Browser Security:
A Requirements-based Approach

by

Nwe Nwe Hlaing

Bachelor of Veterinary Science (*Burma*) - 1981

Thesis submitted in accordance with the regulations for
Master of Information Technology

Information Security Research Centre
Faculty of Information Technology
Queensland University of Technology

March 2003

Keywords

Browser security, security evaluation, the Common Criteria, Protection Profile.

Abstract

A browser is a convenient way to access resources located remotely on computer networks. Security in browsers has become a crucial issue for users who use them for sensitive applications without knowledge of the hazards. This research utilises a structure approach to analyse and propose enhancements to browser security. Standard evaluation for computer products is important as it helps users to ensure that the product they use is appropriate for their needs. Security in browsers, therefore, has been evaluated using the Common Criteria. The outcome of this was a security requirements profile which attempts to formalise the security needs of browsers. The information collected during the research was used to produce a prototype model for a secure browser program. Modifications to the Lynx browser were made to demonstrate the proposed enhancements.

Contents

List of Figures

List of Tables

Statement of Original Authorship

The work contained in this thesis has not been previously submitted for a degree or diploma at any higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

Signed: ..... **Date:** 22/8/03

Previously Published Material

The following paper has been published or presented, and contains material based on the content of this thesis.

Conference Paper

Hlaing, N., Clark, A. and Henderson, R., "Workstation Security: The Need for Evaluation", Proceedings of 1st Australian Workshop on Security Management (AISM), Deakin University, Geelong, 7 November, 2000, (ISBN 0-73002-796-1)

Acknowledgements

I would like to thank the people who helped me complete this study and make me feel confident in my ability. In particular, I sincerely thank my supervisors, Dr Andrew Clark for his patience and for enlightening me with IT security knowledge; Dr Mark Looi for guiding the research direction; and Dr Rose-Marie Henderson for helping me with my understanding of the CC and academic writing. I also wish to acknowledge that the security requirements profile in Appandex A was joint work with Rose-Marie Henderson and Andrew Clark.

I would also like to say thanks to the ISRC and its people for their help in many ways, from just being friends to giving me encouragement when I needed them. Special thanks also go to friends Brian and Pat Streeter for their valuable assistance with the grammatical editing work.

I truly thank my family, as without them I would not be able to achieve this goal.

Finally, I give enormous thanks to my parents for supporting and educating me for the whole of my life.

To my parents

Chapter 1

Introduction

Today's society is truly reliant on the Internet. The Internet has everything for everybody, for example Internet games for fun-loving young people, reference materials for studious people, news for people who want to keep up-to-date, electronic commerce for business people and their customers, instant communication for governments and businesses, and the list goes on. Making use of Internet technology, an organisation can set up a private network, an intranet[1] or an extranet[2], where core business functions can operate. With an intranet, organisations can gain numerous benefits, such as online meeting arrangements, simultaneous email delivery, business database accessibility, online directory access, processing order forms, and so on.

Users need a browser to access servers residing on local networks, intranets, extranets, or the Internet. A browser is an application that allows users to gain access to private and public networks, and obtain benefits from additional features such as accessing email servers, transferring non-text files (e.g., audio, video, etc), and database access.

Browsers were implemented for use in accessing shared information within an organisation where security was not considered important [27]. Millions of computers are now connected across the world and are used for transferring sensitive information. To access remote servers, the only requirement is a computer, which is becoming

[1] A network belongs to an organisation, usually a corporation, accessible only by the organisation's members, employees, or others with authorisation [30].

[2] An extranet provides various levels of accessibility to outsiders if they have a valid username and password, and their identity determines which parts of the extranet they can view [29].

cheaper and cheaper, and a browser that can be obtained freely and operated easily. In the light of these availabilities, some people use the Internet without understanding how vulnerable they are. Perhaps surprisingly, people with knowledge of the potential exposure to threats on the Internet still use it because the Internet and browsers offer simple, efficient, and effective ways of communicating and trading.

An attacker could access information during transmission between two machines (which includes computers, switches, firewalls, routers, etc) [34]. Incorrect information or data coming from the Internet can cause serious problems as it could lead toward financial losses or affect a sensitive issue such as the relationship between two countries. In addition to this, flaws in enhancement features embedded into browsers have caused serious security breaches. It has been shown that Java, for example, could execute arbitrary machine instructions, could initialise a connection with an arbitrary host, could bypass the Java security manager, and was vulnerabe to denial-of-service attacks [16]. A further example is JavaScript, which can read or upload files stored on the user's machine, intercept the user's e-mail address and password, monitor the user's activities and capture the URLs the user viewed, and then transmit information collected to malicious servers on the Internet or across browser frames allowing one browser window to spy on others [40]. A flaw exists in the Windows Script Engine for JScript[3] can exploit by an attacker by constructing a web page that would execute code with the user's privileges. This could happen when the user visits that web site or the user are sent a link to that web page directly in an email [64]. Another security problem in browsers comes from ActiveX[4], because they are native code and have nothing standing between them and the operating system. Therefore it is important for browsers that they always communicate with the server that delivers safe code. Patches for some of these problems are available but many administrators do not apply them to the networks they are responsible for in timely fashion, if they apply them at all.

There is another security issue which arises where authentication is necessary in end to end communication. Modern browsers have an embedded security feature, the Secure Sockets Layer protocol, whereby a browser and a server can mutually authenti-

[3] JScript is the Microsoft implementation of the European Computer Manufacturers Association (ECMA) 262 language specification standard [65].

