 2002

[18] A. Konary - 2004,
"Presentation to the ITAA", IDC Analyst Presentation to the ITAA - February 12, 2004,
http://www.idc.com/

[19] Graham Winch, Wendy L. Currie and Philip Joyce
Making the ASP model work: using SD to explore trade-offs in IT service product design
22nd International System Dynamics Conference, Keble College, Oxford - July 2004

[20] B. Desai and W. Currie
Application Service Providers: A model in Evolution
Proceedings of the 5th international conference on Electronic commerce - 2003

[21] David S. Linthicum
To ASP or Not to ASP? – 2002
http://www.softwaremag.com/

[22] Troy E. Anderson
Management Guidelines for PC Security
ACM SIGICE Bulletin - Volume 20, Issue 1, Pages 7-14, July 1994

[23] S. Weingart and G. Double
An Evaluation System for Physical Security of Computing Systems
Sixth AnnualComputer Security Applications Conference -1990

[24] A. Mader and S. Srinivasan
Curriculum development related to information security policies and procedures
Proceedings of the 2nd annual conference on Information security curriculum development InfoSecCD - 2005

[25] Bhagyavati and Glenn Hicks
A basic security plan for a generic organisation
Journal of Computing Sciences in Colleges – Volume 19, Issue 1, October 2003

[26] Andy Ju An Wang
Security Testing in Software Engineering Courses
34th ASEE/IEEE Frontiers in Education Conference

[27] Kate Gerwig
Apps on Tap: Outsourcing hits the web
Source netWorker archive –Volume 3, Issue 3, September 1999

[28] Chris Lonsdale and Andrew Cox
The historical development of outsourcing: the latest fad?
Industrial Management and data systems, PART 8/9, pages 444-450, 2000

[29] Adrian Baldwin, Simon Shiu and Marco Casassa Mont
Trust Services: A Framework for Service-Based Solutions
26th Annual International Computer Software and Applications Conference

[30] Leonard H. Fine
Computer Security – A Handbook for Management
Irish Management Institute Dublin – 1983

[31] Robert Morris and Ken Thompson
Password Security: A Case History
Communications of the ACM – Volume 22, Issue 11, Pages 594–597, November 1979

[32] Michael Howard and David LeBlanc
Writing Secure Code
Microsoft Press – 2001

[33] Bruce Potter and Gary McGraw
Software security testing
IEEE Security & Privacy Magazine - Volume 2, Issue 5, Pages 81-85, September-October 2004

[34] G. McGraw
Software security
IEEE Security & Privacy Magazine – Volume 2, Issue 2, Pages 80-83, Marrch-April 2004

[35] Paul Kocher, Ruby Lee, Gary McGraw, Anand Raghunathan and Srivaths Ravi
Security as a new dimension in embedded system design.
Proceedings of the 41st annual conference on Design automation, June 2004

[36] S. Miranda
Aspects of data security in general-purpose data base management systems
Proceedings of the 1980 IEEE Symposium on Security and Privacy – April 1980.

[37] Commission launches infringement proceedings against nine Member States for not adopting new privacy rules for digital networks and services.
http://europa.eu.int/rapid/pressReleasesAction.do?reference=IP/03/1663&format=HTML&aged=1&language=EN&guiLanguage=en - file.tmp_Ref_1, 18th May 2005

[38] DIRECTIVE 95/46/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL
http://europa.eu.int/smartapi/cgi/sga_doc?smartapi!celexapi!prod!CELEXnumdoc&lg=EN&numdoc=31995L0046&model=guichett, 18th May 2005

[39] G. Spafford
Internet Security Know the Risk and Control it!
Director, Computer Operations, Audit, and Security Technology (COAST) Project, Purdue University

[40] Frederick M. Avolio
Best Practices in Network Security
CMP Media, March 20, 2000.
http://www.dushkin.com/text-data/articles/27509/body.pdf

[41] The Internet Engineering Task Force
Site Security Handbook
http://www.ietf.org/rfc/rfc2196.txt?Number=2196

[42] Geoffrey H. Wold and Robert F. Shriver
"RISK ANALYSIS TECHNIQUES - The risk analysis process provides the foundation for the entire recovery planning effort", Disaster Recovery World© 1997
http://www.drj.com/new2dr/w3_030.htm, 18th May 2005

[43] Ravi Sandhu and Pierangela Samarati
Authentication, Access Control, and Audit
ACM Computing Surveys, Vol. 28, No. 1, March 1996

[44] Jon Toigo
Disaster Recovery Planning – Preparing for the unthinkable – 3rd Edition - 2003
Prentice Hall PTR

[45] Anjana Susarla, Anitesh Barua and Andrew B. Whinston
Myths about outsourcing to application service providers
IEEE IT Professional - Volume 3 Issue 3, Pages 32-3, May-June 2001

[46] Gerald J. Popek, Charles S. Kline
Encryption and Secure Computer Networks
ACM Computing Surveys (CSUR) - December 1979 - Volume 11 Issue 4

[47] Vijay Ahuja
Network & Internet Security - 1996
Academic Press Inc.

[48] William Stallings
Network Security Essentials, Applications and Standards – Second Edition – 2003
Prentice Hall PTR, by Pearson Education Inc.

[49] Andrew S. Tanenbaum
Computer Networks – 4th Edition – 2003
Prentice Hall PTR, by Pearson Education Inc.

[50] Simson Garfinkel with Gene Spafford
Web Security, Privacy, and Commerce – 2nd Edition – 2002
O'Reilly and Associates Inc.

[51] William Stallings
Cryptography and Network Security, Principles and Practices – 2003
Prentice Hall PTR, by Pearson Education Inc.

[52] Roger J. Sutton
Secure Communications – Applications and Management – 2002
John Wiley & Sons, Ltd.

[53] Ray Hunt
PKI and Digital Certification Infrastructure
Ninth IEEE International Conference on Networks (ICON'01) - October 2001

[54] Albert Levi, M. Ufuk Caglayan, Cetin K. Koc
Use of nested certificates for efficient, dynamic, and trust preserving public key infrastructure
ACM Transactions on Information and System Security (TISSEC) - Volume 7, Issue 1, February 2004

[55] Benny Pinkas and Tomas Sander
Securing Passwords Against Dictionary Attacks
Proceedings of the 9th ACM conference on Computer and communications security, November 2002

[56] Wesley Chou
Inside SSL: The Secure Sockets Layer Protocol
IEEE IT Professional – Volume 4, Issue 4, Pages 47-52, July/August 2002

[57] Southern Methodist University (SMU)
School of Engineering
http://engr.smu.edu/~nair/courses/7349/ssl.ppt, 18th May 2005

[58] Liang Fang, Samuel Meder, Olivier Chevassut, Frank Siebenlist
Secure password-based authenticated key exchange for web services
Proceedings of the 2004 workshop on Secure web service SWS, October 2004

[59] Avishai Wool
A quantitative study of firewall configuration errors
IEEE Computer - Volume37, Issue 6, Pages 62-67, June 2004

[60] Kurt Geihs
Middleware Challenges Ahead
IEEE Computer - Volume34, Issue 6, Pages 24-31, June 2001

[61] Simon N. Foley, Thomas B. Quillinan, Maeve O'Connor, Barry P. Mulcahy, and John P. Morrison
A Framework for Heterogeneous Middleware Security
Proceedings of the 18th International Parallel and Distributed Processing Symposium - 2004

[62] A. Alireza , U. Lang ;, M. Padelis , R. Schreiner , and M. Schumacher
The Challenges of CORBA Security
Darmstadt University of Technology

[63] Zhonghua Yang and Keith Duddy
CORBA: A Platform for Distributed Object Computing
ACM SIGOPS Operating Systems Review - Volume 30, Issue 2 - April 1996

[64] Object Management Group
History of Corba
http://www.omg.org/gettingstarted/history_of_corba.htm

[65] Dionisis X. Adamopoulos, George Pavlou, Constantine A. Papandreou and Emmanuel Manolessos
Distributed Object Platforms in Telecommunications: A Comparison Between DCOM and CORBA
Managing the Network

[66] Chris Cleeland and Rob Martin
CORBA Security: An Overview
Object Computing Inc - 2002
http://www.omg.org/technology/documents/formal/omg_security.htm - Security_Service

[67] August Mayer
VI. Overview: CORBA security
Bowling Green State University - Summer 2001
http://student.cosy.sbg.ac.at/~amayer/projects/corbasec/sec_overview.html

[68] Chris Exton, Damien Watkins and Dean Thompson
Comparisons between CORBA IDL COM/DCOM MIDL: Interfaces for Distributed Computing
Monash University, Australia – 1997

[69] Alexander Davis and Du Zhang
A comparative study of DCOM and SOAP
Proceedings of the IEEE Fourth International Symposium on Multimedia Software Engineering - 2002

[70] Juval Lowy
Web Services Hurdle the Firewall
.Net Magazine, Volume 1, No. 1 - Fall/Winter 2001
http://www.ftponline.com/wss/2001_12/magazine/features/jlowy_2/

[71] Don Box
A Young Person's Guide to The Simple Object Access Protocol: SOAP Increases Interoperability Across Platforms and Languages
MSDN Magazine - March 2000
http://msdn.microsoft.com/msdnmag/issues/0300/soap/default.aspx

[72] OASIS Web Services Security (WSS) technical committee.
Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)
http://www.oasis-open.org/committees/documents.php?wg_abbrev=wss

[73] Jim Waldo
Remote procedure calls and Java Remote Method Invocation
IEEE IT Concurrency - Volume 6 Issue 3, Pages 5-7, July-September 1998

[74] Ann Wollrath, Jim Waldo and Roger Riggs
Java-centric distributed computing
IEEE Micro - Volume 17, Issue 3, Pages 44 – 53, May/Jun 1997

[75] Sun Microsystems
Java RMI over IIOP
http://java.sun.com/products/rmi-iiop/

[76] Arnold Buss and Leroy Jackson
Distributed simulation modeling: a comparison of HLA, CORBA, and RMI
Proceedings of the 1998 Winter Simulation Conference

[77] Sun Microsystems
Security and the Java Platform
http://java.sun.com/security/

[78] Sun Microsystems
Java Security Architecture
http://java.sun.com/j2se/1.5.0/docs/guide/security/spec/security-spec.doc1.html - 21150

[79] Sun Microsystems
Java Authentication and Authorisation Service (JAAS) Overview
http://java.sun.com/products/jaas/overview.html

[80] Tarak Modi
Jazz Up Java Security with JAAS
http://www.fawcette.com/archives/upload/free/Features/Javapro/2001/09sep01/tm0109/tm0109-1.asp

[81] C. Lai, L.Gong, L.Koved, A. Nadalin and R. Schemers
User authentication and authorisation in the Java ™ platform
IEEE Computer Security Applications Conference, 1999

[82] Sun Microsystems
JAAS Authorisation
http://java.sun.com/j2se/1.4.2/docs/guide/security/jgss/tutorials/AcnAndAzn.html

[83] Sun Microsystems
Java Cryptography Extension (JCE) Reference Guide
http://java.sun.com/j2se/1.5.0/docs/guide/security/jce/JCERefGuide.html

[84] Russell Meyers and Charles Frank
Implementing Your Own Cryptographic Provider Using the Java Cryptography Extension
Northern Kentucky University

[85] Sun Microsystems
Java Secure Socket Extension (JSSE) Reference Guide
http://java.sun.com/j2se/1.5.0/docs/guide/security/jsse/JSSERefGuide.html

[86] Raghavan Srinivas
Java security evolution and concepts
http://www.javaworld.com/javaworld/jw-05-2001/jw-0525-security-p3.html

[87] Scott Nevins
Database security: protecting sensitive and critical information
CEO of Protegrity

[88] Elisa Bertino and Ravi Sandhu
Database security - concepts, approaches, and challenges
IEEE Transactions on Dependable and Secure Computing - Volume 2, Issue 1, Pages 2 – 19, January-March 2005

[89] Rajiv Sinha
A Security Checklist for Oracle9i
An Oracle white paper - March 2001

[90] Oracle Corporation
Oracle 9i Advanced Security Release 2 - Data Sheet
http://www.oracle.com/technology/products/oracle9i/datasheets/advanced_security/aso_rel2.html

[91] Oracle Corporation
Oracle9i™ LogMiner™
Database Analysis Tool - Features Overview - January 2002

[92] Oracle Corporation
Oracle9i Fine-Grained Auditing
http://www.oracle.com/technology/products/oracle9i/daily/oct03.html

[93] Jonathan D. Moffett and John A. Clark
An Introduction to Security in Distributed Systems
High Integrity Systems Journal - 1(1): pages 83-92, 1994

[94] Jonathan D. Moffett
Security & Distributed Systems
Department of Computers, University of York, England

[95] Matthew Joyce
IC3-1 Network Seccurity
University of Bergen, NORWAY

[96] Jianxin Jeff Yan
A note on proactive password checking
ACM - Proceedings of the 2001 workshop on New security paradigms

[97] J.Yan, A. Blackwell, R. Anderson and A. Grant,
Password memorability and security: empirical results
IEEE Security & Privacy Magazine - Volume 2, Issue 5, Pages 25-31, September-October 2004

[98] Jianxin Yan, Alan Blackwell, Ross Anderson and Alasdair Grant
The Memorability and Security of Passwords Some Empirical Results
Cambridge University Computer Laboratory
http://download.lawr.ucdavis.edu/pub/CambridgePWStudy.pdf

[99] Theodore Stergiou, Mark S. Leeson, Roger J. Green:
An alternative architectural framework to the OSI security model
Elsevier - Computers & Security - 23, 137e153, 2004

[100] Newstaff, Inc.
Information Security References, ISO 17799
http://newstaff.info/criteria/iso17799/index.html

[101] ISO 17799 Information and Resource Portal
ISO 17799
http://17799.denialinfo.com/

[102] Biju Mukund
ISO 17799 Information and Resource Portal
http://17799.denialinfo.com/biju.htm

[103] John Musser and Paul Feuer
All that JAAS
http://www.javaworld.com/javaworld/jw-09-2002/jw-0913-jaas.html

[104] Apache Software Foundation
SSL Configuration HOW-TO
http://tomcat.apache.org/tomcat-4.0-doc/ssl-howto.html

[105] IBM Redbooks
IBM WebSphere V5.1 Performance, Scalability and High Availability
June 2004

[106] Sun Microsystems
Keytool - Key and Certificate Management Tool
http://java.sun.com/j2se/1.5.0/docs/tooldocs/solaris/keytool.html

[107] Mahfuzur Rahman and Prabir Bhattacharya
Secure Network Communication using Biometrics
IEEE International Conference on Multimedia and Expo (ICME'01)

[108] Gregory R. Doddrell
Information security and the Internet
Internet Research: Electronic Networking Applications and Policy
Volume 6, Number 1, Pages 5–9, 1996

[109] META Group
The Evolution of Network Security: From DMZ Designs to Devices
May 2004
http://www.juniper.net/solutions/literature/white_papers/200084.pdf

Appendices

Appendix I

The following paper '*Perception: The Real Inhibitor to ASP Adoption?*' was presented by the author at the 2005 Information Technology & Telecommunications (IT&T) Conference on *'Information Infrastructures – Systems Architecture and Content'*. The conference was held the 26th-27th of October 2005, at the National Maritime College, Cork Institute of Technology. The paper was published on pages 187-195 of the conference proceedings.

Perception: The Real Inhibitor to ASP Adoption?

Sabrina McNeely[1], Kenneth Kirrane[2], John Healy[3], Sean Duignan[4]
Department of Maths & Computing,
Galway-Mayo Institute of Technology,
Galway, Ireland.
Tel: +353 (0) 91 753161
E-mail[1]: sabrina.mcneely@gmit.ie
E-mail[2]: kenneth.kirrane@gmit.ie
E-mail[3]: john.healy@gmit.ie
E-mail[4]: sean.duignan@gmit.ie

Abstract: Upon its inception, many heralded the ASP paradigm as the death knell of software-as-a-product, and the birth of software-as-a-service. Despite the hype however, uptake is struggling to reach the levels many analysts predicted for the ASP model. This paper identifies and examines the key factors influencing the adoption of ASP, and highlights the inconsistencies in the available literature. We identify several questions that remain unanswered, which may be adversely influencing user perceptions of the model. In order to address these questions, areas of further research are proposed to deconstruct the inhibitors of the ASP paradigm, and ultimately answer the most burning question: *Is perception the primary inhibitor to the uptake of the ASP model?*

1. Introduction

The Application Service Provision (ASP) model has had many guises over the years including software-as-a-service, on-demand computing and utility computing. However, its underlying premise remains unchanged: Application Service Providers (ASPs) offer multiple users a subscription-based access model via the Internet to centrally managed applications [1]. ASPs provide access to software on a one-to-many basis and thus the cost of ownership and maintenance of the solution is shared by several clients. Service level agreements (SLAs) assist in ensuring client expectations are met with regard to the performance of the ASP solution.

Despite the initial hype, user uptake of the ASP model has been slow to materialise. In 2001 the International Data Corporation (IDC) Group forecast that

spending on ASPs would grow to $24 billion by 2005 [2]. By 2002/2003, the ASP market seemed all but dead, with a 90 percent failure rate according to industry analysis [3]. IDC reports that the ASP market had only reached $5 billion by 2003 falling far short of that which was first envisaged [4]. Current estimates indicate worldwide spending on software as a service and associated software license revenue will reach $15.2 billion by 2007, much lower than earlier predictions but substantial nonetheless [4] [5]. The above statistics suggest that ASP has been given a new lease of life.

Although many papers are quick to quote statistics and outline the determinants of ASP adoption, few delve into the reasoning behind these determinants, be they positive or negative. The objective of this paper is to identify and examine the key factors influencing the adoption of the ASP model. To accomplish this objective, this paper explores the existing literature in order to formulate a consensus on the reasons pertaining to the uptake or otherwise of the ASP model. Our analysis is divided into three distinct sections. First, we establish the key factors that individually lead to either an affirmative or adverse decision with regard to the uptake of the ASP model. Second, we attempt to expand on the research to date by examining each of these factors and their relevance to the adoption of the ASP as a whole. Finally, we conclude the paper by highlighting the required direction for further research of the ASP model.

2. Identification of the key factors influencing ASP adoption

In depth analysis of the literature highlights economies of scale as the key driver of the ASP paradigm [1], [6], [7], [8], [9], [10]. ASPs exhibit economies of scale as the cost of the solution is distributed among its customers on a one-to-many basis. In addition to economies of scale, Walsh [10] interestingly highlights security and reliability as major benefits of the ASP model, and states that for small or midsize organisations, ASPs can provide greater levels of security and reliability than the customers own organisation [10]. Walsh's point is contrary to the norm, as uncertainties with regard to security and privacy as well as performance concerns in the form of availability, scalability and reliability, are cited as the main inhibitors to the uptake of the ASP model [5], [11], [12].

This section is dedicated to examining each of the above factors in order to assess the benefits or threats they potentially pose to the adoption of the ASP model.

2.1 Economies of scale

In economic terms, economies of scale are achieved when the average cost of producing a product diminishes as each additional product is produced, as the fixed costs are shared over an increasing number of products. The economies of scale model is equally viable with regard to service provisioning. ASPs achieve economies of scale by lowering the average cost of the service through sharing fixed costs among many users. A survey conducted by the Information Technology Association of America (ITAA) in 2002 investigated key user expectations in selecting ASPs. Results of that survey indicate that 39 percent of respondents estimate their return on investment of between 10 and 50 percent, while an additional 14 percent of respondents placed it between 51 and 100 percent [13]. The cost savings highlighted by the ITAA survey offer some solace to companies burdened by an increased reliance on IT, and its associated costs. ASPs can alleviate this burden, thus allowing a company to focus on other core areas of their enterprise. By contrast, a survey of 250 IT managers conducted by *Informationweek.com* highlights a high degree of scepticism with regard to the claimed cost advantages / economies of scale [14].

2.2 Performance (high availability, scalability and reliability)

Due to ASPs network-centric delivery model, performance considerations in terms of high-availability, scalability and reliability are often cited as the major inhibitors to the widespread uptake of the ASP model [5], [11], [12], [15]. High Availability (HA) requires systems designed to tolerate faults – to detect a fault, report it, mask it, and then continue service while the faulty component is repaired offline [16]. In the majority of cases, availability is expressed as a percentage of system up time, with "five nines" or 99.999% availability a desired level of availability for most ASPs. Scalability refers to the ability of a system to accommodate an increasing number of elements or objects, to process growing volumes of work gracefully, and/or to be susceptible to enlargement [17]. Systems should not only adapt to their new configurations, they should be able to

operate with the same level of efficiency and to the same standard of service. Reliability is defined as the assurance a product will perform its intended function for the required duration within a given environment. Reliability is best described as product performance over time [18]. A reliable system should consistently produce the same results, while meeting, or exceeding customer expectations.

Kern et al (2002) report that 85% of potential ASP customers rate quality of service as being one of the key factors in ASP satisfaction. The majority of potential ASP customers also rate scalability and flexibility as being very important [19]. A large factor in service quality, availability and scalability, for web-hosted applications is the quality and speed of the underlying network in delivering the service offering to its customers. Many factors influence network quality, such as bandwidth limitations, network latency and reliability of the Internet. This is especially true in the ASP paradigm as all the application processing takes place on the application server, with the results returned to geographically dispersed users over the network in a thin-client model. These findings are corroborated by ITAA's (2001) survey of key user expectations with respect to ASP – over 80% of respondents claimed that guarantees on network reliability were a very important feature of Service Level Agreements (SLAs) between ASP and clients [2].