[4] ActiveX is a technology developed by the Microsoft Corporation for distributing software over the Internet. It can be embedded in a Web page for a scrolling marquee, a background sound generator, and an ActiveX control to execute Java applets [11].

cate using certificates issued by a Certificate Authority (CA). In general, users assume that they are communicating with a server they intend to, and this hypothesis is backed up by modern browsers by displaying a confirmation text message or a picture of a lock appearing [43]. If the server's certificate is forged or browers are tricked into talking to a different web server than the intended one (known as server spoofing), then the security supported by TLS will fail and it is likely that the users will not know (as the browser will still display the confirming text or a lock) whether the server they are communicating with is the actual server or a malicious one [43]. The Internet Explorer has some vulnerabilities effecting certificate validation. The two newly discovered vulnerabilities show that both of which could enable an attacker to spoof trusted web sites [66]. In the first vulnerability, digital certificates from web servers are validated without checking verification that the certificate has not expired, verification that the server name matches the name on the certificate, and verification that the issuer of the certificate is trusted. The second vulnerability could enable a web page to display the URL from a different web site in the IE address bar during a valid SSL session with the impersonated site. Both vulnerabilities could be used to trick a user to connect the attacker's web site and provide sensitive information. However, this vulnerability only affects how certificates from web servers are validated. It does not affect on signing certificates or any other type of certificate are validated. The vulnerability also need to provide a way to force users to the attacker's web site such as DNS spoofing or similar attack [66]. To avoid these undesirable circumstances, security in browsers must be addressed sufficiently and accurately.

To address security in browsers effectively involves a number of steps. First of all, an understanding of the browser functionality is necessary in order to identify where security is required within the browser. The thesis therefore begins with discussion on the functionality of the browsers in terms of protocols used and services provided by browsers.

Secondly, a standard evaluation for browsers is conducted using the functionality of the browsers. Standard evaluation for a computer product is important as it helps users to find out whether the product they use is appropriate for their needs. Security in computer products has been evaluated since the 1970s and the standard practice for security measurement has involved identifying threats and countermeasures [9]. As such,

this thesis discusses security issues in browsers by identifying the potential threats and countermeasures for browsers. These standard security evaluation mechanisms have been moving towards the Common Criteria for information security evaluation (CC) initiated by the International Organisation for Standardisation (ISO). Since the CC is recognised internationally as a standard for security evaluation, the security of current browsers was assessed using the CC's methodology and outlined in detail in this thesis.

In the next step, the data transferring protocols used by browsers are examined in order to identify security issues. Browsers use a variety of protocols for network connection and downloading data. Hence it is important to examine these protocols as most security issues in browsers are related to the connection and the downloaded data. This research therefore selected and analysed the most commonly used protocols found in current networking applications. In particular, the Hypertext Transfer Protocol over Transport Layer Security Protocol (HTTP/TLS) is selected for the analysis, because it has been implemented in all browsers and its use will only increase as people need secure connections to protect their private information (for example an account name and password for online banking). The research also analysed some other file transfer protocols and the outcomes of that analysis are included in Appendix B. These alternative protocols are File Transfer Protocol (FTP); Secure Copy with Secure Shell protocol (SCP/SSH), that has already been widely used for establishing a secure connection; and File Transfer, Access, and Management protocol (FTAM), standard protocol to be used between different file system environments.

Finally, the research makes use of the security evaluation results in developing a prototype model for a secure browser program. An existing browser was modified to include the identified enhancements. A summary of chapters in this thesis is given in the following section.

1.1 Organisation of the Thesis

Chapter 2 starts with a history of browsers in terms of motivation, innovation, and progress made along the line. This chapter then analyses browser functionality in relation to security and identifies three mechanisms; a connectivity mechanism, a data transferring mechanism, and a data presentation mechanism. The core mechanism of a

browser is the connectivity by which users are able to connect to a remote server. For the connection and transferring mechanisms, a browser employs the TCP/IP protocol suite and has certainly inherited all the advantages and disadvantages of TCP/IP. The analysis of browser functionality is useful for identifying where security is required in browsers. By following the systematic, conventional way of identifying security measures, this thesis discusses security in browsers by determining threats, vulnerabilities, and countermeasures. There are always threats when a browser connects to a remote server and downloads data, and they are critical if inadequate security is employed to the host operating system. Common countermeasures in use today are: deploying secure protocols for connection and data transferring, scanning contents of the downloaded file, and creating trusted operating environments by making use of host security tools. When a security tool for each protection is deployed, problems arise as the number of software flaws increases proportionally. Also many tools require human interaction and a considerable amount of time to operate.

As mentioned above, evaluation is important as it helps users to measure security in browsers. Therefore Chapter 3 introduces the Common Criteria (CC), the international standard for security evaluation. Before the CC was available, there were four standards for security evaluation but they are recognised only by a specific geographical area, for example the Trusted Computer System Evaluation Criteria (TCSEC) had been employed in the United States and the Information Technology Security Evaluation Criteria (ITSEC) had been exercised in European countries. As they were not accepted by other countries, it became a dilemma when computer products were imported. The ISO solved this problem by developing an international standard, known as the Common Criteria, in the early 1990s. The CC defines three constructs - a protection profile (PP), a security target (ST), and packages - allowing product developers and users to be able to identify a specific evaluation set for their product.

This research followed the approach recommended in the CC for developing protection profiles and produced a *Security Requirements Profile (SRP)* for Browsers. Chapter 4 shows details of the sequence of processes used during the profile development and the selection criteria for the chosen security components of the profile. The profile, as the CC standard, contains six sections: introduction; "Target Of Evaluation" (TOE), a definition of a product or system; security environment of the TOE;

objectives of the profile; security requirements; and rationale. The profile is attached in Appendix A.

The research then assessed the most commonly used current browsers and the services they employ from the host operating systems. The findings of this evaluation are included in Chapter 5. Netscape and Internet Explorer have been chosen as they are the most popular commercial browsers, and other selected browsers are Opera and the Lynx text browser. Since browsers make use of user identity and access control mechanisms provided by the OS, Windows 2000 and RedHat Linux version 7.3 have also been selected for evaluation with respect to these two features.

This research has found that security in browsers is inadequate for what they are used for, such as in current vital business operations. However, this research shows that an existing browser can be modified in order to implement an enhanced secure browser with which management and enforcement of browser security can be achieved easily. Hence, in Chapter 6, the research introduces a prototype model presenting a framework for a secure browser program that can be used to securely communicate with a remote server. The model consists of three mechanisms providing access control on the program, mutual authentication between a server and a user, and a secure channel between hosts. Some components will interact with the underlying operating system in order to complete their tasks. Each component of the model is discussed in terms of functionality, and interaction with other components or the underlying operating system.

Once the model is designed, it is necessary to experiment with a browser to demonstrate the results of the research. Therefore the source code of the Lynx browser was modified to implement some security features identified by the design requirements such as trusted channels between browsers and servers, user data protection, and security audit. The research experiment shows that the enhanced features can make browsers much more secure and that the required security features (highlighted by this research investigation) are achievable.

The conclusions, Chapter 7, provides a summary of the outcomes of this work and discusses further research possibilities.

Appendix A contains the Security Requirements Profile (SRP) for browsers developed by the thesis author and the research supervisors, Rose-Marie Henderson and

Andrew Clark. Appendix B discusses the common used file transferring protocols in a networked environment, namely File Transfer Protocol (FTP), Secure Copy (SCP) and Secure Shell (SSH), and File Transfer, Access, and Management protocol (FTAM). This discussion includes general functionality and overview analysis for each protocol, and the protocol comparison and analysis.

Chapter 2

Browser Security

A browser is a client application that is used to access information stored on servers. With browsers, users cannot only have access to private and public networks, they can also gain benefits from additional features which are embedded into browsers as enhancements and for users' convenience. Such features include accessing email servers, transferring non-text files (e.g., audio, video), and database access. Thereby browsers have become invaluable tools for network environments and e-commerce where customers can deal with businesses through the Internet.

As browsers have a simple interface and are easy to use, the number of users is increasing and these users have varying degrees of technical competency. In addition, the browser is used for a multitude of different applications. For example home users use a browser to perform daily functions (e.g., banking, buying goods, communications, etc) while government departments and businesses use them for their core functions. As more and more browsers are used in such functions, security is becoming a crucial issue for users and organisations alike. This is especially true for organisations whose users rely on browsers for performing business critical operations.

This chapter aims to analyse browser functionality as this will provide the necessary information for locating where security in browsers currently resides. Following a systematic approach used widely today, the security of the browsers is addressed by studying the threats, vulnerabilities, and countermeasures. The layout of the chapter is as follows.

This chapter consists of six sections. The first section, Section 2.1, discusses

the functions of browsers in terms of protocols used, common services provided by browsers, and associated applications. Section 2.2 highlights threats in the connection level, downloaded data, and host systems. Then browser vulnerabilities and countermeasures are examined in Section 2.3 and Section 2.4, respectively. This is followed by Section 2.5, which discusses the related works that aim to provide browsers with security in specific ways. Finally, Section 2.6 contains a summary of the chapter.

2.1 Overview of Browser Functionality

Browsers require a number of protocols to accomplish their various functions. These protocols fall into three groups according to the function they perform:

- *Connectivity*: The protocol suite commonly used for networking is the TCP/IP protocol suite.
- *Transferring data*: The protocols used for this function are HTTP, FTP, etc. In this work only HTTP is considered as it is used most widely.
- *Presentation*: Browsers can display files and their embedded pictures if the downloaded file is HTML. Other types of files may be handled by plug-ins, or helper applications (see below).

Browsers make use of services provided by the underlying OS and other applications, called "Helper Applications". The OS assigns access attributes to restrict access to browsers and their associated files. It also provides browsers with user identification and authentication mechanisms and some system calls. When browsers create user associated files such as history files, they use such system calls to obtain user ID, time, etc. Browsers then pass the created file to the OS for access control and storage purposes. For file types, browsers use the MIME mechanism to determine file type and if the file type is "text/html" then they display the file [20]. For other file types such as "application/pdf", browsers check the file name extension against the list of registered file types provided by the OS to find its associated application. If the associated application is found in the list, the file is passed to the application for further process; otherwise browsers prompt users to save the file. System administrators can modify

the registered file types list or add new file types into the list by entering a new file type extension and selecting an associated application from available applications displayed by the OS. Section 2.1.2 contains detailed information about helper applications.

As discussed above, browsers rely on the underlying OS for many services. Therefore, the browser security may be affected if the OS security is compromised by any malicious action such as virus. Thus it is important to ensure that the OS maintains an appropriate security level at least for the components which provide functions to the browsers, such as authentication mechanism. The research addresses the security level of browsers in Chapter 4 where the security level of the OS is discussed as well. The following sections outline the thorough analysis of the browser services.

2.1.1 Common Services Provided by the Browser

All services, supported by browsers, have been implemented with the aim to provide a basic functionality which can be described as presenting information obtained from the server. The data transferring service, for instance, allows users to download a file from a remote server. In response to the user's request, the browser first finds the specific server on the Internet/Intranet, establishes a connection between them, and then requests the server to transfer the file. The most common protocol utilised for communication between the browser and the server is the HyperText Transfer protocol (HTTP) which is discussed in greater detail in Section 2.1.1.1. The browser then obtains the file, and presents it to the user (or the interpretation/presentation program) in an appropriate format.

The sequence of events for downloading data from a remote server can be summarised as follows:

- User chooses a URL by clicking or typing directly into the URL field.
- The request is sent to a server (HTTP).
- The server sends back a response that includes HTTP headers.
- The browser uses the HTTP header to determine how the response will be presented, for example Content-type text/html will be displayed natively by browser

and other types may be presented through the use of a "helper application" (see Section 2.1.2 also).

When a browser requests a server to transfer a file, the transferred file potentially has to travel along physical connections passing through many computers (servers, switches, routers, etc). In general, these external systems are all unknown systems and their human controllers are unaccountable in terms of trust. For this reason the protocols used by browsers should be implemented with some security features to protect the data on transmission.