Availability and performance are probably two of the most important characteristics of an ASP. Consequently, ASPs will generally invest heavily in backup and redundant systems in order to minimise service disruption. Walsh notes that these safeguards go beyond what many small to midsize companies can afford, and are thus seen as a benefit of the ASP model [6]. Tao also suggests that most online service providers do a better job of ensuring 24/7 application availability than customers could [20]. Walsh and Tao's position is further strengthened by various other references in the literature pertaining to availability, scalability and reliability as benefits of the ASP model [8], [11], [21], [22].

2.3 Security

Several researchers refer to security and privacy of data as one of the primary areas for concern with regard to the realisation of an ASP solution [6], [8], [5], [12], [23]. Fears of compromised security and privacy have prevented many firms from fully investigating

and integrating the ASP business model [12]. Although both security and privacy are concerned with guarding the clients sensitive data, they can be distinguished as follows:

- Security is used to refer to protection of the ASP solution and the data exchanged or stored as part of the ASP solution. ASP security can be broken into three distinct considerations: physical security, solution security and security and integrity of client data [24].
- Privacy is exclusively concerned with ensuring the protection and integrity of the client data exchanged or stored as part of the ASP solution from unauthorised access.

Linthicum [23] outlines three possible security issues, which can be used to collaborate the above definition of security. Poor network security may leave the hosted solution open to external intrusion. Second, an unsatisfactory physical security policy may result in an internal attack. Finally, there are concerns around the security firewalls that are placed between the hosted application domains [23].

While the majority of research literature focus on the negative aspect of security, Walsh [10] looks at security from a different perspective concentrating on the security benefits that can be leveraged from an ASP solution. ASPs are responsible for defining and adhering to a security policy, which meets the needs of their clients. Walsh [10] states that often the security and reliability safeguards implemented by ASPs go beyond what many small to midsize companies can afford and thus are a benefit of the ASP model.

3. Analysis of the key factors influencing ASP adoption

Based on an in-depth analysis of the available literature, *Economies of Scale, Performance* and *Security* have been identified as the key factors that influence the uptake of the ASP model. The aim of this section is to expand on the research to date by examining each of these factors and to assess their relevance to the success of the ASP model in greater detail.

By operating a one to many business model ASPs can achieve economies of scale in terms of applications, network costs, server technology and implementation expertise [8]. It is argued however that clients who demand a high degree of customisation destroy

much of the value that economies of scale provide [25]. This results in the need to pay higher fees for customised solutions. *Do client requirements for customisation need to adversely affect the benefits obtained through economies of scale?* If the ASP offering is based on open standards, then high levels of customisation may not equate to higher costs. For example, many dedicated concert and entertainment venues see economies of scale inherent in outsourcing their ticketing operations, an illustration that customisation can be accommodated within a centralised environment – the ticketing solution provided by "ASPs" can be customised with regards to venue, artist, date, etc, while the underlying service offering remains the same.

Much of the literature suggests that performance considerations (and in particular the issues of high-availability, scalability and reliability) are significant inhibitors to ASP adoption. Notwithstanding that, other researchers suggest that given the sizeable investments undertaken by ASP solution providers, the ASP model may actually offer enhanced availability, scalability and reliability to the solution adopter. Indeed, many mature and industry proven solutions can be cited that offer support to this notion. Hewlett Packard's flagship version of the popular Unix operating system (HP-UX 11i) is a case in point. Like many other product offerings in the marketplace, this operating system is implemented on systems ranging from workstations and access servers to application servers and data center servers - systems where high availability is of paramount importance. HP-UX 11i scales easily to 64 processors and is designed to allow for future scaling to 256 processors in a single system [26]. Other players in the operating systems marketplace provide similar functionality in their products. Sun Microsystems, for instance, offers Sun Fire E25K Server; a massively scalable, highly available data center server that scales to 72 UltraSPARC IV processors. A key factor in the design of the Sun Fire E25K Server is the ability to consistently deliver high levels of reliability and availability [27]. *Why then is performance perceived as an inhibitor to ASP uptake, when industry proven solutions exist that support high-availability, scalability and reliability?*

Time and time again security and privacy are cited as major drawbacks to the uptake of the ASP model. As previously outlined by Walsh [10], ASPs often have the ability and resources to provide a higher level of security than many small to midsize

companies [10]. The above begs the question: *Is trust the key to viewing security as a benefit or a threat to the uptake of the ASP model?* In an attempt to answer this question we examine security and specifically consider physical security, solution security and security and integrity of client data.

Information Technology (IT) organisations have trusted data centers with the security of their solution hardware for decades. Data centers have gained customer trust by implementing strict physical security policies, ensuring access is restricted to authorised personnel through the use of biometric scanners such as fingerprint or IRIS identification, in addition to the use of passwords and armed guard protection of facilities [6]. *Surely ASPs have a vested interest in ensuring the physical security of their hardware?* Authentication, authorisation and encryption all fall under the solution security umbrella. Organisations have made signification progress in securing systems through authentication by enforcing the use of strong passwords. Restriction of access (both local and network) is achieved through authorisation policies. Advances in network security have alleviated the fear of transferring sensitive data such as bank and credit card details. HyperText Transfer Protocol over Secure Sockets Layer (HTTPS) ensures the privacy, integrity and consistency of data through the use of the encryption. Secure Sockets Layer (SSL) has been widely implemented and is now the de facto standard for providing secure e-commerce transactions over the web [28]. *Are ASPs not equally dedicated to preventing unauthorised access to machines on their network as their clients?* Although appropriate levels of physical and solution security assist in ensuring security and privacy of data it is also essential that proven encryption techniques are used and redundant hardware is disposed of in an appropriate manner. Software and data that is no longer needed should be uninstalled and erased to ensure that it is not accessible to unauthorised individuals. Yet again by implementing an appropriate security policy ASPs could overcome this perceived problem. *Surely ASPs strive to meet or even exceed customer expectations?*

Economies of scale, *Performance* and *Security* have been identified as the key factors that influence the uptake of the ASP model. However our analysis of these factors suggests than 'perception' may in fact be the factor most relevant to the success of the

ASP model. *Is it possible that 'bad press' has influenced perception of the ASP model and ultimately its uptake?*

4. Conclusion

Our analysis of the factors influencing the uptake of the ASP model suggests that the research conducted to date is at best, incomplete, or at worst, vague and ambiguous. Further research is required to fully clarify the relevance of Economies of Scale, Performance and Security, and most importantly *Stakeholder Perceptions* (Users, IT Managers, Software Developers, Industry Analysts and Academics) to the adoption of the ASP model as a whole. *Perceptions* have in the past greatly influenced the emergence or otherwise of new paradigms and / or the rate of adoption of new products. Global adoption of the automobile for instance was initially predicted to be in the low thousands due to the *fact* that not enough people would work as chauffeurs. History tells us that this *fact* was in the end an ill-informed and poorly validated *perception*. Closer to the world of ASP there are similar examples of perceptions influencing critical thinking – consider some of the IT industry's view of the potential market for personal computing some 20 - 25 years ago! We suggest that *perception* is in fact a key inhibitor to the uptake of the ASP model and seek to further this hypothesis. *Highlighting* the issue through this paper is an initial step. Furthering the body of research focusing on ASP adoption is another step in assessing our hypothesis. To begin to achieve this second step then, we propose a survey of IT organisations that specifically explores all the factors pertinent to ASP adoption at a much finer level of granularity. The primary focus of such a survey is to explore "perceptions" in particular and how they influence ASP adoption rates. Interviews should supplement the survey where appropriate to clarify any ambiguities that arise. It is intended to finalise the design of the survey and target a representative population in Ireland in 2005. An analysis of the research methodology and of the survey results will be the subject of a later paper.

5. References

[1] T. Kern et al – 2002,
"Exploring ASP as sourcing strategy: theoretical perspectives, propositions for practice", *Journal of Strategic Information Systems 11 (2002) 153-177*

[2] A. Suasaria et al – 2001,
"Making the Most out of an ASP Relationship", *IEEE Computer Society* http://csdl.computer.org/comp/mags/it/2001/06/f6063abs.htm, 7th March 2005

[3] L. Haber – 2004,
"ASPs Still Alive and Kicking", *ASPnews.com – Trends* http://www.aspnews.com/trends/article.php/3306221, 7th December 2005

[4] A. M. Konary - 2004,
"Presentation to the ITAA", IDC Analyst Presentation to the ITAA - February 12, 2004, http://www.IDC.com

[5] N. Daylami et al – 2005,
"Determinants of Application Service Provider (ASP) Adoption as an Innovation", *Proceedings of the 38th Hawaii International Conference on System Sciences (HICSS-38 2005)*

[6] K. R. Walsh – 2003,
"Analysing the Application ASP Concept: Technologies, Economies, and Strategies", *Communications of the ACM, August 2003 Vol. 46, No. 8*

[7] B. Desai and W. L. Currie – 2003,
"Application Service Providers: A model in Evolution", *The Fifth International Conference on Electronic Commerce, ICEC 2003*

[8] T. Kern and J. Kreijger – 2001,
"An Exploration of the Application Service Provision Outsourcing Option", *Proceedings of the 34th Hawaii International Conference on System Sciences (HICSS-34 2001)*

[9] B. Vassiliadis et al – 2004,
"Application Service Provision through the Grid: Business models and Architectures", *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)*

[10] K. R. Walsh – 2003,
"Analysing the Application ASP Concept: Technologies, Economies, and Strategies", *Communications of the ACM, August 2003 Vol. 46, No. 8*

[11] B. Jaruzelski et al – 2001,
"ASP 101: Understanding the Application Service Provider Model", *Booz, Allen, Hamilton Consulting,*
http://extfile.bah.com/livelink/livelink/61813/?func=doc.Fetch&nodeid=61813, 11th March 2005

[12] B. Desai – 2002,
"Market Entry Strategies of Application Service Providers: Identifying Strategic Differentiation", *Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS-36 2003).*

[13] T. Burton – 2002,
"ITAA Survey Shows ASP Customers Achieve Real Benefits from Outsourcing", *Information Technology Association of America,*
http://www.itaa.org/news/pr/PressRelease.cfm?ReleaseID=1017252264, 8th March 2005

[14] J. Mateyaschuk - 1999,
"Leave the Apps to us! ASPs offer benefits through economies of scale", *Informationweek.com,*
http://www.informationweek.com/756/asp.htm, 25th April 2005

[15] D. Greschler and T. Mangan – 2002,
"Networking lessons in delivering 'Software as a Service' – Part 1", *International Journal of Network Management 2002; 12: 317 (DOI: 10.1002/nem.446)*

[17] A. B. Bondi – 2000,
"Characteristics of Scalability and Their Impact on Performance", *Proceedings of the 2nd International Workshop on Software and Performance*

[16] J. Gray and D. P. Siewirok – 1991,
"High-Availability Computer Systems", *IEEE Computer archive Volume 24, Issue 9*

[18] IEEE Reliability Society – 2000,
"Reliability Engineering", *IEEE Reliability Society,*
http://www.ewh.ieee.org/soc/rs/Reliability_Engineering/index.html, 25th April 2005

[19] T. Kern et al – 2002,
"Netsourcing: Renting Business Applications and Services Over a Network", *ISBN: 0-13-092355-9, Pretence Hall Publishing Inc.*

[20] L. Tao – 2001,
"Shifting Paradigms with the Application Service Provider Model", *IEEE Computer Society,*
http://csdl.computer.org/comp/mags/co/2001/10/rx032abs.htm, 08th Dec 2004

[21] B. Cohen – 2004,
"Smart Advice: Consider Purchasing Real-Time Collaboration Applications through an ASP", *InformationWeek Advisory Council,*
http://www.informationweek.com/shared/printableArticleSrc.jhtml?articleID=55800253, 25th April 2005

[22] D. Sovie and J. Hanson – 2001,
"Application Service Providers: Where are the real profit zones? ", *Mercer Management Consulting,*
http://www.mercermc.com/Perspectives/Perspectives_pdfs/hanson_sovie-ASPprofit.pdf, 20th January 2005

[23] D. Linthicum – *2002,*
"To ASP or Not to ASP?",
http://www.softwaremag.com/L.cfm?Doc=archive/2000apr/ASPorNot.html, 27th April 2005

[24] T. Anderson – 1994,
*"*Management Guidelines for PC Security*", ACM SIGICE Bulletin - July 1994 - Volume 20, Issue 1, Pages 7-14*

[25] P. Bendor-Samuel – 2001,
"Maximising the Benefits of Economies of Scale", *Outsourcing Journal Insights*, Outsourcing Journal May 2001

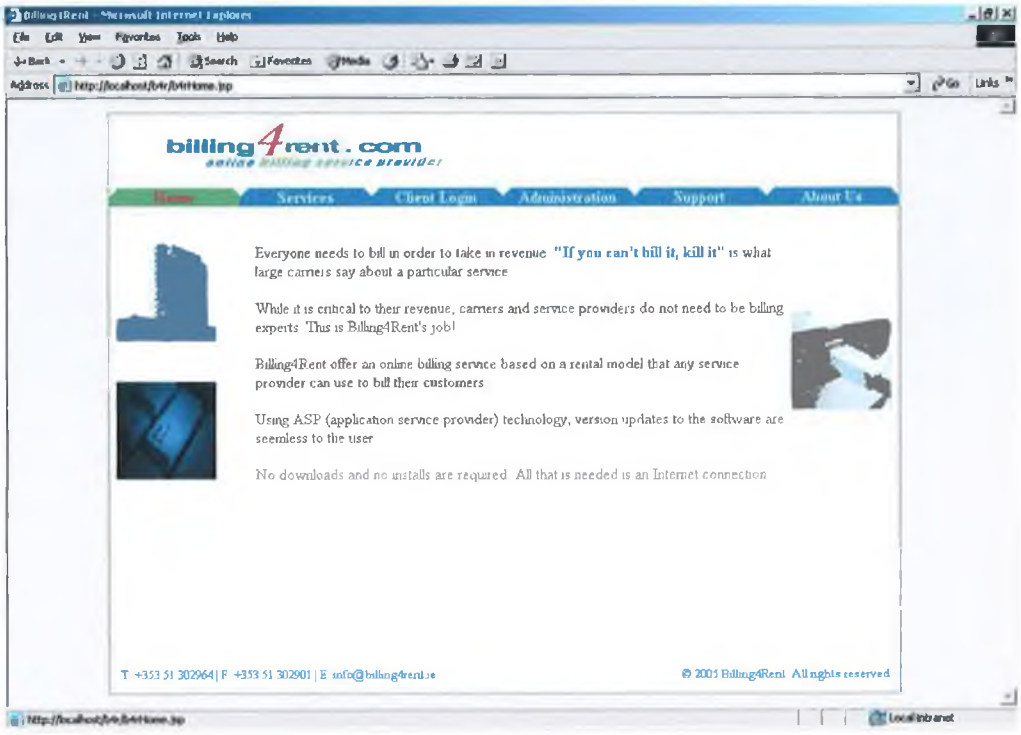
[26] Hewlett-Packard – 2005,
"HP-UX 11i, the Proven Foundation for the Adaptive Enterprise", *Hewlett-Packard,*
http://www.hp.com/products1/unix/operating/index.html, 2nd May 2005

[27] Sun Microsystems – 2005,
"HP-UX 11i, the Proven Foundation for the Adaptive Enterprise", *Sun High End Servers,*
http://www.sun.com/servers/highend, 2nd May 2005

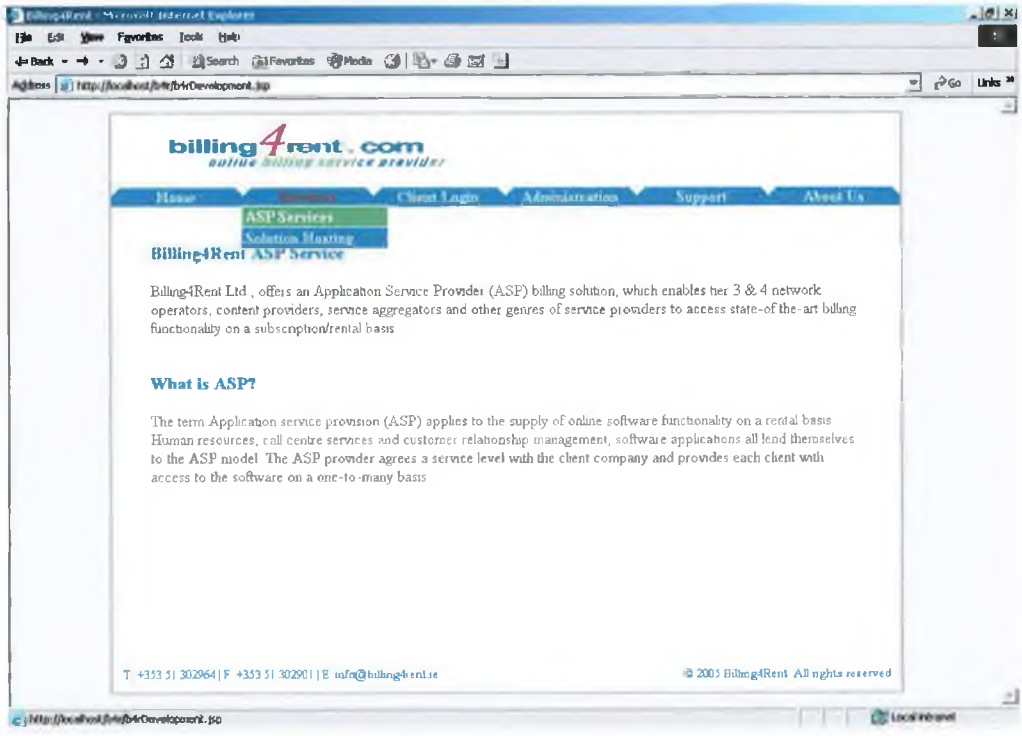
[28] W. Chou
"Inside SSL: The Secure Sockets Layer Protocol", *IT Professional – July/August 2002 Vol. 4, No. 4 Pages* 47-52

Appendix II

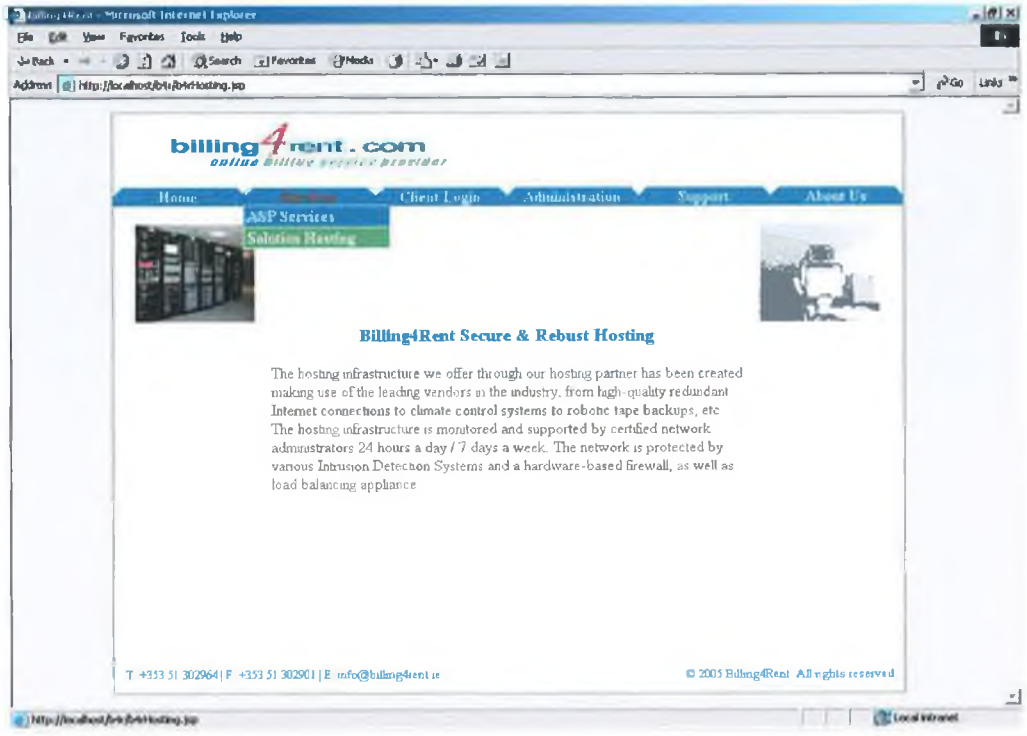
The following screenshots were taken from the Application Service Provision (ASP) billing prototype case study, formally known as Billing4Rent. The prototype is subdivided into three interconnected applications: the Billing4Rent Ltd. web application, the billing solution and an administration tool.