Browsers employ several different protocols to provide users the requested services. Figure 2.1 illustrates a browser, its communicating parties, and essential protocols. The services and associated protocols are also shown in Table 2.1 [6]. More information for some of the services is given in Section 2.1.1.1.

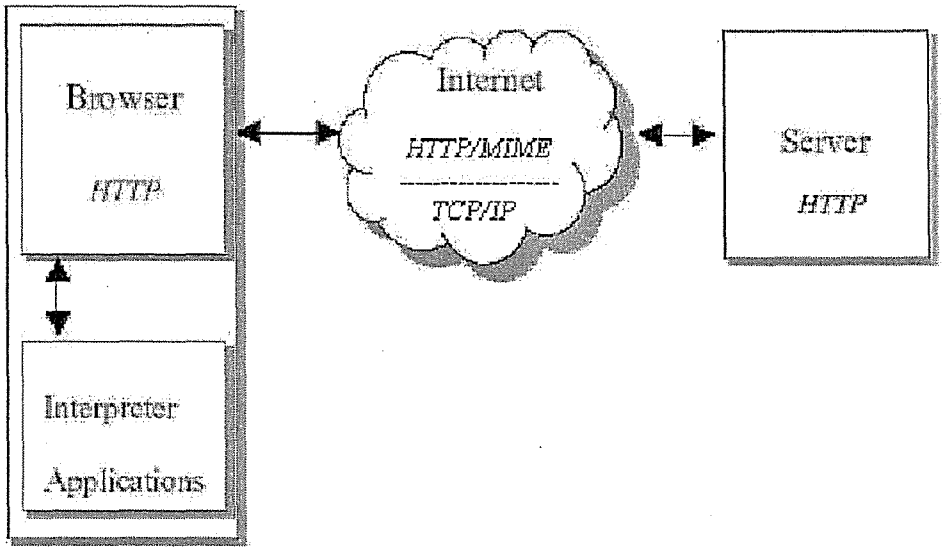

Figure 2.1: A browser, its communicating parties, and essential protocols.

The reader is referred to Chapter 4 for the description of a browser. The research is primarily interested in the data transferring service as it is a core service in browsers. However, modern browsers generally support many other services. This chapter therefore includes a brief discussion on other services for completeness.

Services	Protocols	Description
Data transferring service	FTP	File transfer protocol
	HTTP	Hypertext transfer protocol
Internet resources accessing services	WAIS	Database access
	Gopher	Database access
	Prosper	File and directory services
Mail service	SMTP	Simple mail transfer protocol
	IMAP	Internet message access protocol
	POP	Post office protocol
News services	NNTP	Network news transfer protocol
Interactive service	Telnet	Remote login protocol
	Chat	Interactive real-time communication
	Multimedia	Integrated service for audio visual data presentation
	Internet Telephony	Telephone service over data networks

Table 2.1: Browser services and their associated protocols

2.1.1.1 Data Transferring Service

This service satisfies users who access servers and request data transmission. Since this is a major service in browsers, a detailed discussion of the protocols used by this service and their alternatives have been included in Chapter 5. The following section summarises HTTP, the key data transferring protocol.

HTTP

The Hypertext Transfer Protocol (HTTP) [5] is an application-level protocol used to distribute information across the World-Wide Web. The first version of HTTP (HTTP/0.9) was a simple protocol for raw data transfer across the Internet. To improve the next version, HTTP/1.0 was developed in order to allow MIME-like mes-

sages to be formatted, to contain meta-information about the data transferred, and to include modifiers on the request/response semantics[1]. However, HTTP/1.0 does not include mechanisms for hierarchical proxies, caching, persistent connections, and virtual hosts. These features are implemented in the current HTTP/1.1 protocol and it becomes one of the core protocols for data transfer across networks [20].

Being a generic, stateless protocol, HTTP works for many tasks such as name servers and distributed object management systems. Its main function is to carry required information for establishing a communication between two hosts. Such information is used for negotiating application version, performing simple authentication of remote users, transferring data in hypertext format, and closing the connection. It can also negotiate data representation allowing any system to be built independently of the data being transferred [20]. To allow basic hypermedia access to resources offered by various applications, HTTP also acts as a generic protocol for communication between user agents and proxies/gateways to other systems (e.g., SMTP, NNTP, FTP, Gopher, and WAIS servers).

Methods	**Description**
Get	Retrieve data from the servers.
Post	Request the server that the enclosed information is to be treated as a new subordinate of the requested URI.
Put	Request the server that the enclosed information is to be stored under the provided URI.
Head	Request only for the header of the HTTP response to find the associated information of a web page.
Option	Request available methods on the URI (normally a server) in order to determine the capabilities of a server.
Delete	Request a server to delete the resource identified in the URI.
Trace	Request the previous request message to confirm that servers receive the correct request.

Table 2.2: HTTP methods and their Descriptions

[1] A detailed discussion of MIME is presented in Section 2.1.2.

When browsers send HTTP requests to remote servers, the first line of the message includes the method to be applied to the resource, the URI of the resource, and the HTTP version used for the request. An example of a request would be

GET http://qut.edu.au HTTP/1.1

There are seven request methods used in HTTP requests as shown in Table 2.2. The most used request methods are: "Get", "Post", and "Put". The "Get" method is used to retrieve data from the servers. There are three different formats used in "GET" method. The normal "GET" method retrieves the information identified by the requested URI. To reduce unnecessary network usage, browsers can use a "conditional GET" method[2] or a "partial GET" method[3]. If a browser sends a "conditional GET" method, the server will return the requested information only if the condition is met. If a "partial GET" method is requested, the server replies with the requested part of the entity. A security issue in the "GET" method is sending forms with sensitive data, which is encoded in the request URI. A server, proxies (if any exists), and the browsers will log the request URI in some place where third parties may have access. In this case the "Post" method must be used. The "Post" method is used when the server needs the enclosed information in the browser's request in order to response it. Sending a message to a bulletin board, newsgroup, or mailing list usually uses the "Post" method rather than others.

The "Put" method requests the server to store the enclosed information under the provided URI. This method is used for transferring files to servers. The difference between the "Post" and "Put" methods is reflected in the consequential processing of the information. "Post" contains a URI that identifies the resources for data processing, such as a CGI gateway. In "Put" method, browser users know the intended URI and the information must be stored only in this URI. The other methods are "Head", "Option", "Delete", and "Trace". Like the "Get" method, the "Head" method requests a resource but it asks only for the header of the HTTP response in order to determine the information associated with a web page, such as Content-Type. "Option" is used to request available methods on the URI (normally a server) in order to determine the capabili-

[2]The conditional GET method requests that the entity be transferred only under the circumstances described by the conditional header field, such as If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match, or If-Range header field.

[3]The request message includes a Range header field.

ties of a server, and an example of the correct response could be "Allow: Option, Get, Post". The "Delete" method requests a server to delete the resource identified in the URI. To confirm that servers receive the correct request, the "Trace" method is used and a browser should receive a response containing its own previous entire request message.

Many servers today (especially in e-commerce businesses) request browsers to use the post method when they need to obtain information from their clients. In this case a server embeds text messages, known as cookies, to record client ID assigned by the server, user visited links within the server, and other user associated information. From a security standpoint, cookies may violate the user privacy [32]. However, browser users can choose whether to accept all cookies, to deny all cookies, or to view the warning message asking the user acceptance for every cookies. Browser users can also control cookies through the browser configuration by listing the web sites that can or cannot store cookies on their computer. Since cookies originate from servers, it is also important to verify that a browser is communicating with a trusted server (using TLS connection).

When a server receives a request from a browser requesting a file, the server usually sends a response message which contains status code, access authentication scheme if the server uses one to challenge request, and HTTP header information. The status codes are grouped under four classes and they are indicated by the first digit of the code that can be 2, 3, 4, or 5. The code starts with 2, for example "200 OK", indicates that the client's request was successfully received, understood, and accepted; 3, for example "301 Moved Permanently", indicates that the browser has to take further action in order to fulfil the request; 4, for example "404 not found", indicates client error; and 5, for example "501 Not Implemented", indicates a server side error.

HTTP provides a basic authentication mechanism that uses a user identity and a password for each realm. The basic mechanism allows resources on a server to be partitioned into a set of protected areas where its own authentication mechanism is used. HTTP also allows browser users to send authentication information to a proxy using Proxy-Authenticate and Proxy-Authorization headers. It is important to note that the HTTP basic authentication mechanism is not secure method as it sends authentication information in clear text disclosing possibly sensitive information. Thus, it should only

be used in conjunction with additional security mechanisms such as transport layer encryption.

HTTP responses also contain a standard HTTP header with a number of fields. In fact, these fields carry information in relation to cache, allow methods, authorisation, connection, URI, Content-Type, date, host, etc [20]. Additional information for these fields can be found in [20].

The Content-Type field identifies the media type of the message body sent to the browsers, for example: *Content-Type: text/html.* Servers use MIME (Multipurpose Internet Mail Extensions) format for the Content-Type field in order to allow data to be transmitted in open representations. MIME was defined in 1992 by the Internet Engineering Task Force (IETF) [47]. The HTTP header also includes a MIME version field to identify the version of MIME used to construct the message. MIME version 1.0 is used by default in HTTP 1.1. An example of such a field is: *MIME-Version: 1.0.* MIME is discussed in greater detail in Section 2.1.2.

In general, the HTTP header is followed by the body of the HTTP message containing the browser requested file, and a response from servers is then completed. In Section 5.2.1, the functionality of the HTTP protocol over TLS is also discussed.

2.1.1.2 Other Network Services

Browsers also facilitate a number of services which allow users to exploit resources available in the networking environment. The service names and associated protocols are listed as follows. These services are not relevant to this research but are summarised here for completeness.

- Internet resource accessing service: this service enables users to access resources such as databases, directories, and files stored in servers. The protocols used for this service are WAIS, Gopher, and Prospero.

- Email service: the mail service allows users to read their e-mail stored in email servers and to send email. The protocols used are Simple Mail Transfer Protocols (SMTP) for sending mail and Post Office Protocols (POP) for retrieving mail.

- News service: the news service provides accessibility to Internet news groups to users who have subscribed. The Network News Transfer Protocol (NNTP) is

used in forums for news groups [39].

- Interactive service: This service enables users to interact simultaneously with other entities. Many users access the Internet in order to use interactive services such as Chat, Internet Phone, or Multimedia computer networked games.

2.1.2 Associated Applications

As browser technology has grown to support many types of data in an integrated way, many applications can be accessed and launched by browsers using information from the HTTP header. As discussed in the previous section, a server's response to a browser includes a HTTP header containing the Content-type of the file requested. If the Content-type is text/html, the browser displays the file, otherwise it is transferred to the associated application. These applications extend the capabilities of browsers in a specific way. For example, users are able to play audio samples or view video movies that are initiated by browsers (example applications are RealPlayer for audio and Windows Media Player for video).

As mentioned above, a mechanism that is used in browsers in order to identify file type is MIME, a specification for formatting non-ASCII messages that are sent over the Internet [21]. Servers use MIME to determine the content of the transferred file and it is included in their response to browsers as "Content-type" in the HTTP header. There are many predefined MIME types, for instance GIF for graphics files and PS for PostScript files. Examples for the Content-type are: text/html for normal text, application/pdf for Adobe Portable Document Format, and application/msword for Microsoft Word files. It is also possible to define new MIME types by registering them with the Internet Assigned Numbers Authority (IANA) [1].