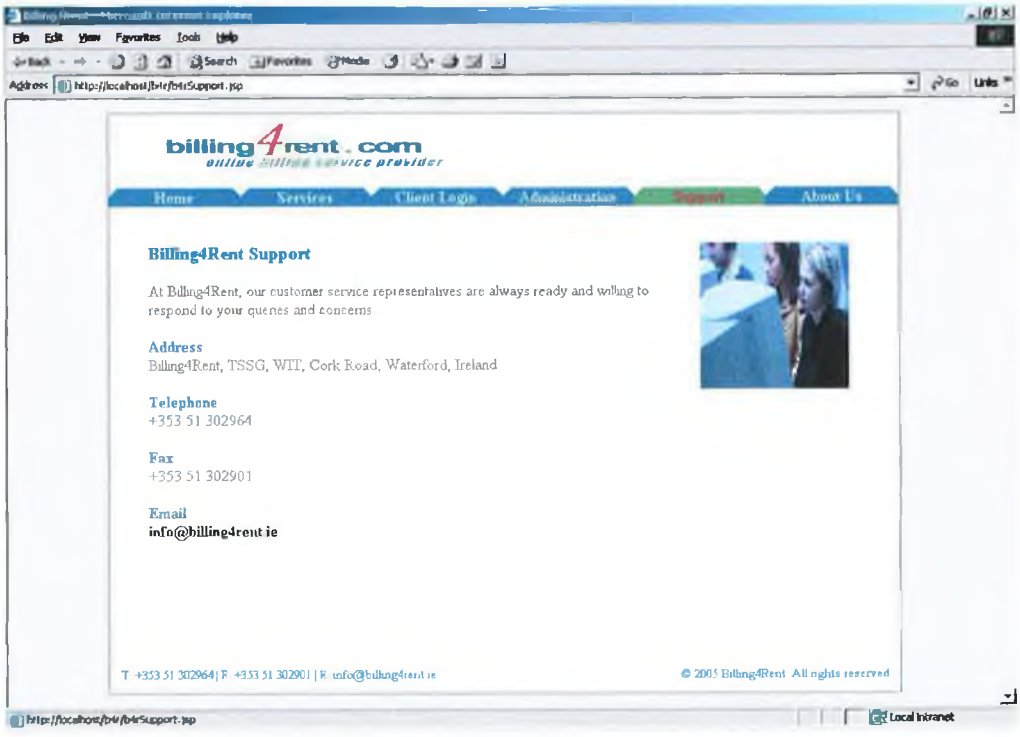
Billing4Rent web application 'Home' page



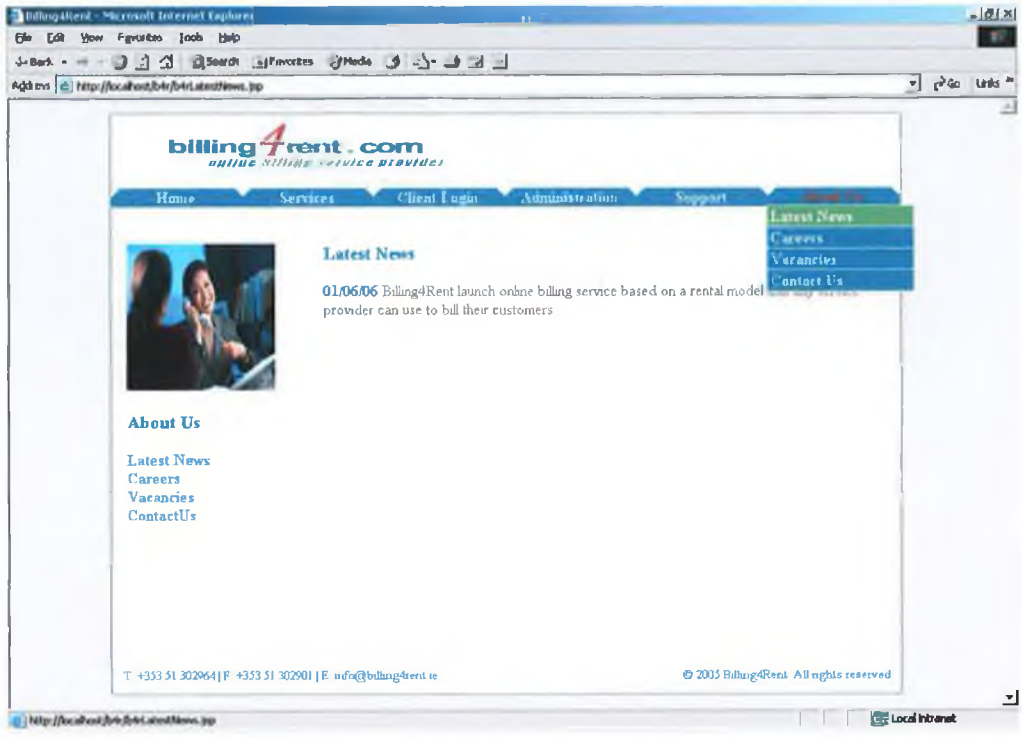
Billing4Rent web application 'ASP Services' page



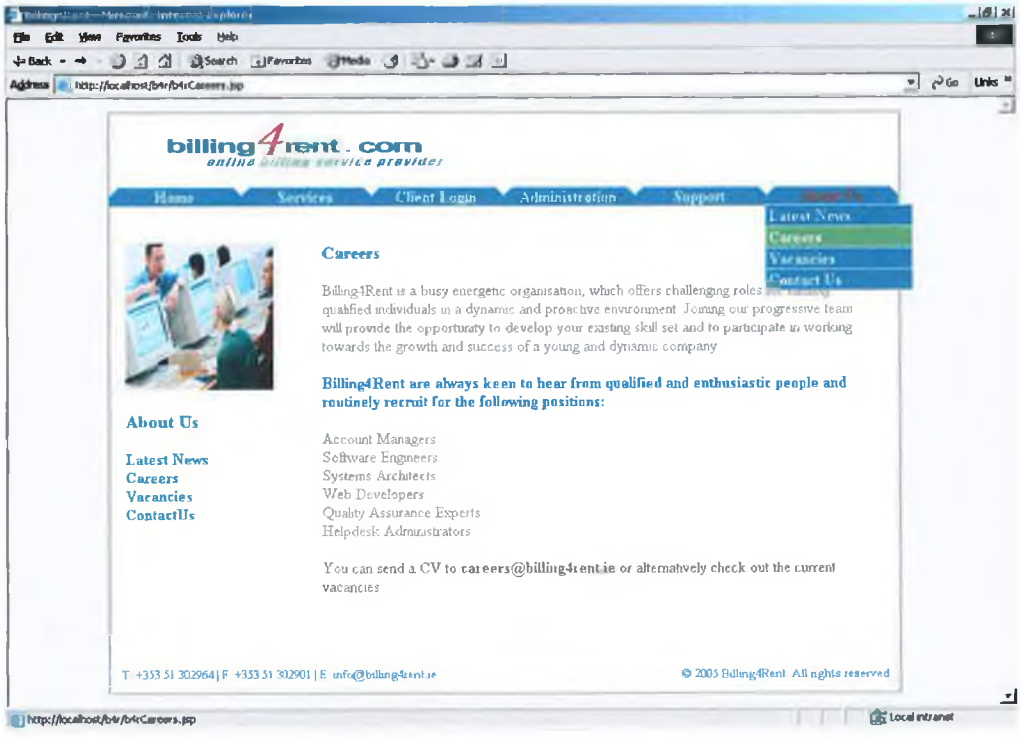
Billing4Rent web application 'Solution Hosting' page



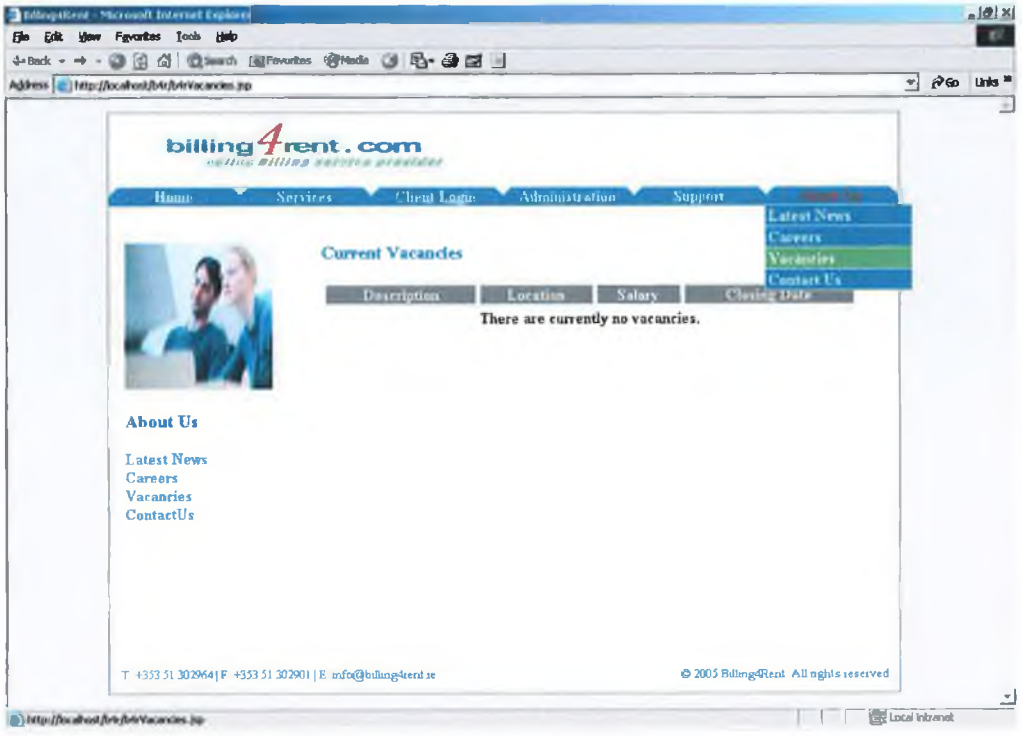
Billing4Rent web application 'Support' page



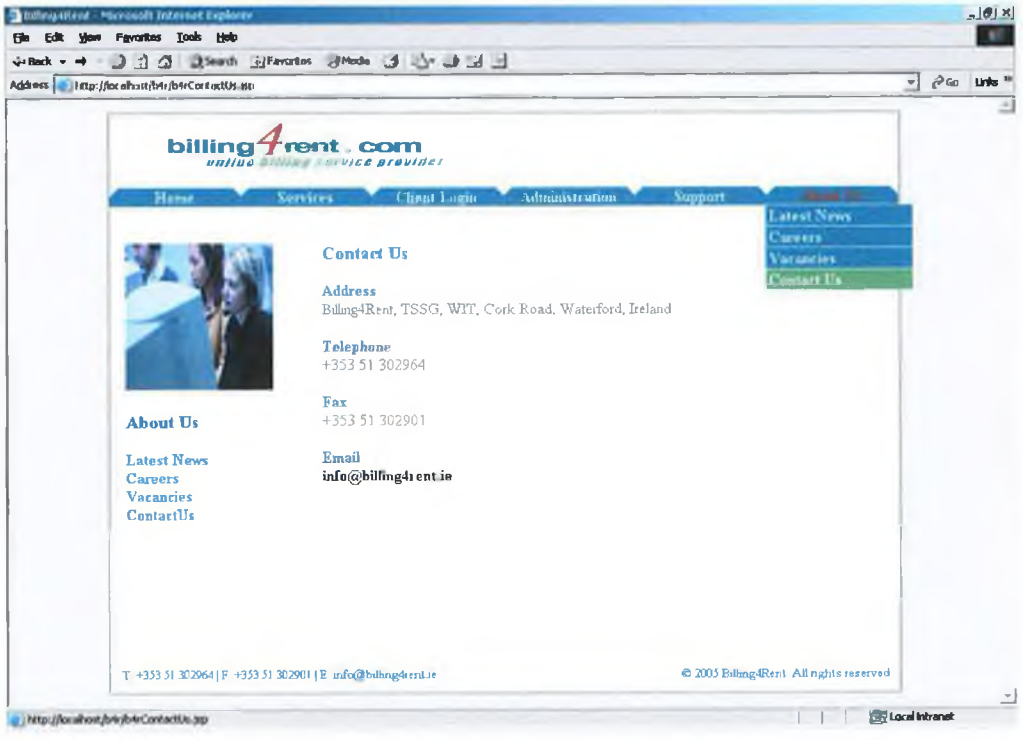
Billing4Rent web application 'Latest News' page



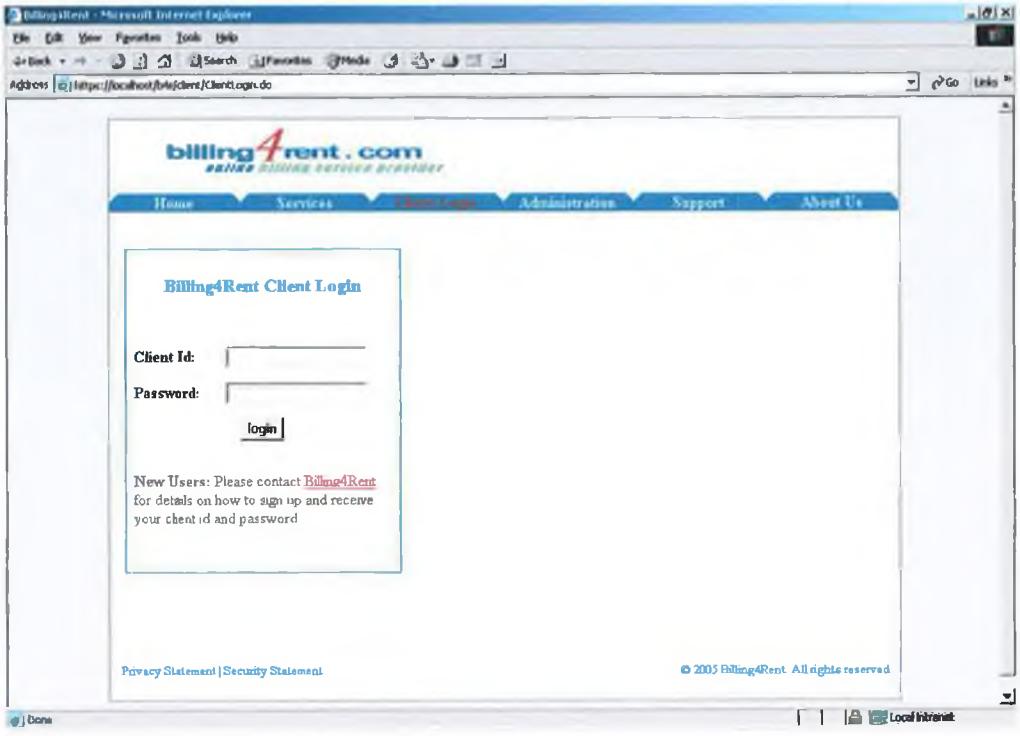
Billing4Rent web application 'Careers' page



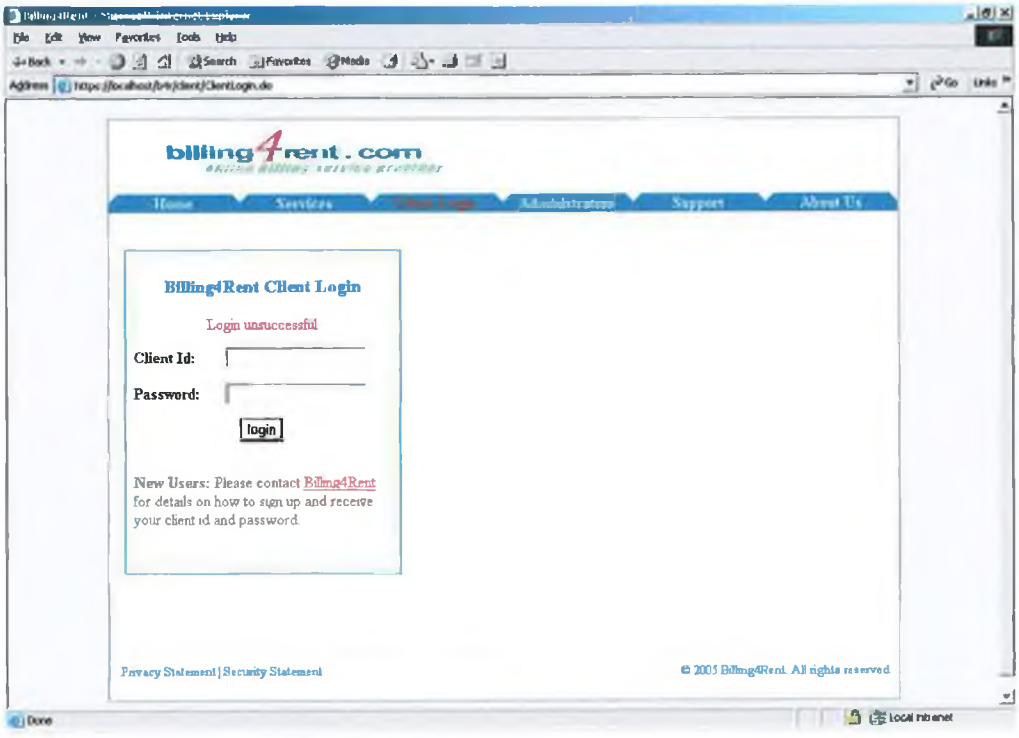
Billing4Rent web application 'Vacancies' page



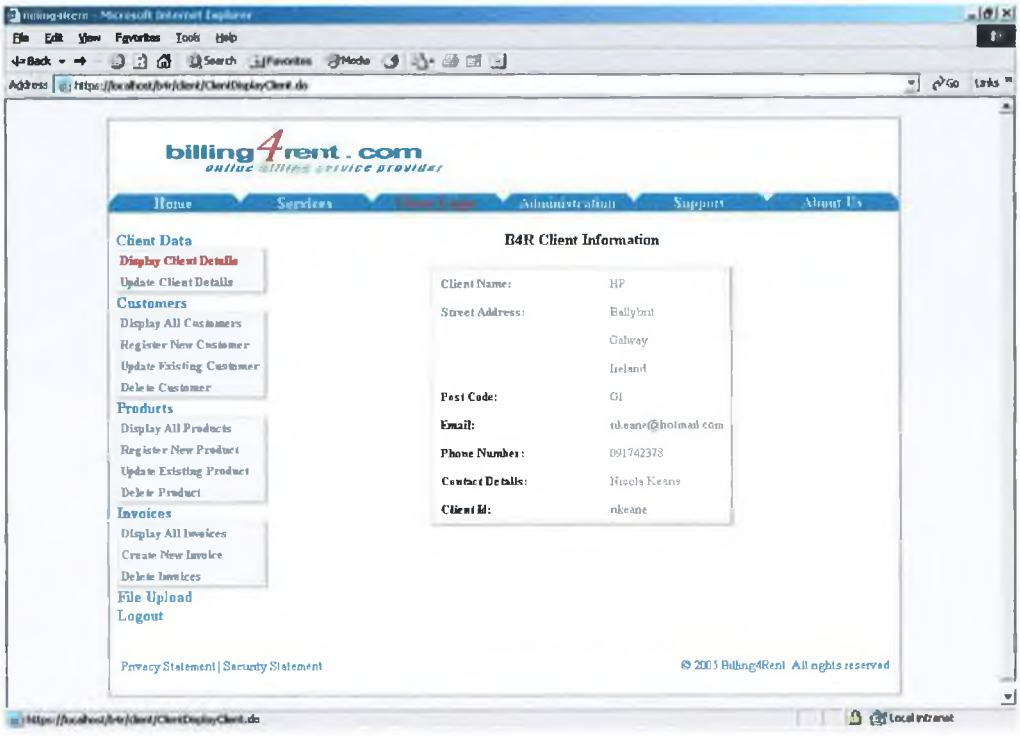
Billing4Rent web application 'Contact Us' page



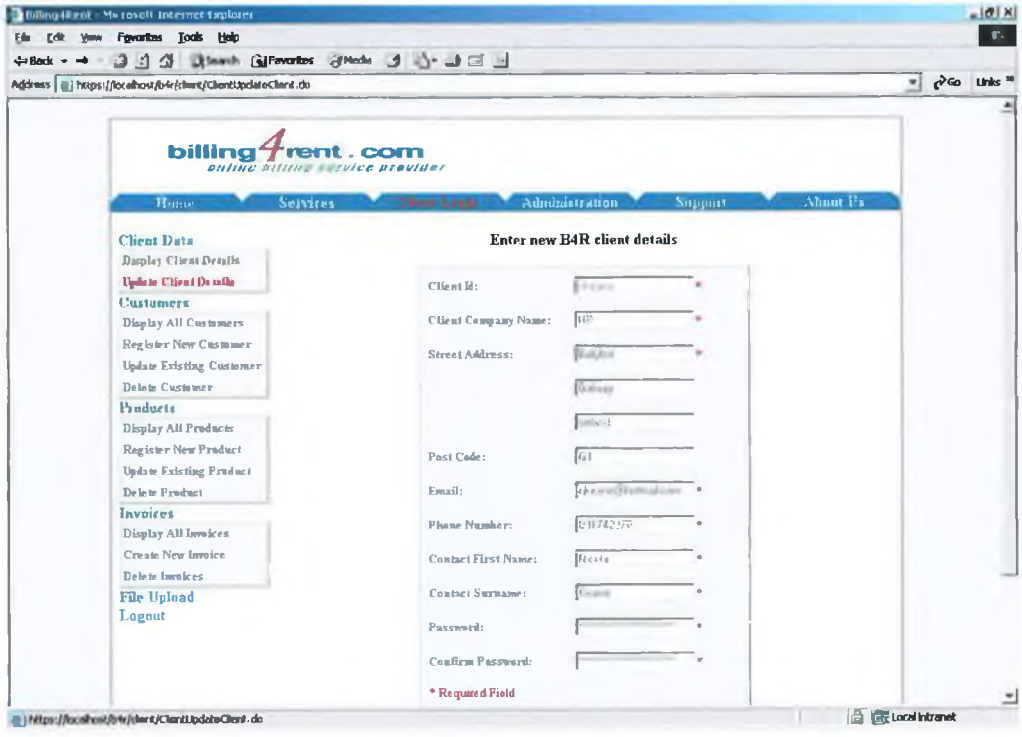
Billing4Rent billing solution 'Client Login' page



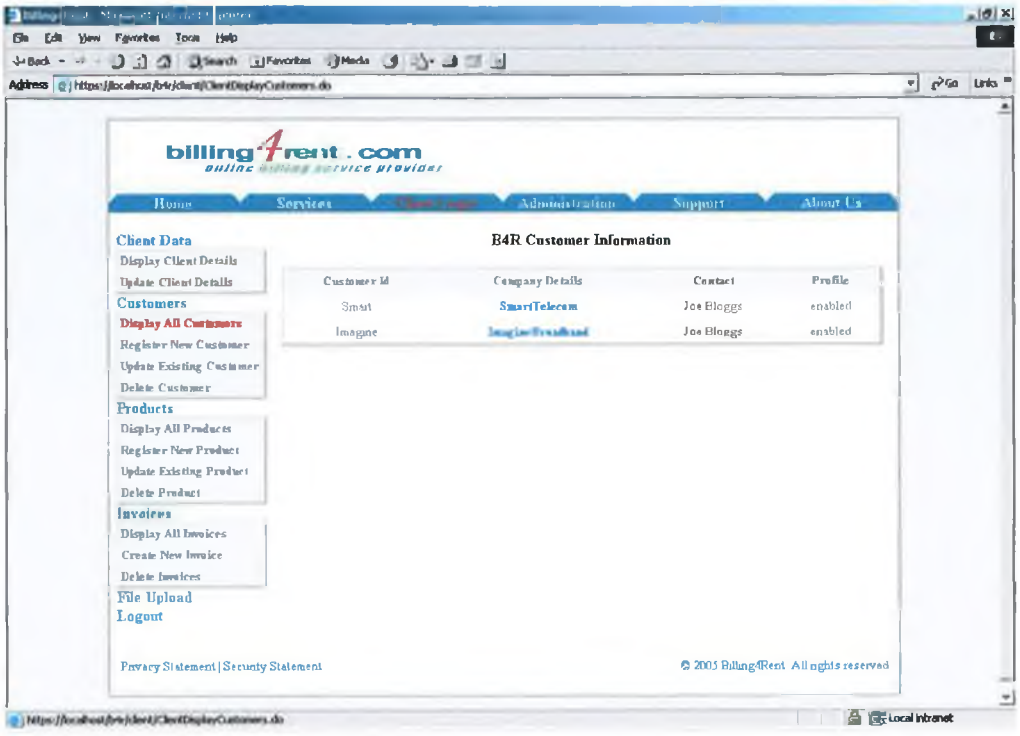
Billing4Rent billing solution 'Login unsuccessful' page



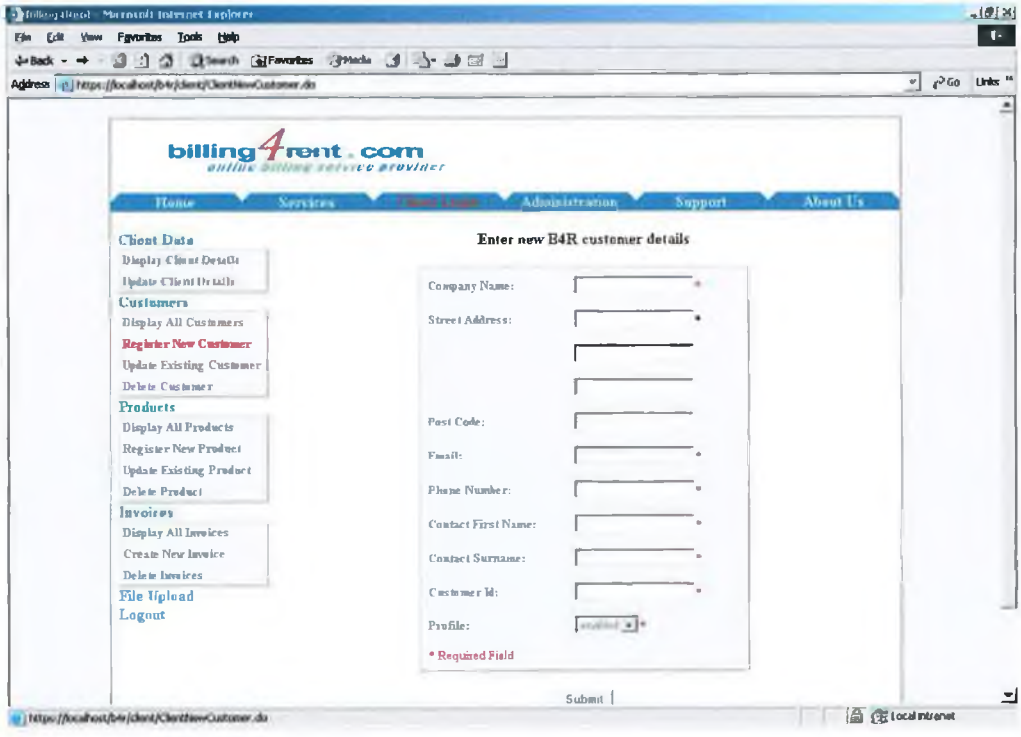
Billing4Rent billing solution 'Display Client Details' page



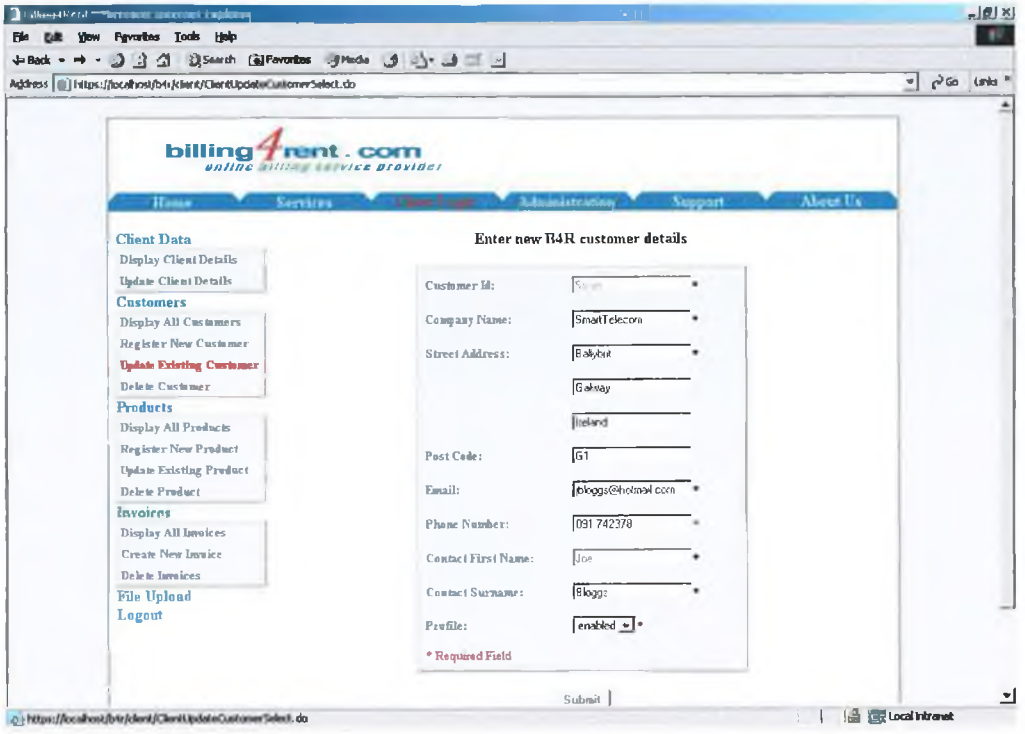
Billing4Rent billing solution 'Update Client Details' page



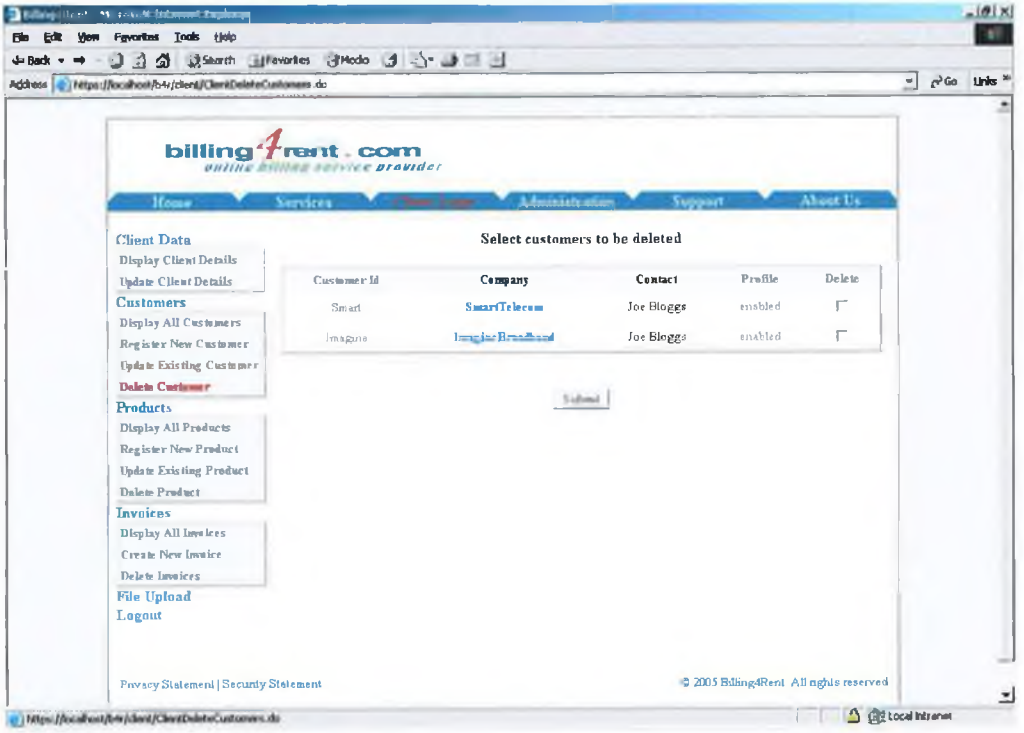
Billing4Rent billing solution 'Display All Customers' page



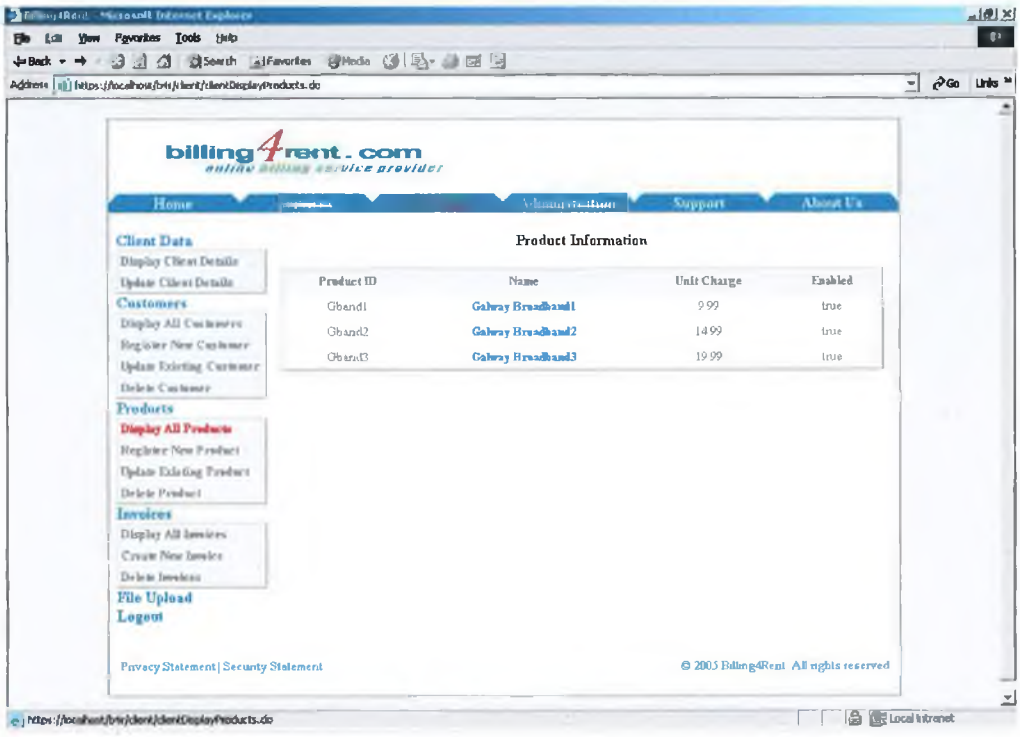
Billing4Rent billing solution 'Register New Customer' page



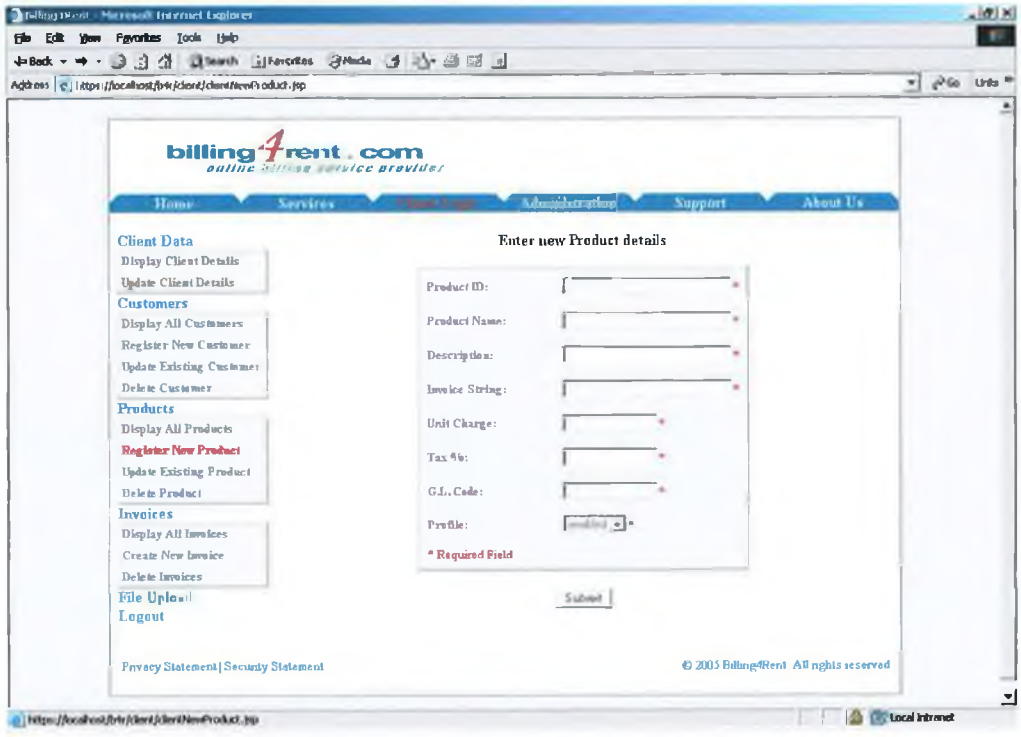
Billing4Rent billing solution 'Register New Customer' page



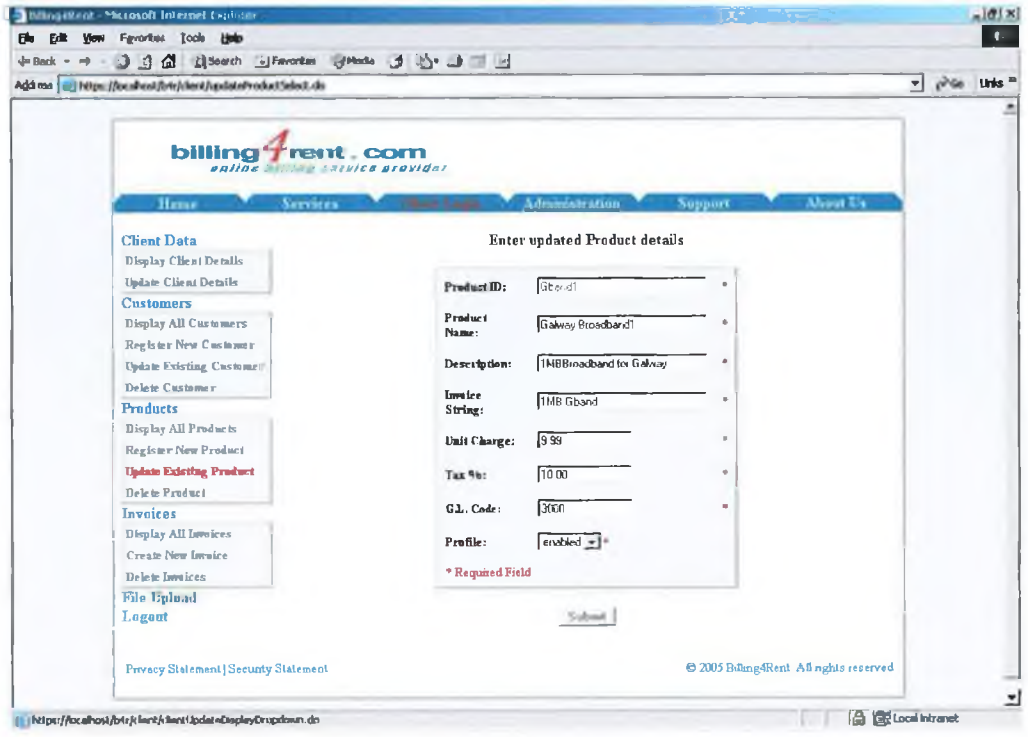
Billing4Rent billing solution 'Delete Customer' page



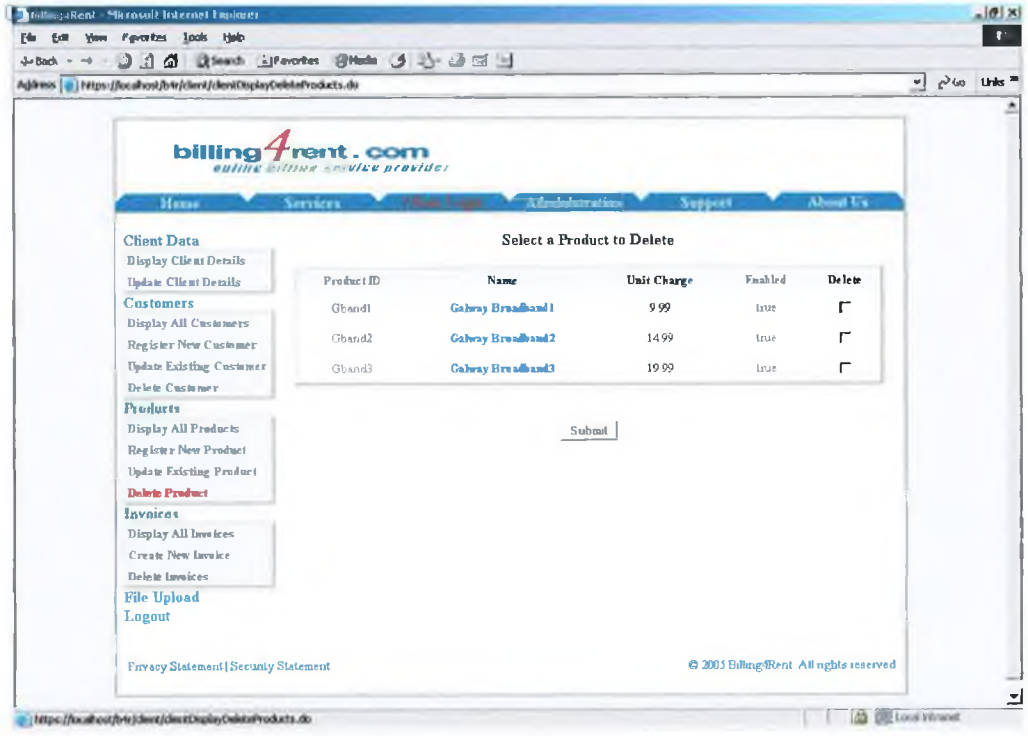
Billing4Rent billing solution 'Display All Products' page



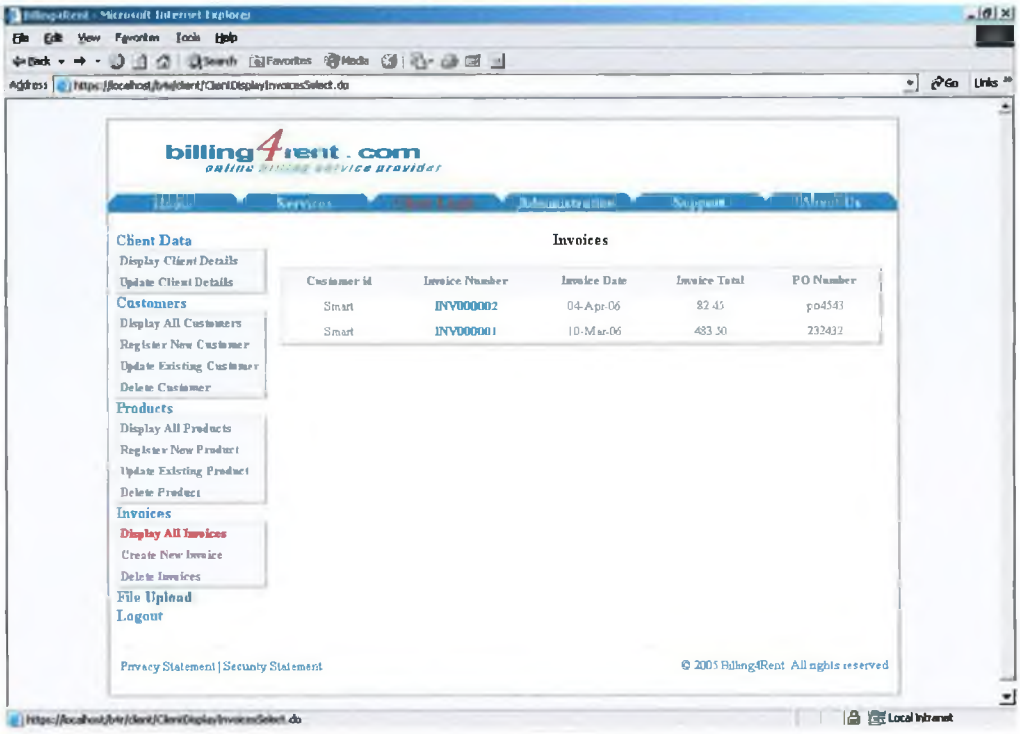
Billing4Rent billing solution 'Register New Product' page