There are two methods for accessing an associated application. Browsers operate differently according to the method of access. An application is labelled as a *plug-in application* if it behaves as part of the browser (open the file within the browser window), or it is called a *helper application* if the application handles the transferred data independently without interaction with the browser (the application is launched separately). When a browser encounters a MIME type, it searches for a registered plug-in first. If that particular MIME type is not registered, then it looks for a helper application [15]. Plug-ins add a specific feature or service to browsers so that they can

display different types of data format such as audio, video, text, and picture. Figure 2.2 illustrates the way a browser processes an HTML file containing specific embedded codes.

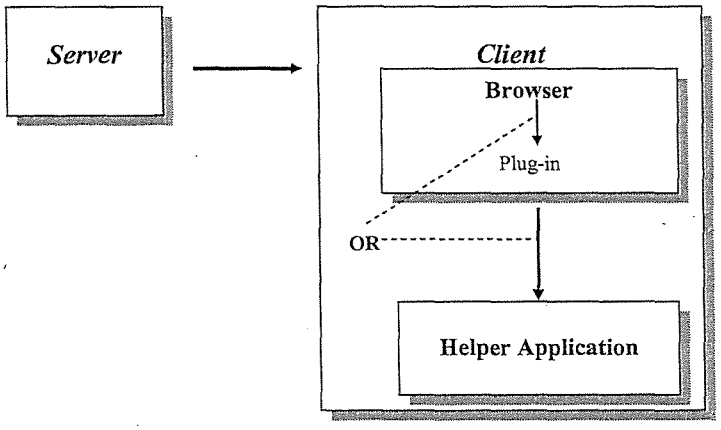

Figure 2.2: Data handling processes in browsers

Some popular plug-in applications are:

1. **Sun Java**: The Java Plug-in allows the execution of applications called "applets" that are written in the Java programming language [60]. When a user accesses a Java plug-in embedded web page, applets are downloaded onto the user's computer in order to process a particular function (e.g., online chat, calculating mortgage interest, images in 3D, etc). This technology is also used by organisations for intranet applications and inter-business communication.

2. **Apple QuickTime**: The Apple QuickTime plug-in supports QuickTime animation, music, audio, video, and Virtual Reality panoramas to be run on a Web page [37, 3]. A "fast-start" feature in the QuickTime Plug-in can display some part of the content while it's still downloading . It can be used within firewall and the server needs no special server software.

3. **Adobe Acrobat**: Adobe Acrobat allows user to view and print Adobe Portable Document Format (PDF) files on all computer platforms [37, 33].

4. **Macromedia Flash**: This program provides animation and entertainment on the

web. It can usually be found in web pages containing diagrams with animated picture and sound, cartoons, games, etc [37, 42].

5. **RealNetworks RealPlayer**: RealPlayer is designed for playing streaming audio, video, and animations on the web over narrow or broad bandwidth connections [37, 50].

In this section the various functions that a browser may perform have been explored. Now some of the security issues associated with that functionality are examined as follows.

2.2 Threats to Browser Services

Browsers have been implemented with services which aim to function proficiently in open system environments. This goal inevitably allows people to act illegitimately, and their actions may result in a violation of information security in browsers or the hosts they run on. Consequently a question arises: How can the browsers protect themselves and their associated data? To determine an answer to this question, security issues are often addressed by identifying threats and their countermeasures. This approach is widely accepted as being the most systematic available.

A threat can be described as the potential for abuse of protected assets. They can be caused by either authorised users or illegitimate users. The threats are now identified under the three major functionalities of browsers.

2.2.1 Connection Level Threats

Browsers rely on the TCP/IP protocol suite for connectivity. The Internet Protocol (IP) is unreliable (delivery of packets is not guaranteed). The Transmission Control Protocol (TCP) provides a connection-oriented, reliable service which runs on top of IP [54]. The "reliable service" provided by TCP only means that data is delivered to the specified destination and it cannot be implicitly taken that the delivered data is accurate and comes from the claimed source.

TCP uses the three-way handshake connection establishment protocol as depicted in Figure 2.3. In short, a client first sends a synchronise (SYN) message X to a server.

That message is followed by the server's response containing its SYN message Y and an acknowledge message (ACK) for X to the client. Then the client sends the server an ACK message for Y to successfully complete the handshake procedure.

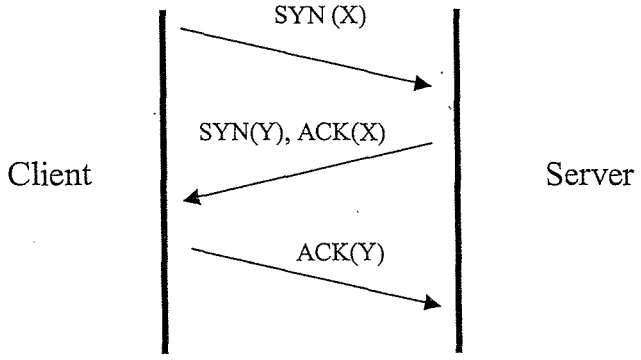

Figure 2.3: Three-way Handshake Process in TCP

The "reliable service" provided by TCP only means that data is delivered to the specified destination without providing the confidentiality and integrity of the data, and it cannot be implicitly taken that the delivered data is comes from the claimed source. This problem in combination with IP spoofing allows an attacker to initiate a connection while masquerading as another host [67].

Another well-known attack is a TCP SYN attack in which the attacking host sends numerous SYN messages to the victim host but no further ACK messages are sent [68]. This will result in half-open connections at the server end. This attack is normally used in conjunction with either an IP spoofing attack and/or a denial of service attack, in which the number of half-open connections exceeds the number the server can handle [63]. This SYN attack only affects at the server end and browsers are not affected by these attacks.