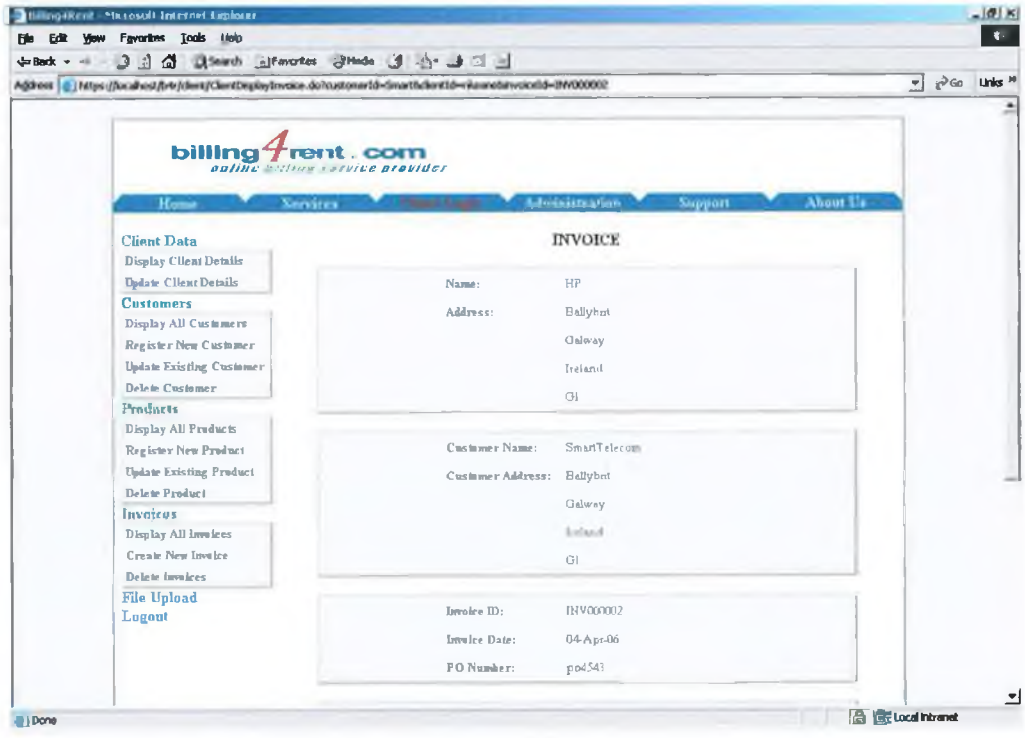
Billing4Rent billing solution 'Update Existing Product' page



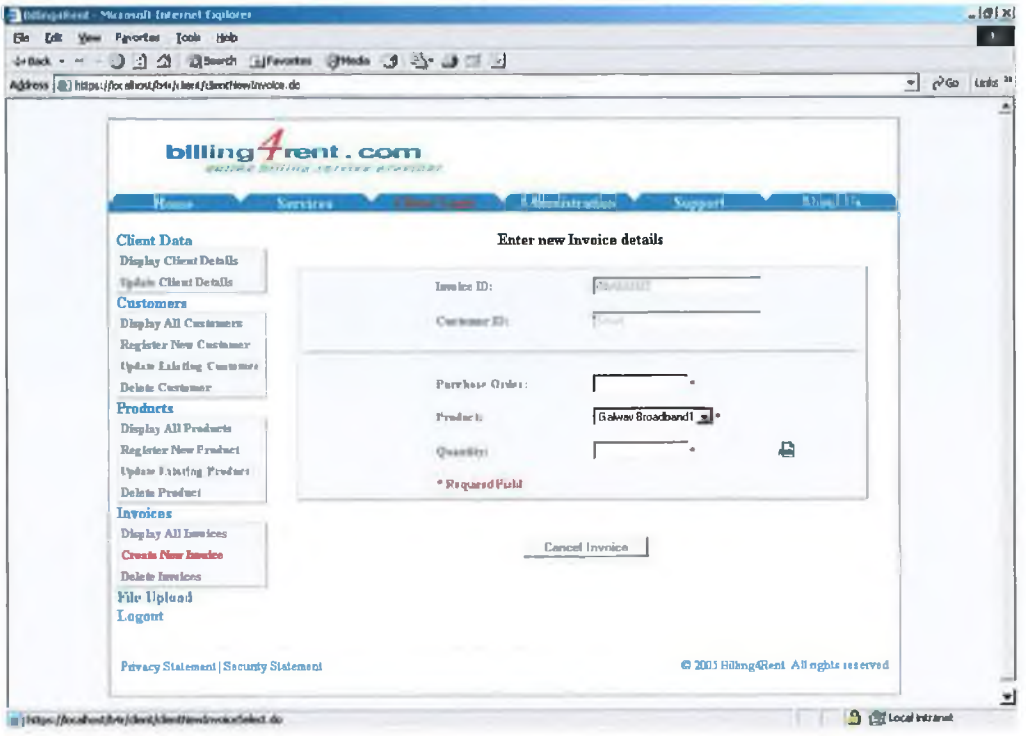
Billing4Rent billing solution 'Delete Product' page



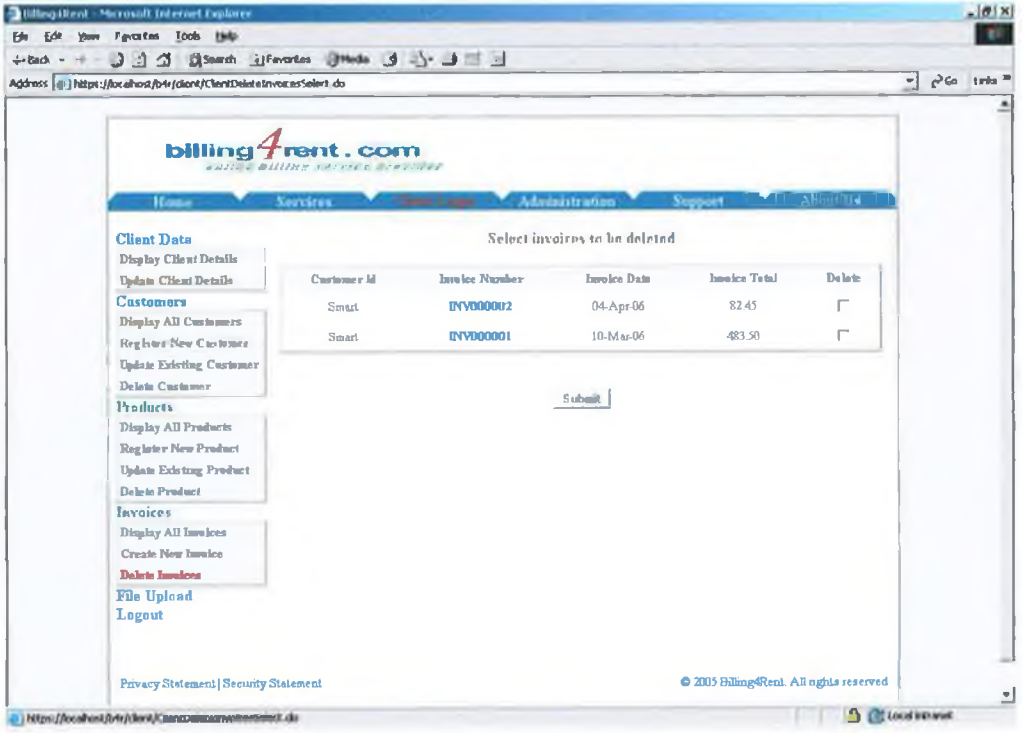
Billing4Rent billing solution 'Display All Invoices' page



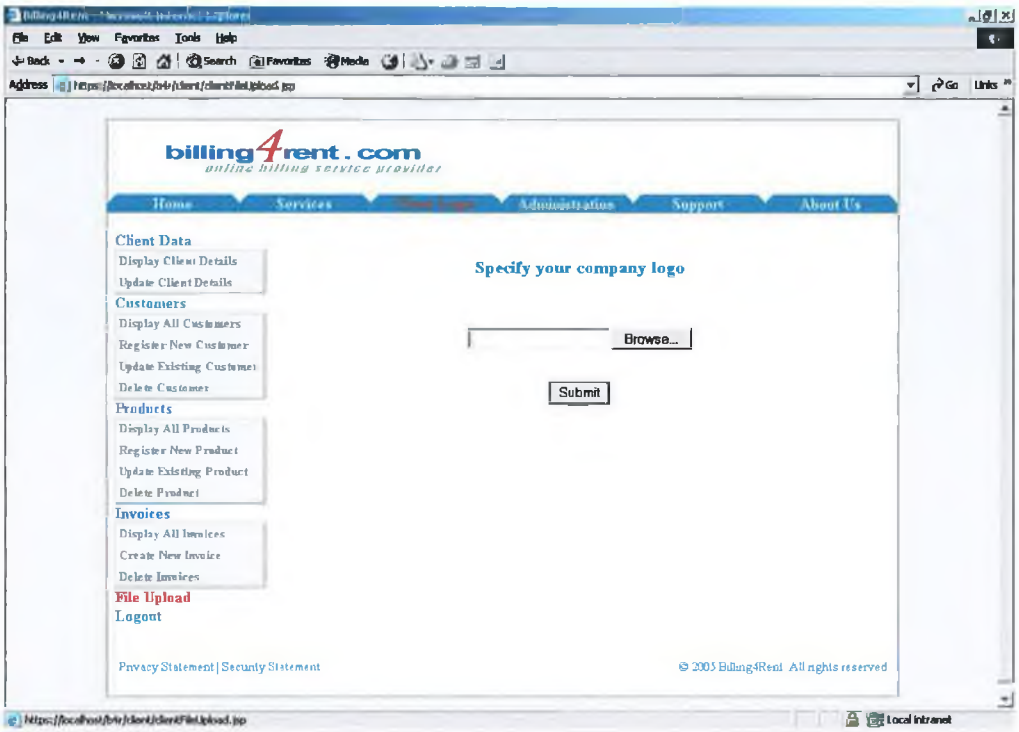
Billing4Rent billing solution 'Display Invoice' page



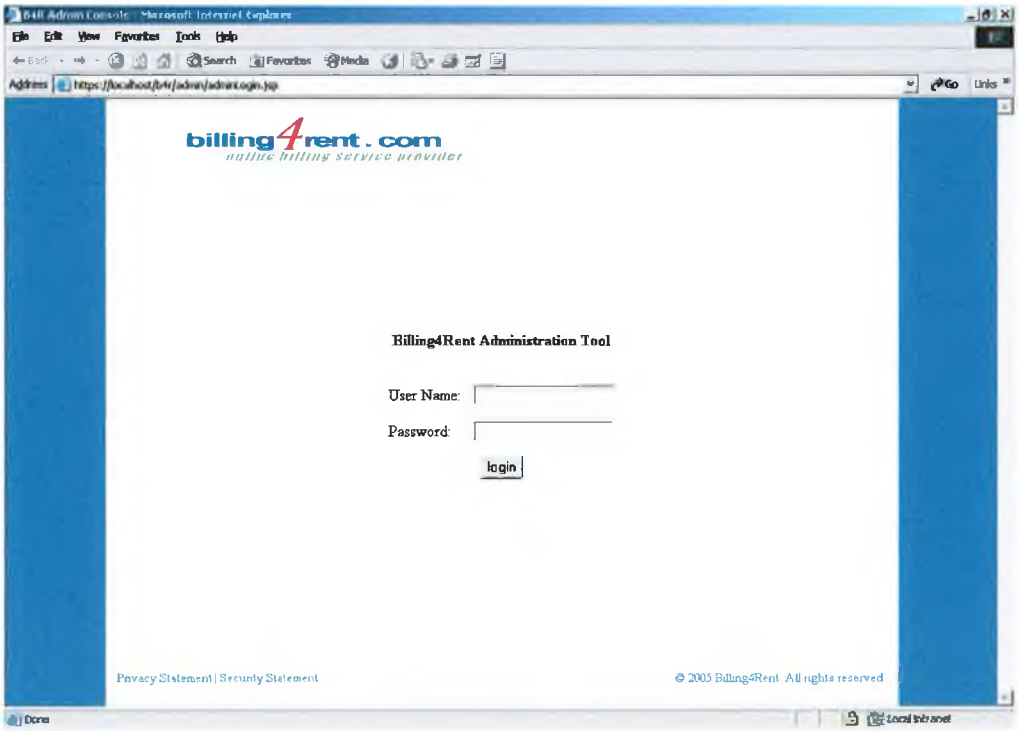
Billing4Rent billing solution 'Create New Invoice' page



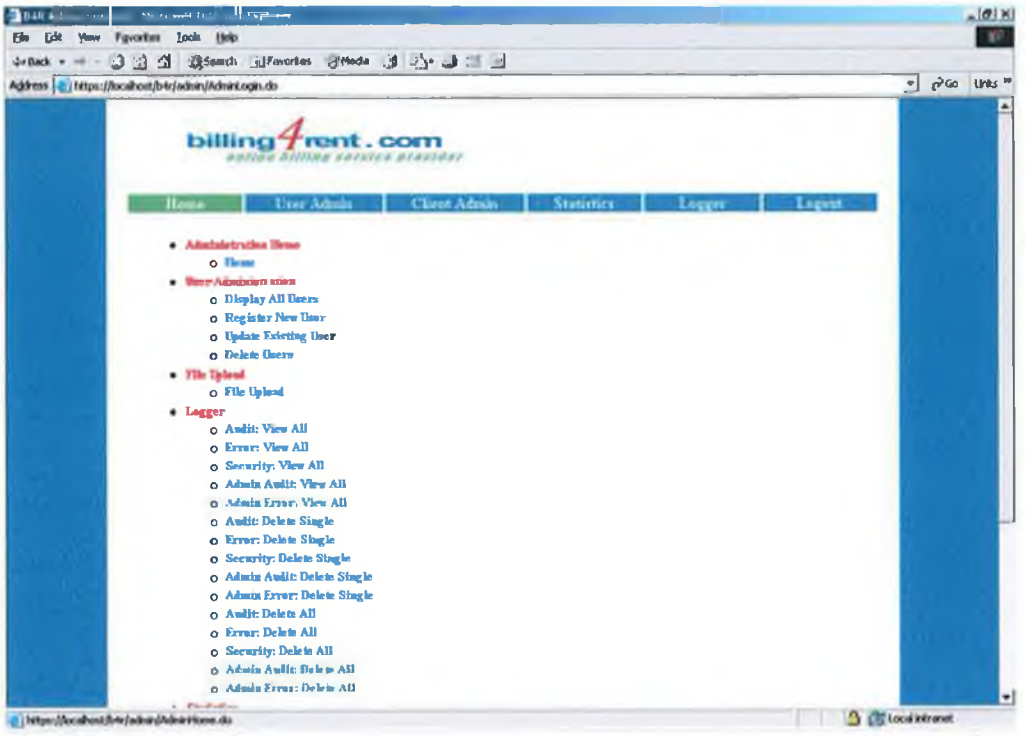
Billing4Rent billing solution 'Delete Invoices' page



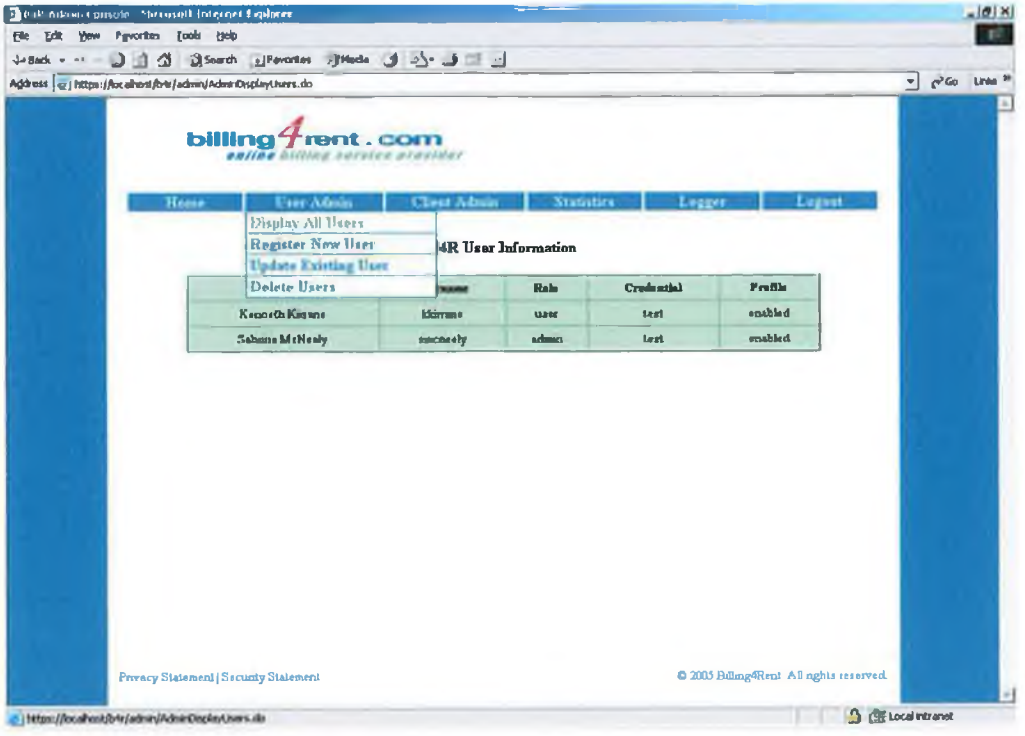
Billing4Rent billing solution 'File Upload' page



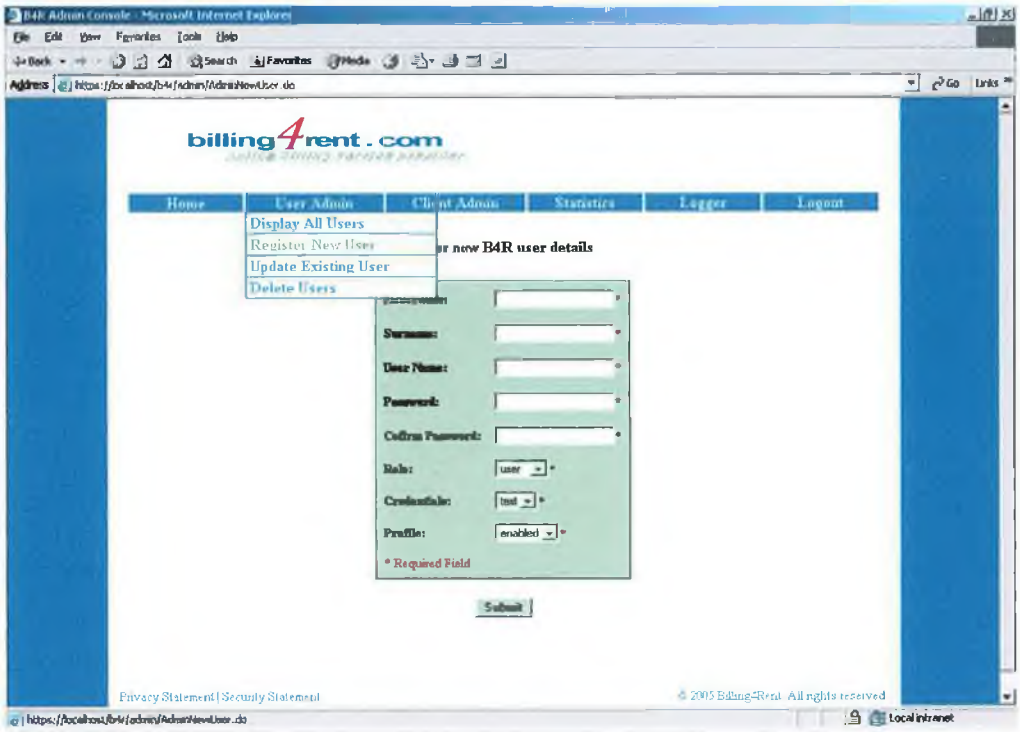
Billing4Rent administration tool 'Login' page