A problem affecting browsers is a server spoofing attack where browsers may not know that the server has been spoofed and the transmitted data has been forged. Browsers are usually more concerned with the integrity and secrecy of the transfered data as any system between two hosts is able to view and alter the passing data packets without being detected unless additional security measures are employed (because neither TCP nor HTTP provide an cryptographic integrity or confidentiality service). The

current protocol used to achieve data integrity and confidentiality is Transport Layer Security (TLS). Section 5.2.1 discusses TLS in detail.

2.2.2 Threats to Content Data

Browsers commonly use HTTP and FTP for data transmission and these two protocols carry any type of data provided by servers. None of the current browsers can filter based on the content type of the downloaded file even though accurate file types can be used to decide whether or not the downloading should continue. On the other hand servers could provide files as long as permissions set on the files are appropriate, for example when read permission is set.

As has been discussed in the previous section, data can be easily altered during transmission. It could happen along the wire or at transit computers such as routers, and there are no mechanisms to prevent or detect such changes.

The final threat to data security that is considered here lies on a mechanism that determines file content type. Browsers use Content-type information in the HTTP header sent by servers, and the server typically determines Content-type according to the file name extension (the extension is registered with IANA as discussed in Section 2.1.2). This could cause an undesired occurrence when the file name misrepresents the file content [10]. For instance if an executable code is embedded in an adobe acrobat reader file (.pdf), the browser will then inform the user the file type as "PDF", so the user will download and open the file. According to its embedded file content, the file could be executed on the local system with the privileges of the user who downloads the file [34]. This behaviour results in a vulnerability that allows attackers to bypass security measures.

2.2.3 Host Threats

Some functions in browsers require interaction with the underlying operating system. The significant example, in relation to security issues, is the user authentication function, and the access control function for browsers, their associated data and its corresponding applications. In short, the security level of browsers and their data are the same with the underlying operating system as they reside and rely on host computers in terms of security measures.

Furthermore, the operating system can fully support browsers with its security mechanisms only if the surrounding environment is trusted. A simple example is that password authentication would not be effective if authorised users are hostile, or attackers have somehow obtained a valid password, or attackers can enter into the system by other means.

The malicious code in the downloaded data can also threaten host security. As discussed in the previous section, an associated interpreter application is launched according to the downloaded data type; the data is then processed; and this may result in an unexpected situation.

2.3 Vulnerabilities

An understanding of vulnerabilities is required as it helps to construct security in browsers to overcome those vulnerabilities. These vulnerabilities are attributed to errors and only known errors (or vulnerabilities) have been remedied while unknown errors are still there to be exploited in future. The well known vulnerabilities are discussed in this section. Further information can be found in the references included.

2.3.1 TLS Vulnerabilities

Secure Sockets Layer (SSL) was developed by Netscape for transmitting data over a secure connection within networks. To form an Internet standard in the transport area, the IETF designed the Transport Layer Security (TLS) protocol that is based on the specifications of SSL [17]. TLS encrypts data that is transferred over the TLS connection. Many web sites use the TLS protocol to obtain confidential user information, such as credit card numbers. An TLS connection can be started by using "https:" in URLs (instead of "http") [22].

There are some vulnerabilities in the implementations of TLS. Many of them are concerned with certificates (which contain public keys) that are used for authentication and to generate a session key for data transmission. Many of the problems described here are not specific problems with TLS. Instead they typically result from errors made by the programmers implementing TLS.

When certificates are used for user authentication, the private key of the user must

be used. The private key is normally stored in encrypted form on hard disks, but it is decrypted and stored in memory when it is in use. Hence it may be possible that the private key can be recovered from memory if the browser has been modified or malicious software has been installed [57].

TLS uses a session key to encrypt a message for one session and therefore security of the session provided by TLS depends very highly on the session key. In June 1998 researchers at Bell Laboratories discovered that the session key could be found by analysing reports of a number of systematically structured messages from a web server [57]. This vulnerability is now fixed.

With default installations of some Windows operating systems, it is relatively easy for an attacker to add a malicious TLS certificate to the list of trusted root CA certificates using a variety of attacks, including application vulnerabilities and viruses. The attacker would only then need to redirect a browser's request to a malicious server which proxies TLS requests, for example the malicious server would first view data, record personal information for future use, and finally send the data to the user intended server [43]. This problem affected Microsoft IE and it can be fixed by applying appropriate ACLs to the related registry entry.

Another research effort showed that users of Netscape Navigator 4.7 or Internet Explorer 5.5 can be tricked by attackers providing users with forged certificates and a visual TLS symbol like padlock when no actual TLS connection is in use [70]. In this attack, users are shown usual TLS confirmation windows, users can view the server's certificate in the normal way, and a standard padlock is displayed as if the TLS connection has been established. Users can use this information to verify whether the connection is secure, but most users make a decision based on what the browser seems to be informing them. In this case, users may submit sensitive information via the insecure connection.

The problems discussed above are caused by administrators and browser implementers who lack good implementation practices and knowledge of cryptography. As these problems are whether exist or not is unknown to the browser users, the browser security evaluation by independent experts is extremely important.

For TLS by itself, it is an adequate secure connection protocol if it is configured properly and is used in conjunction with a secure operating system.

2.3.2 Java Applets and JavaScript Vulnerabilities

Java is an object-oriented language developed by Sun Microsystems. Java applets are Java programs designed to run inside the browser. Java applets can run on most computers because Java interpreters and runtime environments (known as Java Virtual Machines - JVM) are available for most operating systems [59]. Web programmers can implement HTML code (APPLET tag) to accommodate Java applets in web pages and Java applet programmers create Java system classes, called *java.applet package* to cooperate with the HTML tag. Like normal Java applications, Java Applets can be programmed to access files on the system or network resources by using appropriate Java packages[4]. Thus, additional security mechanisms are required for the Java applets running on the host computer (the browser's host). This mechanism has been implemented using the *applet sandbox* model [44].