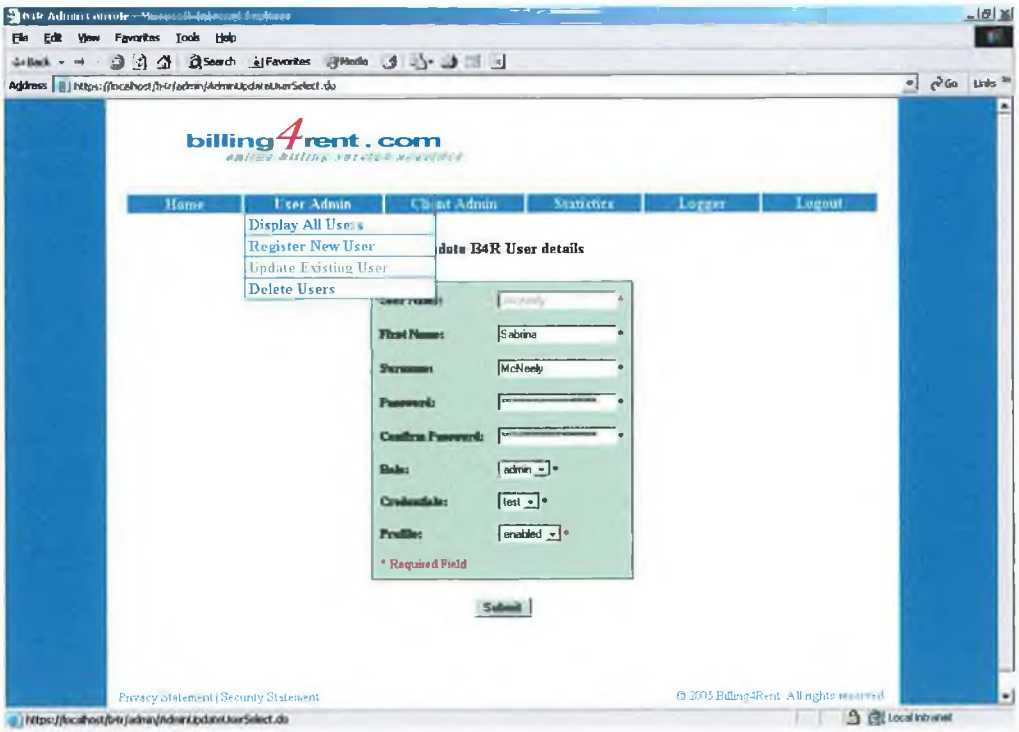
Billing4Rent administration tool 'Home' page



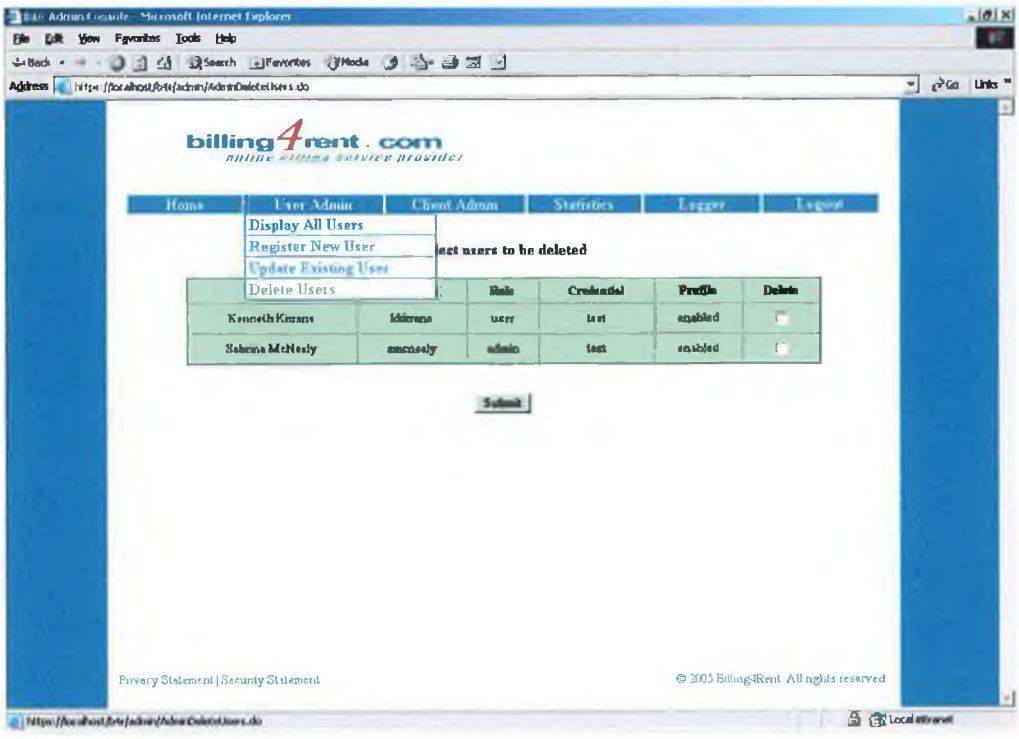
Billing4Rent administration tool 'Display All Users' page



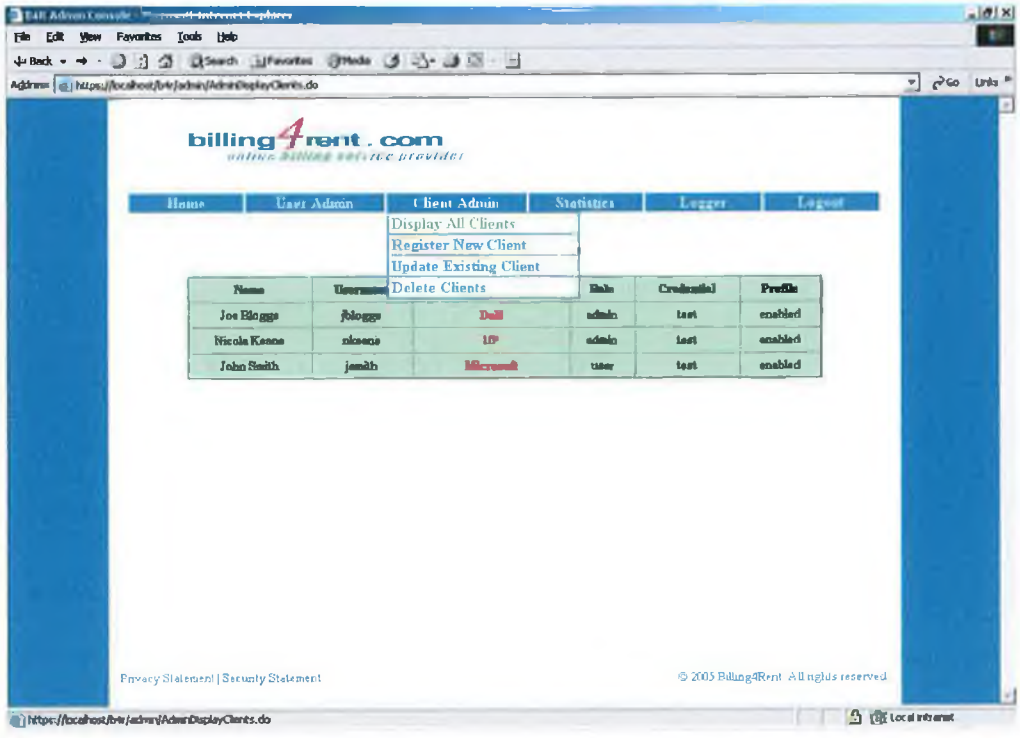
Billing4Rent administration tool 'Register New User' page



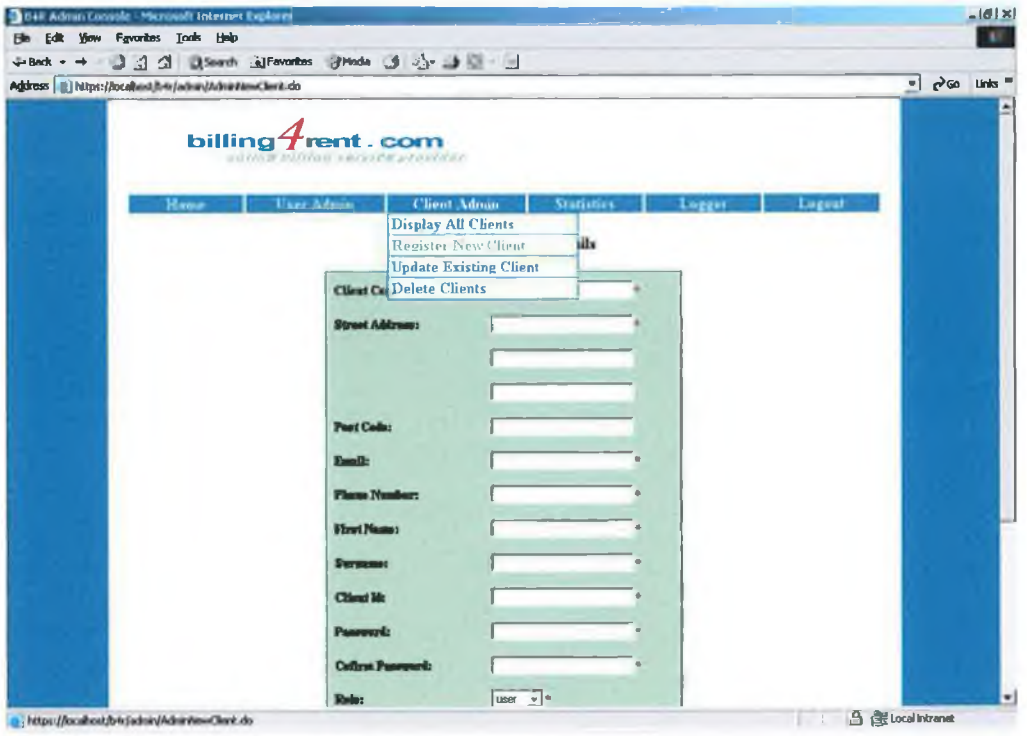
Billing4Rent administration tool 'Update Existing User' page



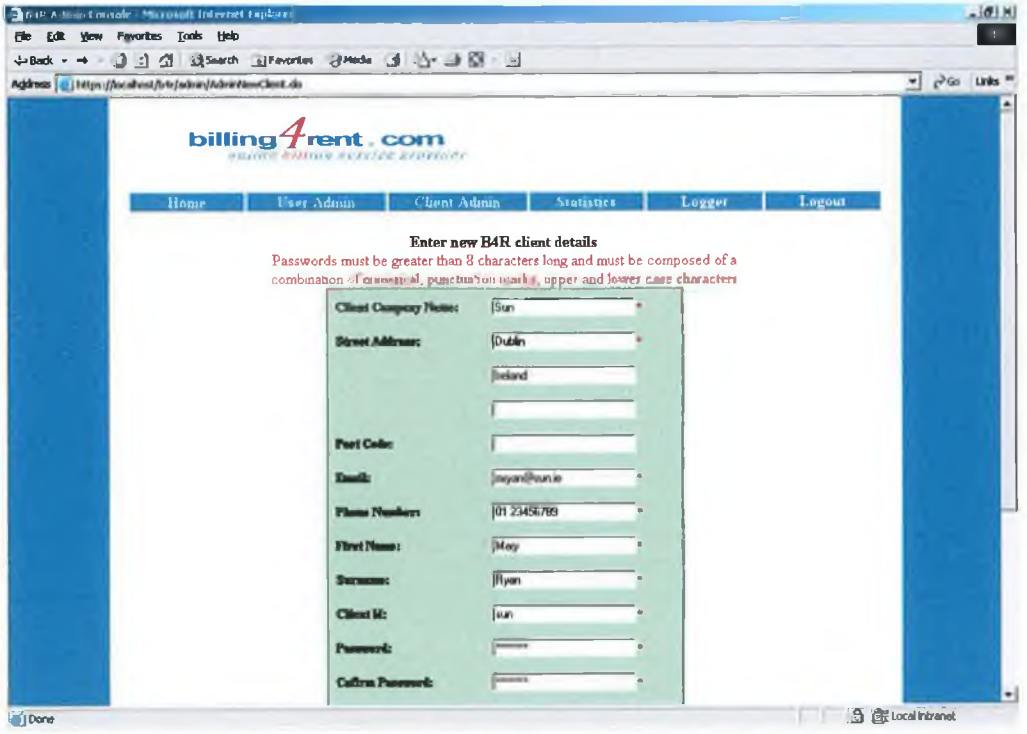
Billing4Rent administration tool 'Delete Users' page



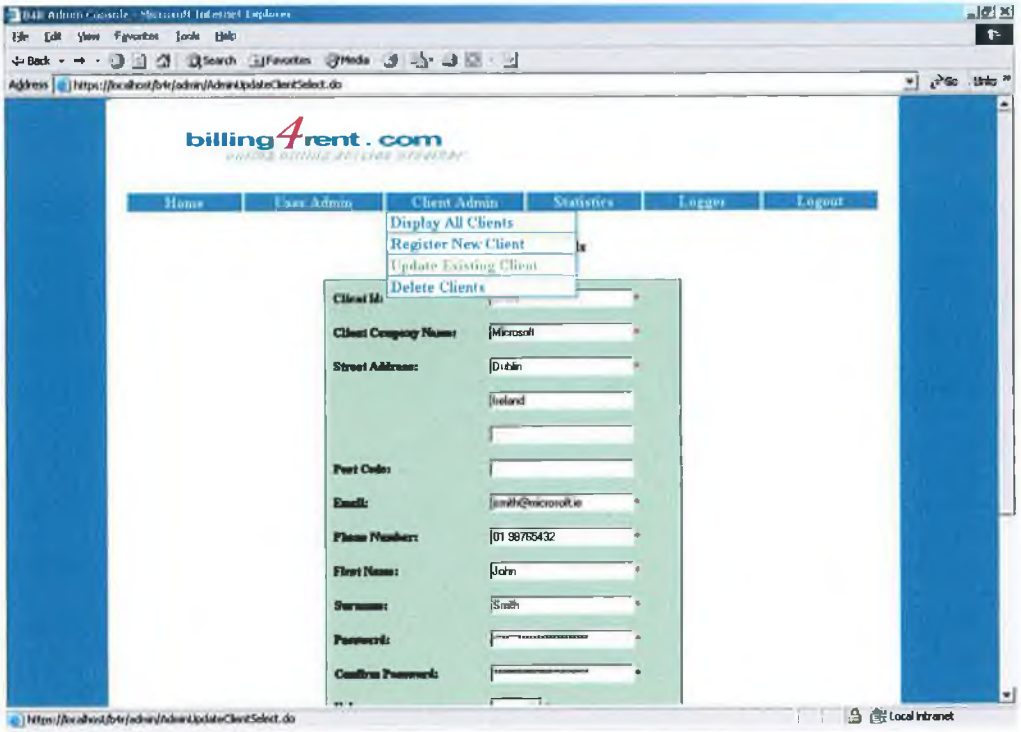
Billing4Rent administration tool 'Display All Clients' page



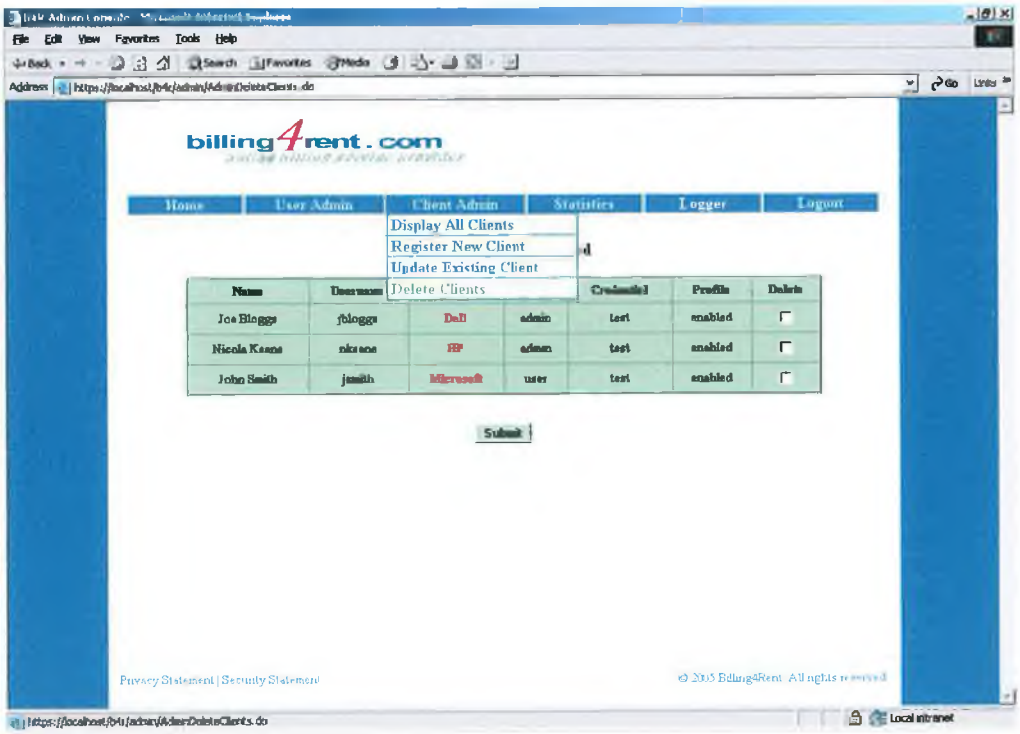
Billing4Rent administration tool 'Register New Client' page



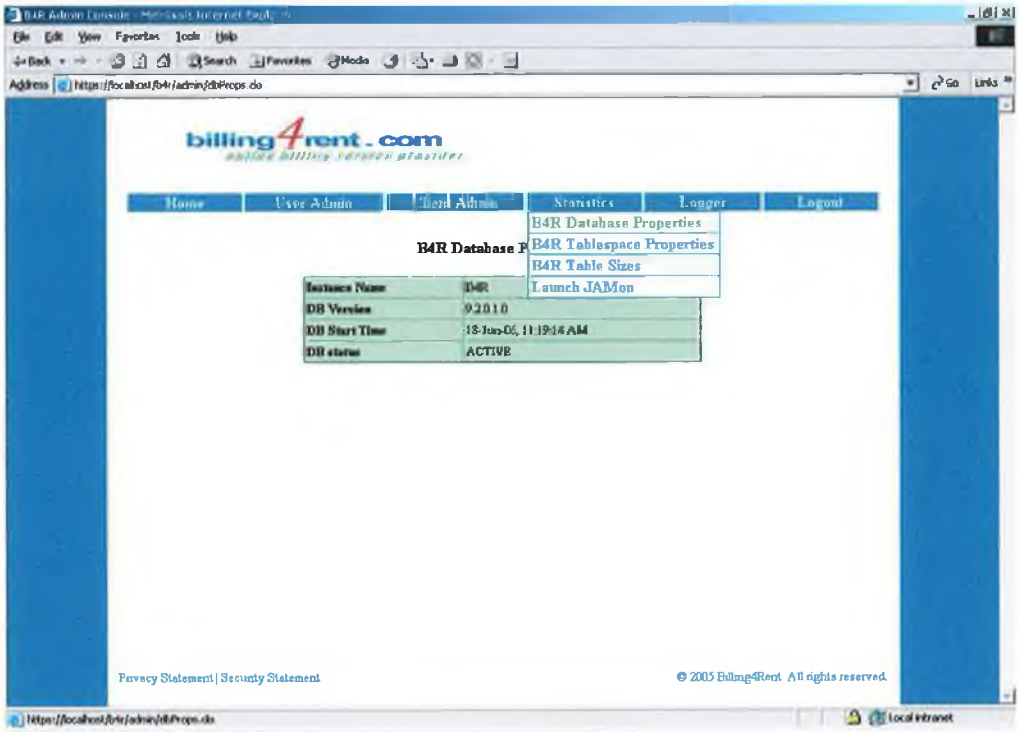
Billing4Rent administration tool 'Invalid Client Password' page



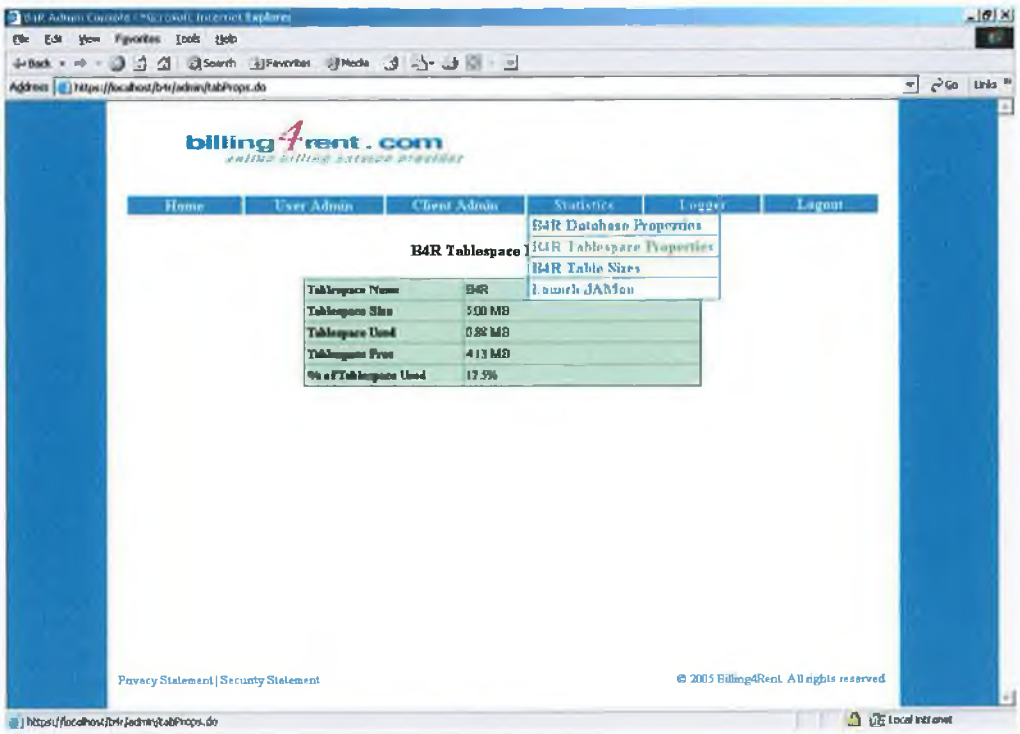
Billing4Rent administration tool 'Update Existing Client' page



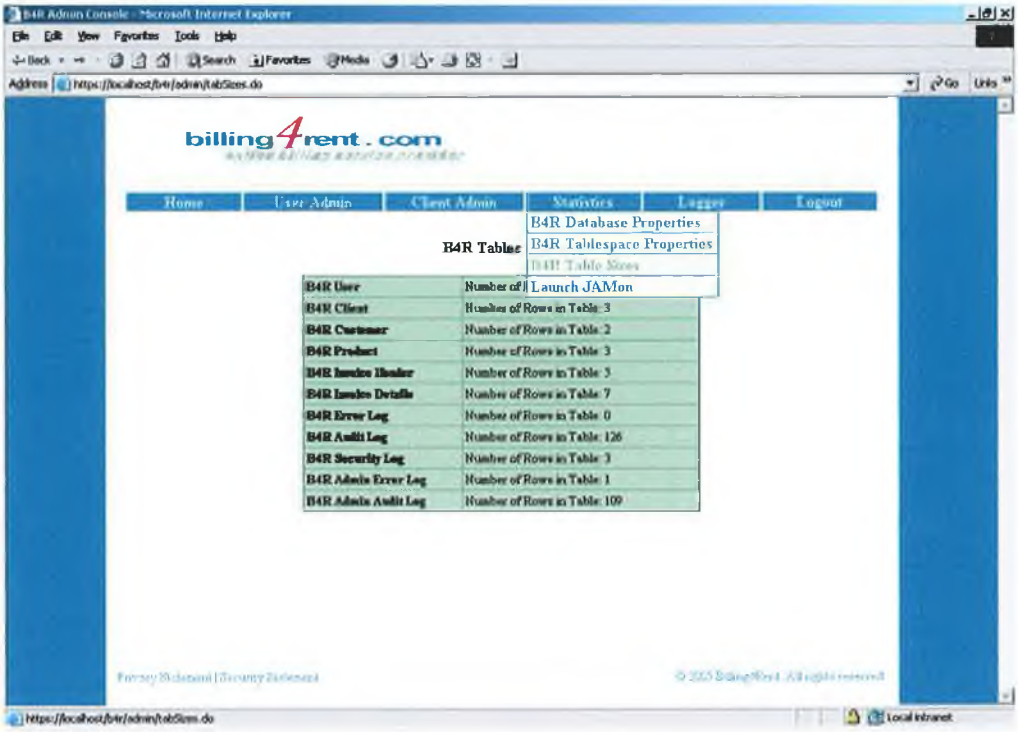
Billing4Rent administration tool 'Delete Clients' page



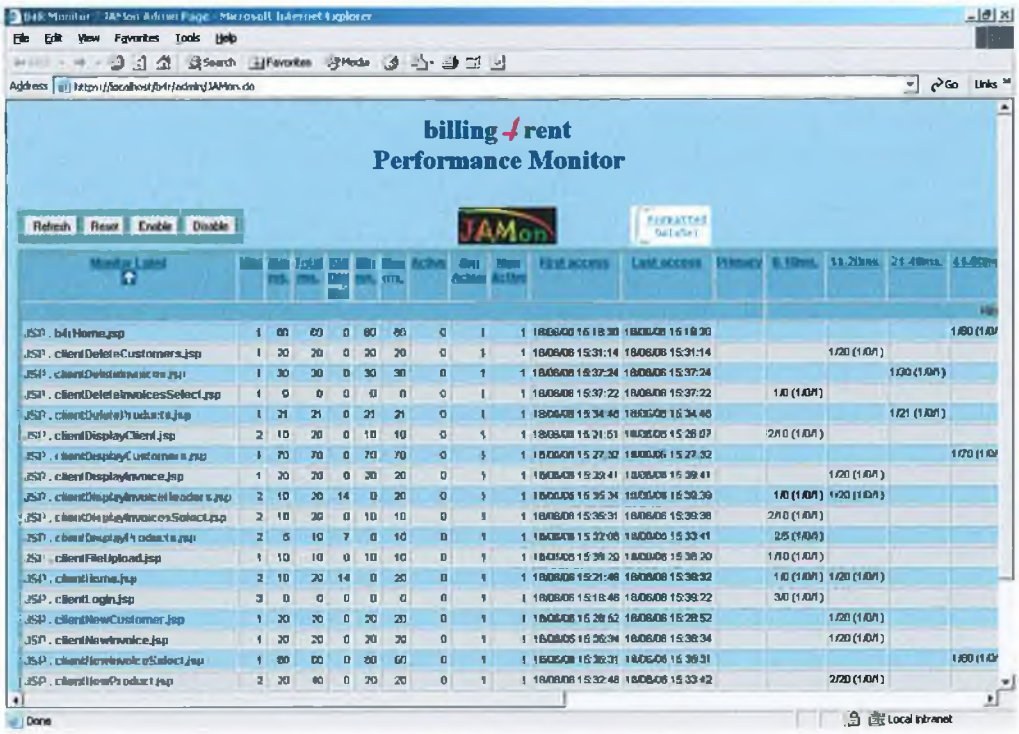
Billing4Rent administration tool 'B4R Database Properties' page



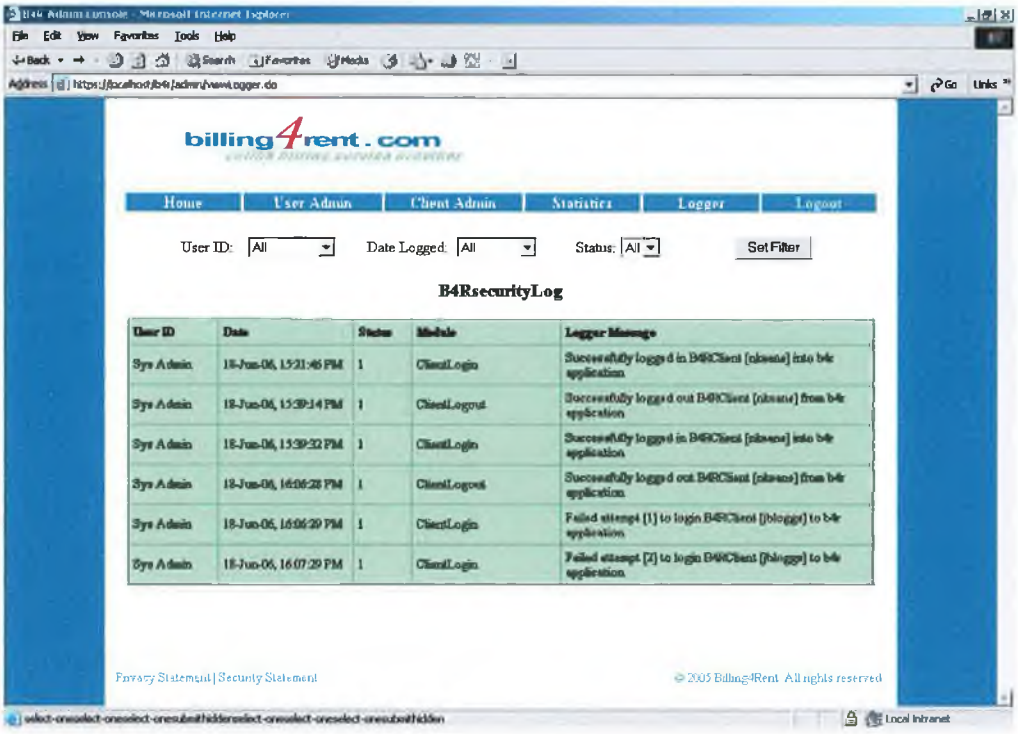
Billing4Rent administration tool 'B4R Tablespace Properties' page



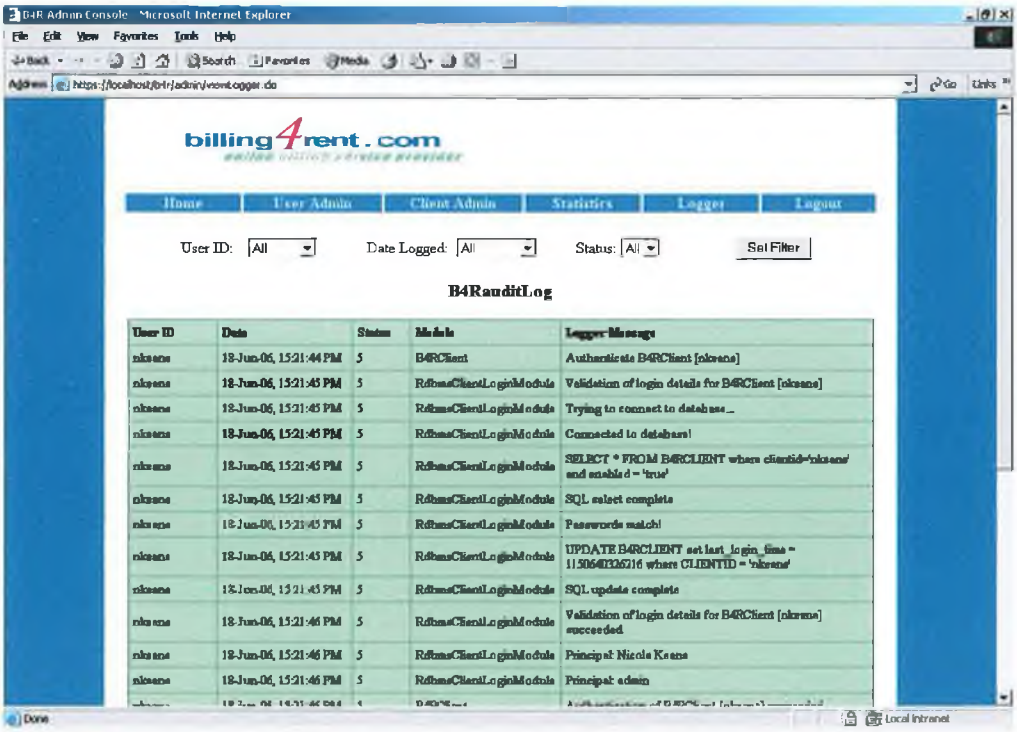
Billing4Rent administration tool 'B4R Table Sizes' page



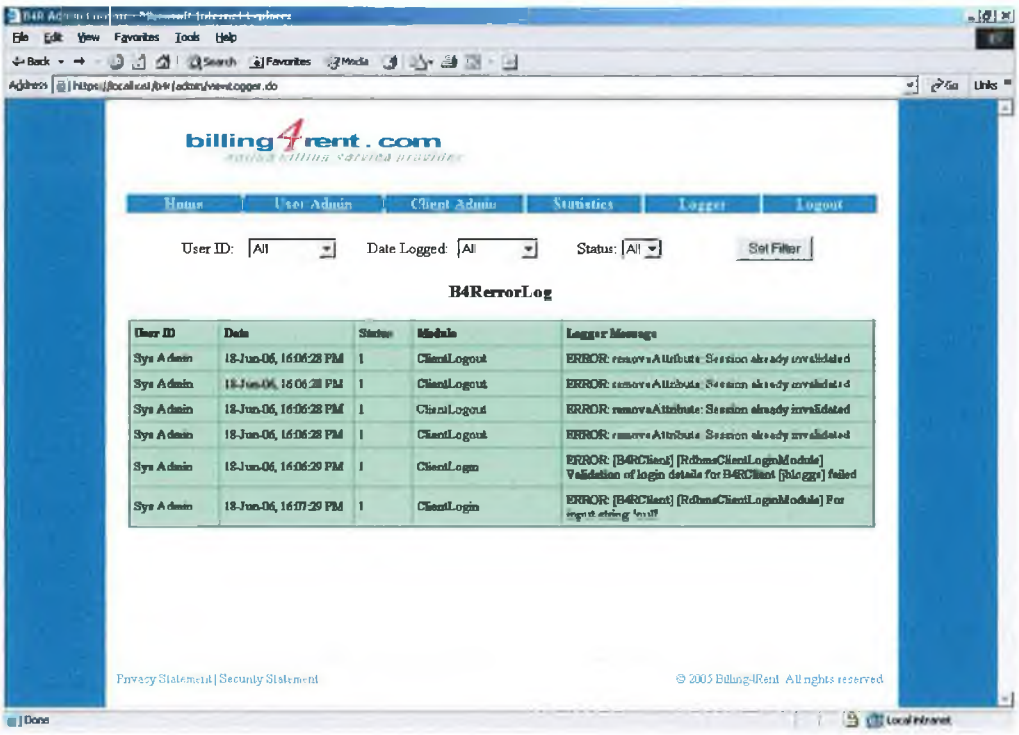
Billing4Rent administration tool 'B4R Performance Monitor' page



Billing4Rent administration tool 'B4R Security Log' page



Billing4Rent administration tool 'B4R Audit Log' page



Billing4Rent administration tool 'B4R Error Log' page

Appendix III

The following is an extract from the Billing4Rent security policy namely the 'Security Policy for the B4R ASP Service' document. The security policy is composed of a combination of guidelines with regard to general security, physical security, host security, network security and data security adapted from the SANS (SysAdmin, Audit, Network, Security) Institute security policy templates. A security policy is an ever-changing document, constantly amended to cater for updates to the organisation's systems and procedures.

2 SECURITY STANDARDS & POLICIES

2.1 Standards

2.1.1 General Security

Billing4Rent reserves the right to periodically audit the Billing4Rent ASP infrastructure to ensure compliance with the Security Policy for the Billing4Rent ASP Service.

Billing4Rent shall provide a complete architecture document that includes a full network diagram of the Billing4Rent ASP environment, illustrating the relationship between the ASP environment and any other relevant networks, with a full data flowchart detailing where client data resides, the applications that manipulate it, and the security thereof.

Billing4Rent should be able to immediately disable all or part of the functionality of the ASP solution should a security issue be identified. A Disaster Recovery Plan should be documented and tested, to minimise the impact of interruption due to unavoidable events and to ensure ease of recovery.

Non-intrusive Billing4Rent ASP network audits (basic portscans, etc.) may be performed randomly, without prior notice. More intrusive network and physical audits may be conducted with 24 hours notice.

2.1.2 Physical Security

The Billing4Rent ASP infrastructure (hosts, network equipment, etc.) should be located in a physically secure facility otherwise known as a data center. Access to the facility should be restricted to authorised personnel, through the use of biometric scanners plus user pins or passwords.

The Billing4Rent ASP infrastructure should be further secured physically, through the use of security cables, padlocks and other such devices.

Billing4Rent shall document in the Billing4Rent ASP *Access Control Policy* stringent access control policies and procedures, outlining who is authorised to enter the data center, in addition to who is authorised to access the Billing4Rent ASP infrastructure.

Closed circuit television should be installed in order to monitor activities throughout the data center. Billing4Rent shall document in the Billing4Rent ASP *Closed Circuit Television Policy* the type of system used and security footage storage methods.

2.1.3 Host Security

Appropriate service packs and patches should be applied to the Billing4Rent ASP operating systems and applications as soon as they are made available. Billing4Rent shall document in the Billing4Rent ASP *Service Packs /Patches Policy* all current patches on hosts, including host operating system patches, web servers, databases, and any other applicable applications.

The corporate standard anti-virus/anti-spyware software should be installed on all Billing4Rent machines. The anti-virus/anti-spyware site should be checked periodically for a list of updates, which should be installed as they become available. Billing4Rent shall document in the Billing4Rent ASP *Anti-virus/Anti-spyware Policy* how the organisation will keep up to date with virus and spyware vulnerabilities and how updates will be applied.

Access to Billing4Rent ASP solution should be restricted to authorised personnel by enforcing strong passwords. Billing4Rent shall document in the Billing4Rent ASP *Password Policy* guidelines for the generation of passwords for all Billing4Rent ASP

infrastructure, including minimum password length and how often passwords are changed.

Billing4Rent shall document in the Billing4Rent ASP *Account Maintenance Policy* the account generation, maintenance and termination process, for both maintenance as well as user accounts.

Client data stored locally should be encrypted to ensure security and integrity of the data, as outlined in the Billing4Rent ASP *Cryptography Policy*.

Billing4Rent shall document the organisations processes for monitoring the integrity and availability of Billing4Rent ASP hosts in the Billing4Rent *Audit Policy*.

2.1.4 Network Security

The Billing4Rent ASP solution should be tested rigorously to ensure that neither the application nor the configuration of the ASP environment pose a security risk.

The Billing4Rent ASP solution firewall should be configured to filter undesired traffic between the Internet and Billing4Rent ASP infrastructure. All unnecessary services running on the ASP infrastructure should be disabled, by shutting down unused ports.

Remote access to Billing4Rent ASP solution hardware should be limited to an absolute minimum. Billing4Rent shall document in the Billing4Rent ASP *Remote* and *Dial-in Access Policies*, standards for connecting to Billing4Rent's network from remote hosts.

Data sent over the Internet should be encrypted to ensure security and integrity of data being transmitted as outlined in the Billing4Rent ASP *Cryptography Policy*.

2.1.5 Data Security

Billing4Rent shall document the organisations processes for protecting sensitive data in the Billing4Rent *Information Sensitivity Policy*.

Redundant Billing4Rent ASP hardware should be disposed of in an appropriate manner. Software and data should be uninstalled and erased to guarantee that sensitive data is not accessible to unauthorised individuals.

Billing4Rent ASP solution software and the configuration of the Billing4Rent ASP environment should be tested rigorously to ensure the security and the integrity of client data. ASP solution security threats should be identified and the appropriate action should be taken to combat unauthorised access to sensitive data.

2.2 Security Policies

2.2.1 Cryptography Policy

Proven, standard algorithms such as Advanced Encryption Standard (AES) and Ron Rivest, Adi Shamir and Len Adleman Encryption (RSA) should be used as the basis for encryption technologies. AES consists of 128-bit blocks with a 128-bit, 192-bit or alternatively a 256-bit cipher key length. The number of rounds/iterations of the encryption/decryption algorithm is dependent on the cipher key length. RSA encryption security is based on the difficulty in factoring large numbers, with a key that varies depending on the implementation used. RSA can also be used as a digital signature to authenticate the originator and ensure the integrity of the data. For example, Secure Socket Layer (SSL) uses RSA encryption. Encryption algorithms should be reviewed regularly and updated based on technological advances. The use of proprietary encryption algorithms is prohibited for any purpose.

Connections to the Billing4Rent ASP utilising the Internet should be protected using any of the following cryptographic technologies: IPSec, SSL, SSH/SCP, PGP.

2.2.2 Anti-virus/Anti-spyware Policy

Computer systems should be protected against infected by viruses (malicious applications loaded onto a computer system without ones knowledge), worms (malicious applications that can replicate themselves across a network) and spyware (software that covertly gathers user information through the user's Internet connection without his or her knowledge). The following procedure should be adhered to in order to provide optimal protection against viruses, worms and spyware:

- The corporate standard anti-virus/anti-spyware software should be installed on all Billing4Rent machines. The anti-virus/anti-spyware site should be checked periodically for a list of updates, which should be installed as they become available.

- Under no circumstances should files or macros attached to an email from an unknown, suspicious or untrustworthy source be opened. Such files should be deleted immediately, and emptied from the deleted items/trash folder.

- Under no circumstances should spam, chain, and other junk email be forwarded either internally or externally. Such files should be deleted immediately, and emptied from the deleted items/trash folder.

- Under no circumstances should files be downloaded from unknown or suspicious sources.

- Direct disk sharing with read/write access should be avoided unless there is an absolute business requirement to do so.

- External media such as floppy disk's, CDs and flash drives from unknown sources should always be scanned before usage.

- Critical data and system configurations should be backed up on a regular basis and stored in a safe place, in accordance with the Billing4Rent *Disaster Recovery Policy*.

- If lab-testing conflicts with anti-virus/anti-spyware software, run the anti-virus/anti-spyware utility to ensure a clean machine, disable the software, then run the lab test. After the lab test, enable the anti-virus/anti-spyware software. When the anti-virus/anti-spyware software is disabled, do not run any applications that could transfer a virus, e.g., email or file sharing.

2.2.3 Password Policy

Passwords are an important aspect of computer security as they are the front line of protection for user accounts. A poorly chosen password may result in the compromise of Billing4Rent's entire corporate network. As such, all Billing4Rent employees (including contractors and vendors with access to Billing4Rent systems) are responsible for taking the appropriate steps, as outlined below, to select and secure their passwords:

- All system-level passwords (e.g., root, enable, NT admin, application administration accounts, etc.) should be changed on at least a quarterly basis. Password changes will be enforced with an automatic expiration and prevention of repeated or reused passwords.

- All user-level passwords (e.g., email, web, desktop computer, etc.) should be changed at least every six months. Password changes will be enforced with an automatic expiration and prevention of repeated or reused passwords.

- User accounts that have system-level privileges granted through group memberships should have a unique password from all other accounts held by that user.

- All user-level and system-level passwords should conform to the following guidelines:
  1) Passwords should contain both upper and lower case characters (e.g., a-z, A-Z).
  2) Passwords should include digits and punctuation characters as well as letters e.g., 0-9, !@#$%^&*()_+|~-=\`{}[]:";'<>?,./).
  3) Passwords should be at least eight alphanumeric characters long.
  4) Passwords should not be word in any language, slang, dialect, jargon, etc.
  5) Passwords should not be based on personal information, names of family, etc.

- Billing4Rent accounts should be different from all non-Billing4Rent passwords (e.g., personal ISP account, option trading, benefits, etc.). Where possible, different passwords should be used for various Billing4Rent access needs. For example, select one password for the engineering systems and a separate password for IT systems.

- Billing4Rent passwords should not be shared with anyone, including administrative assistants or secretaries. All passwords are to be treated as sensitive, confidential Billing4Rent information. If someone demands a password, refer them to this document or have them speak to a member of the Billing4Rent management team:
  1) Don't reveal a password over the phone to ANYONE.
  2) Don't reveal a password in an email message.
  3) Don't reveal a password to the boss.
  4) Don't talk about a password in front of others.

5) Don't hint at the format of a password (e.g., "my family name").
6) Don't reveal a password on questionnaires or security forms.
7) Don't share a password with family members.
8) Don't reveal a password to co-workers while on vacation.

- The "Remember Password" feature of applications (e.g. Eudora, Outlook, Netscape Messenger) should not be used.

- Passwords should not be written down and stored anywhere in your office or stored electronically on any computer system (including Palm Pilots or similar devices) without encryption.

- Passwords should not be inserted into email messages or other forms of electronic communication.

- If it is suspected an account or password has been compromised, the incident should be reported immediately to a member of the Billing4Rent management team and all passwords should be changed.

- Application developers should ensure their programs contain the following security precautions:
  1) Support authentication of individual users, not groups.
  2) Passwords should not be stored in clear text or in any easily reversible form.
  3) Passwords should provide for some sort of role management, such that one user can take over the functions of another without having to know the other's password.

- Access to the Billing4Rent networks via remote access is to be controlled using either a one-time password authentication or a public/private key system with a strong passphrase.

Passphrases are generally used for public/private key authentication. A public/private key system defines a mathematical relationship between the public key that is known by all, and the private key, that is known only to the user. Without the passphrase to "unlock" the private key, the user cannot gain access.

Passphrases are not the same as passwords. A passphrase is a longer version of a password and is, therefore, more secure. A passphrase is typically composed of multiple words. Because of this, a passphrase is more secure against "dictionary attacks."

A good passphrase is relatively long and contains a combination of upper and lowercase letters and numeric and punctuation characters. An example of a good passphrase:

"The*?#>*@TrafficOnThe101Was*&#!#ThisMorning"

All of the rules above that apply to passwords apply to passphrases.

2.2.4 Dial-In Access Policy

The purpose of this policy is to protect Billing4Rent's electronic information from being inadvertently compromised by authorised personnel using a dial-in connection.

- Billing4Rent's employees and authorised third parties (customers, vendors, etc.) can use dial-in connections to gain access to the corporate network. Dial-in access should be strictly controlled, using one-time password authentication. Dial–in passwords should be requesting using the Billing4Rent Account Request Process.

- It is the responsibility of individuals with dial-in access privileges to ensure unauthorised individuals do not gain access to company information system

resources. An employee and/or authorised third party who is granted dial-in access privileges should remain constantly aware that dial-in connections between their location and Billing4Rent are literal extensions of Billing4Rent's corporate network, and that they provide a potential path to the company's most sensitive information. The employee and/or authorised third party individual should take every reasonable measure to protect Billing4Rent 's assets.

- Analog and non-GSM digital cellular phones cannot be used to connect to Billing4Rent's corporate network, as their signals can be readily scanned and/or hijacked by unauthorised individuals. Only GSM standard digital cellular phones are considered secure enough for connection to Billing4Rent's network. For additional information on wireless access to the Billing4Rent's network, consult the *Wireless Communications Policy*.

- Dial-in accounts are considered 'as needed' accounts. Account activity should be monitored, and if a dial-in account is not used for a period of six months the account should expire and no longer function. If dial-in access is subsequently required, the individual should request a new account.

2.2.5 Remote Access Policy

The purpose of this policy is to define standards for connecting to Billing4Rent's network from any host. These standards are designed to minimise the potential exposure to Billing4Rent from damages, which may result from unauthorised use of Billing4Rent resources. Damages include the loss of sensitive or company confidential data, intellectual property, damage to public image, damage to critical Billing4Rent internal systems, etc.

It is the responsibility of Billing4Rent employees, contractors, vendors and agents with remote access privileges to Billing4Rent's corporate network to ensure that their remote

access connection is given the same consideration as the user's on-site connection to Billing4Rent.

General access to the Internet for recreational use by immediate household members through the Billing4Rent network on personal computers is prohibited.

The following policies provide details in relation to the protecting of information when accessing the corporate network via remote access methods, and acceptable use of Billing4Rent's network:

1) Cryptography Policy
2) Virtual Private Network (VPN) Policy
3) Wireless Communications Policy

General Remote Access Guidelines:

- Secure remote access should be strictly controlled. Control should be enforced via one-time password authentication or public/private keys with strong pass-phrases. For information on creating a strong pass-phrase see the Billing4Rent Password Policy.

- At no time should any Billing4Rent employee provide their login or email password to anyone, not even family members.

- Billing4Rent employees and contractors with remote access privileges should ensure that their Billing4Rent owned or personal computer or workstation, which is remotely connected to Billing4Rent 's corporate network, is not connected to any other network at the same time, with the exception of personal networks that are under the complete control of the user.

- Billing4Rent employees and contractors with remote access privileges to Billing4Rent 's corporate network should not use non-Billing4Rent email accounts (i.e., Hotmail, Yahoo, AOL), or other external resources to conduct Billing4Rent

business, thereby ensuring that official business is never confused with personal business.

- Routers for dedicated ISDN lines configured for access to the Billing4Rent network should meet minimum authentication requirements.

- Reconfiguration of a home user's equipment for the purpose of split-tunneling or dual homing is not permitted at any time.

- Frame Relay should meet minimum authentication requirements of DLCI standards.

- A member of the Billing4Rent management team should approve non-standard hardware configurations and security configurations for access to hardware.

- All hosts that are connected to Billing4Rent internal networks via remote access technologies, should use the most up-to-date anti-virus/anti-spyware software, this includes personal computers.

- Personal equipment that is used to connect to Billing4Rent's networks should meet the requirements of Billing4Rent owned equipment for remote access.

2.2.6 Wireless Communication Policy

This policy prohibits access to Billing4Rent networks via unsecured wireless communication mechanisms. Only wireless systems that meet the criteria of this policy are approved for connectivity to Billing4Rent 's networks.

- Personal equipment that is used to connect to Billing4Rent's networks should meet the requirements of Billing4Rent owned equipment for remote access.

- All wireless access points/base stations connected to the corporate network should be registered and approved the Billing4Rent management team. These access points/base stations are subject to periodic penetration tests and audits. All wireless Network Interface Cards (i.e., PC cards) used in corporate laptop or desktop computers should be registered with he Billing4Rent management team.

- All wireless LAN access should use corporate-approved vendor products and security configurations.

- All computers with wireless LAN devices should utilise a corporate-approved Virtual Private Network (VPN) configured to drop all unauthenticated and unencrypted traffic. To comply with this policy, wireless implementations should maintain point to point hardware encryption of at least 56 bits. All implementations should support a hardware address that can be registered and tracked, i.e., a MAC address.

- The Service Set Identifier (SSID) shall be configured so that it does not contain any identifying information about the organisation, such as the company name, division title, employee name, or product identifier.

2.2.7 Information Sensitivity Policy

The *Information Sensitivity Policy* is intended to help employees determine what information can be disclosed to non-employees, as well as the relative sensitivity of information that should not be disclosed outside of Billing4Rent without proper authorisation.

The information covered in these guidelines includes, but is not limited to, information that is either stored or shared via any means. This includes: electronic information, information on paper, and information shared orally or visually (such as telephone and video conferencing).

All employees should familiarise themselves with the information labeling and handling guidelines that follow this introduction. It should be noted that the sensitivity level definitions were created as guidelines and to emphasise common sense steps that you can take to protect Billing4Rent Confidential information (e.g., Billing4Rent Confidential information should not be left unattended in conference rooms).

Questions about the proper classification of a specific piece of information or with regard to these guidelines should be addressed to a member of the Billing4Rent management team.

The sensitivity guidelines below provide details on how to protect information at varying sensitivity levels. Use these guidelines as a reference only, as Billing4Rent Confidential information in each column may necessitate more or less stringent measures of protection depending upon the circumstances and the nature of the Billing4Rent Confidential information in question.

Minimal Sensitivity

General corporate information; some personnel and technical information.

Marking guidelines for information in hardcopy or electronic form:
Marking is at the discretion of the owner or custodian of the information. If marking is desired, the words "Billing4Rent Confidential" may be written or designated in a conspicuous place on or in the information in question. Other labels that may be used include "Billing4Rent Proprietary" or similar labels at the discretion of your individual business unit or department. Even if no marking is present, Billing4Rent information is presumed to be "Billing4Rent Confidential" unless expressly determined to be Billing4Rent Public information by a Billing4Rent employee with authority to do so. Any of these markings may be used with the additional annotation of "3rd Party Confidential".

Access: Billing4Rent employees, contractors, people with a business need to know.

Distribution within Billing4Rent: Standard interoffice mail, approved electronic mail and electronic file transmission methods.

Distribution outside of Billing4Rent internal mail: An Phost mail and other public or private carriers, approved electronic mail and electronic file transmission methods.

Electronic distribution: No restrictions except that it be sent to only approved recipients.

Storage: Keep from view of unauthorised people; erase whiteboards, do not leave in view on tabletop. Machines should be administered with security in mind. Protect from loss; electronic information should have individual access controls where possible and appropriate.

Disposal/Destruction: Deposit outdated paper information in specially marked disposal bins on Billing4Rent premises; electronic data should be expunged/cleared. Reliably erase or physically destroy media.

Medium Sensitivity

Business, financial, technical, and most personnel information.

Marking guidelines for information in hardcopy or electronic form:
As the sensitivity level of the information increases, you may, in addition or instead of marking the information "Billing4Rent Confidential" or "Billing4Rent Proprietary", wish to label the information "Billing4Rent Internal Use Only" or other similar labels at the discretion of your individual business unit or department to denote a more sensitive level

of information. However, marking is discretionary at all times. Any of these markings may be used with the additional annotation of "3rd Party Confidential".

Access: Billing4Rent employees and non-employees with signed non-disclosure agreements who have a business need to know.

Distribution within Billing4Rent: Standard interoffice mail, approved electronic mail and electronic file transmission methods.

Distribution outside of Billing4Rent internal mail: Sent via An Phost. mail or approved private carriers.

Electronic distribution: No restrictions to approved recipients within Billing4Rent, but should be encrypted or sent via a private link to approved recipients outside of Billing4Rent premises.

Storage: Individual access controls are highly recommended for electronic information.

Disposal/Destruction: In specially marked disposal bins on Billing4Rent premises; electronic data should be expunged/cleared. Reliably erase or physically destroy media.

Maximum Sensitivity

Trade secrets & marketing, operational, personnel, financial, source code, & technical information integral to the success of our company.

Marking guidelines for information in hardcopy or electronic form:
To indicate that Billing4Rent Confidential information is very sensitive, you may should label the information "Billing4Rent Internal: Registered and Restricted", "Billing4Rent Eyes Only", "Billing4Rent Confidential" or similar labels at the discretion of your

individual business unit or department. Once again, this type of Billing4Rent Confidential information need not be marked, but users should be aware that this information is very sensitive and be protected as such. Any of these markings may be used with the additional annotation of "3rd Party Confidential".

Access: Only those individuals (Billing4Rent employees and non-employees) designated with approved access and signed non-disclosure agreements.

Distribution within Billing4Rent: Delivered direct - signature required, envelopes stamped confidential, or approved electronic file transmission methods.

Distribution outside of Billing4Rent internal mail: Delivered direct; signature required; approved private carriers.

Electronic distribution: No restrictions to approved recipients within Billing4Rent, but it is highly recommended that all information be strongly encrypted.

Storage: Individual access controls are very highly recommended for electronic information. Physical security is generally used, and information should be stored in a physically secured computer.

Disposal/Destruction: Strongly Encouraged: In specially marked disposal bins on Billing4Rent premises; electronic data should be expunged/cleared. Reliably erase or physically destroy media.

2.2.8 Virtual Private Network (VPN) Policy

The purpose of this policy is to provide guidelines for Remote Access IPSec or L2TP Virtual Private Network (VPN) connections to the Billing4Rent corporate network. Approved Billing4Rent employees and authorised third parties (customers, vendors, etc.)

may utilise the benefits of VPNs, which are a "user managed" service. This means that the user is responsible for selecting an Internet Service Provider (ISP), coordinating installation, installing any required software, and paying associated fees. Further details may be found in the *Remote Access Policy*.

- It is the responsibility of employees with VPN privileges to ensure that unauthorised users are not allowed access to Billing4Rent internal networks.

- VPN use is to be controlled using either a one-time password authentication such as a token device or a public/private key system with a strong passphrase.

- When actively connected to the corporate network, VPNs will force all traffic to and from the PC over the VPN tunnel: all other traffic will be dropped.

- Dual (split) tunneling is NOT permitted; only one network connection is allowed.

- VPN gateways will be set up and managed by Billing4Rent network operational groups.

- All computers connected to Billing4Rent internal networks via VPN or any other technology should use the most up-to-date anti-virus/anti-spyware software that is the corporate standard this includes personal computers.

- VPN users will be automatically disconnected from Billing4Rent's network after thirty minutes of inactivity. The user should then logon again to reconnect to the network. Pings or other artificial network processes are not to be used to keep the connection open.

- The VPN concentrator is limited to an absolute connection time of 24 hours.

- Users of computers that are not Billing4Rent owned equipment should configure the equipment to comply with Billing4Rent 's VPN and Network policies.

- Only VPN clients approved by the Billing4Rent management team may be used.

- By using VPN technology with personal equipment, users should understand that their machines are a de facto extension of Billing4Rent's network, and as such are subject to the same rules and regulations that apply to Billing4Rent owned equipment.

2.2.9 Account Maintenance Policy

Billing4Rent login accounts are an important aspect of computer security, as along with a corresponding password they provide access to data stored locally and on other machines connected to the Billing4Rent corporate network. As such, all Billing4Rent employees (including contractors and vendors with access to Billing4Rent systems) are responsible for taking the appropriate steps to safeguard account login information.

- The Billing4Rent network administration is responsible for the generation, maintenance and termination of all Billing4Rent accounts i.e. user, system, email etc. and associated privileges.

- It is the responsibility of the Billing4Rent department manager to notify the Billing4Rent network administration of employee (including contractors and vendors with access to Billing4Rent systems) commencement or termination.

- On commencement of employment *an account maintenance* form should be completed by the Billing4Rent department manager and returned via email to the network administrator. The Billing4Rent network administrator will confirm account creation via return email.

- All users are required to sign a declaration that they agree to comply with the Billing4Rent *Acceptable Use Policy*, prior to using their computer account.

- Where possible, systems will be configured in order to force users to change their first log-on password.

- Any user, who has being assigned an account to access a network or resource, should keep the account details and corresponding password confidential.

- On termination of employment *an account maintenance* form should be completed by the Billing4Rent department manager and returned via email to the network administrator. The Billing4Rent network administrator will confirm account deletion via return email.

- The allocation of system privileges is to be strictly controlled. Users are to be given specific account profiles and privileges in accordance with their particular function or role. Where additional privileges are required *an account maintenance* form should be completed by the Billing4Rent department manager and returned via email to the network administrator. The Billing4Rent network administrator will review and respond to the request via return email.

- Accounts will be frozen after three failed logon attempts. All erroneous password entries will be recorded in an audit log for later inspection and action, as necessary.

- All PCs, laptops and workstations should be secured by locking or logging-off (control-alt-delete for Win2K users) when the host will be unattended.

- The Billing4Rent network administrator will delete accounts unused for a period of six months from the system.

- The Billing4Rent network administrator may revoke accounts at any time if computing privileges are abused.

2.2.10 Closed Circuit Television Policy

Close Circuit Television's (CCTV) should be used in the data center as a security device and also a deterrent. Cameras should be placed throughout the data center to ensure that all critical Billing4Rent systems are monitored.

The CCTV camera output should be recorded onto digital storage, which eliminates daily tape changing and storage.

Digital Video Recorders should be networked to allow stored or live images to be viewed over the Billing4Rent network by multiple users.

The frame per second (fps) record rate should be a minimum of 15 fps. The higher the fps rate is, the more natural the footage will look when played back. The lower the fps rate the more footage you can fit on the hard drive.

2.2.11 Service Pack and Patches Policy

The Billing4Rent network administrator shall be responsible for maintaining a list of software, service packs and patches for all Billing4Rent machines and ensuring all machines are patches appropriately.

The Billing4Rent network administrator shall be responsible for the implementation of a sign in/out policy for all software CDs/DVDs.

Relevant software vendor sites (operating systems, web servers, databases etc.) should be checked periodically for a list of updates. The Billing4Rent network administrator shall ensure service packs and patches are rolled out to all machines.

2.2.12 Audit Policy

Billing4Rent shall document the organisations processes for monitoring the integrity and availability of Billing4Rent ASP hosts in the Billing4Rent *Audit Policy*.

2.2.13 Acceptable Use Policy

The following activities are, in general, prohibited. Employees may be exempted from these restrictions during the course of their legitimate job responsibilities (e.g., systems administration staff may have a need to disable the network access of a host if that host is disrupting production services). Under no circumstances is an employee of Billing4Rent authorised to engage in any activity that is illegal under local, state, federal or international law while utilising Billing4Rent-owned resources.

The lists below are by no means exhaustive, but attempt to provide a framework for activities which fall into the category of unacceptable use.

2.2.13.1 System and Network Activities

The following system and network activities are strictly prohibited, with no exceptions:

- Violations of the rights of any person or company protected by copyright, trade secret, patent or other intellectual property, or similar laws or regulations, including,

but not limited to, the installation or distribution of "pirated" or other software products that are not appropriately licensed for use by Billing4Rent.

- Unauthorised copying of copyrighted material including, but not limited to, digitisation and distribution of photographs from magazines, books or other copyrighted sources, copyrighted music, and the installation of any copyrighted software for which Billing4Rent the end user does not have an active license is strictly prohibited.

- Exporting software, technical information, encryption software or technology, in violation of international or regional export control laws, is illegal. The appropriate management should be consulted prior to export of any material that is in question.

- Introduction of malicious programs into the network or server (e.g., viruses, worms, Trojan horses, e-mail bombs, etc.).

- Revealing your account password to others or allowing use of your account by others. This includes family and other household members when work is being done at home.

- Using a Billing4Rent computing asset to actively engage in procuring or transmitting material that is in violation of sexual harassment or hostile workplace laws in the user's local jurisdiction.

- Making fraudulent offers of products, items, or services originating from any Billing4Rent account.

- Making statements about warranty, expressly or implied, unless it is a part of normal job duties.

- Effecting security breaches or disruptions of network communication. Security breaches include, but are not limited to, accessing data of which the employee is not

an intended recipient or logging into a server or account that the employee is not expressly authorised to access, unless these duties are within the scope of regular duties. For purposes of this section, "disruption" includes, but is not limited to, network sniffing, pinged floods, packet spoofing, denial of service, and forged routing information for malicious purposes.

- Port scanning or security scanning is expressly prohibited unless prior notification to Billin4Rent is made.

- Executing any form of network monitoring which will intercept data not intended for the employee's host, unless this activity is a part of the employee's normal job/duty.

- Circumventing user authentication or security of any host, network or account.

- Interfering with or denying service to any user other than the employee's host (for example, denial of service attack).

- Using any program/script/command, or sending messages of any kind, with the intent to interfere with, or disable, a user's terminal session, via any means, locally or via the Internet/Intranet/Extranet.

- Providing information about, or lists of, Billing4Rent employees to parties outside Billing4Rent.

2.2.13.2 Email and Communications Activities

The following email and communications activities are strictly prohibited, with no exceptions:

- Sending unsolicited email messages, including the sending of "junk mail" or other advertising material to individuals who did not specifically request such material (email spam).

- Any form of harassment via email, telephone or paging, whether through language, frequency, or size of messages.

- Unauthorised use, or forging, of email header information.

- Solicitation of email for any other email address, other than that of the poster's account, with the intent to harass or to collect replies.

- Creating or forwarding "chain letters", "Ponzi" or other "pyramid" schemes of any type.

- Use of unsolicited email originating from within Billing4Rent'snetworks of other Internet/Intranet/Extranet service providers on behalf of, or to advertise, any service hosted by Billing4Rentor connected via Billing4Rent'snetwork.

- Posting the same or similar non-business-related messages to large numbers of Usenet newsgroups (newsgroup spam).

2.2.14 Email Policy

The general public tends to view emails sent from Billing4Rent addresses as an official policy statement. The following procedure should be adhered to in order to protect Billing4Rent's public image:

Reasonable usage of Billing4Rent resources for personal emails is acceptable, but non-work related email should be saved in a separate folder from work related email.

- The Billing4Rent email system shall not to be used for the creation or distribution of any disruptive or offensive messages, including offensive comments about race, gender, hair colour, disabilities, age, sexual orientation, pornography, religious beliefs and practice, political beliefs, or national origin. Employees who receive any emails with this content from any Billing4Rent employee should report the matter to their manager immediately.

- Sending chain letters or joke emails from a Billing4Rent email account is strictly prohibited. Sending virus or other malware warnings and mass mailings from a Billing4Rent email account should be approved by a member of the Billing4Rent management team prior to transmission. These restrictions also apply to the forwarding of mail received by Billing4Rent employees.

- Monitoring of Billing4Rent email accounts is purely at the discretion of the organisation and thus employees shall have no expectation of privacy in anything they store, send or receive on the company's email system.

2.3 Enforcement

Any individual found to have violated this policy may be subject to disciplinary action, up to and including civil and/or criminal prosecution to the full extent of the law and where applicable termination of employment.