In the sandbox model, the downloaded applets can still be executed on the host computer but cannot access its resources by default. However, an applet can request for access to the system resources; the sandbox checks the request according to the appropriate applet security policy (implemented by system administrators); and the applet cannot perform the requested action if the check fails. The sand box mechanism may also allow secure code (known as trusted code) to perform an insecure action if the browser user is informed of such an action and agrees to it. Nevertheless, applet code that is located on a local file system where all Java system classes are sited, is treated as trusted code [44].

Java applets in a sandbox are not allowed to make network connections to other hosts apart from the host from which the applets were downloaded. The same restriction is applied to accepting connections from network hosts. In addition, applets placed in a sandbox cannot start other programs on the host, load libraries, or define Java native method calls that are implemented for a specific machine language to have direct access to the underlying system. Java applets use threads, a part of a program that can execute independently of other parts, and the sandbox mechanism ensures that applets can only use and control the threads they have created.

There have been many security vulnerabilities found in the Java applet implementations. The security issues arising from Java applets have included an ability to execute

[4]These language system packages are, for example, java.io, java.net.*, etc [44]

arbitrary machine instructions, a vulnerability to denial-of-service attacks, an ability to make networking connections with arbitrary hosts, and an ability to bypass the Java security manager with hand-crafted byte code. With regard to the sandbox mechanism, the authors of [44] discussed a sandbox vulnerability found in Microsoft Internet Explorer 4.0 and Internet Explorer 5.0. In this vulnerability, the sandbox would allow a Java applet to operate outside its boundary and perform any desired action on the user's computer. This problem can be rectified by applying the related patch. Like the SSL protocol, security flaws in Java applets are often caused by applet programmers. Readers can find detailed information for Java applet vulnerabilities in [44].

JavaScript is a scripting language developed by Netscape to allow Web authors to design interactive sites (JavaScript can also interact with HTML source code). Although the language has some features in common with Java applets, it was developed independently. Problems caused by JavaScript are mostly related to violation of the users privacy. JavaScript can read and upload arbitrary files on the users machine, intercept the users e-mail address and preferences including password, monitor the users activities, capture the URLs the users viewed, and then transmit information collected to malicious servers on the Internet. Additionally, information can be leaked across browser frames allowing one browser window to spy on others [40, 62].

Both Java applets and JavaScript may continue to run (and compromise the system if malicious codes are downloaded) even after users leave the page which has originally launched these programs or close the browser windows that host these programs [57, 69].

2.4 Countermeasures

In browser security, countermeasures can be described as the mechanisms necessary to protect against the threats identified in Section 2.2. Once threats and vulnerabilities are identified, the next step is to analyse the available countermeasures supported by current technology. These countermeasures must be feasible in the operational environment. The following countermeasures are organised relative to the order of the identified threats in Section 2.2.

2.4.1 Secure Protocols

Due to the vulnerabilities in the TCP/IP protocol, strong secure protocols for network and transport layers have been implemented and widely used in an attempt to protect the transmitted data. Therefore the security requirements profile for browsers in Chapter 4 includes a component (Trusted path/channels), which uses a secure network protocol.

In the network layer, the latest version of the IP protocol, namely IPv6, introduces a new security architecture including data authentication and encryption mechanisms [12]. This new security architecture includes the IP security protocol that can be adapted to the older IP version 4 and is known as the "IPsec" protocol.

For the transport layer, the first implementation is Secure Socket Layer (SSL) protocol which is followed by Transport Layer Security (TLS) protocol. Netscape Communications Corporation followed IPsec in the move toward a security architecture and produced SSL to be used in Netscape browsers. SSL has made use of public key cryptography to authenticate the communicating peers and to establish a session key for data encryption [47, 22]. Once a session key is set the connection is transparently encrypted, and confidentiality and integrity of the transmitted data are attained.

To form an Internet standard in the transport area, the IETF designed the TLS protocol that is based on the specifications of SSL [17]. These two protocols consist of two layers: a handshake layer, where key certificates and a session key are exchanged; and a record layer, which handles data processing.

In order to make the transport layer protocols effective, some restrictions such as allowing SSL connections to trusted servers, using only port 443 for SSL connection, and limiting data amount for transfer, should be in place [43].

In the application layer, many protocols using public key cryptography have been created. The most widely used protocols are, for example, Secure HTTP, security Extension Architecture (SEA for HTTP), Secure Telnet (STEL), Secure MIME, Privacy Enhanced Mail (PEM), Pretty Good Privacy (PGP) for email, and Secure Electronic Transaction (SET) for E-commerce [47].

2.4.2 Content Scanning

There is no security architecture currently built into browsers that can be used to scan the contents of the downloaded files. As mentioned above, the MIME protocol determines file type based on the file name extension, and the file contents could be different from the general description given by the file type. For example a file name with an extension ".doc" containing executable data such as an embedded macro is labelled as document file by the MIME, but that may be executed when the file is opened.

As such, the users of the browser usually use a virus protection program which scans the downloaded file and reports the result. The anti-virus program checks the file for malicious data that is predefined by using known formats. Thus, the program doesn't help to corroborate that the file type given by the name is matched with the contents of the file.

2.4.3 Trustworthy Operating Environment

The operating environment falls into two categories: the environment of the host computer and the host computer itself (including the interactions between the underlying operating system, interpreter applications, and a browser).

Due to the fact that browsers reside and rely on host computers in terms of security measures, the host computers should be trusted and should operate in a trusted environment. If this essential trusted environment is unavailable, security in browsers is virtually nonexistent. In other words, the host computer can only be as trusted as its surrounding environment. Nevertheless, creating the physically trusted environment is out of the scope of browsers' capabilities, but administered by the human controller. Therefore the profile for browsers includes the trusted environment in the Security Objectives for the Environment, see Section 4.3.2 for detailed information.

From the technology perspective, there are some host security tools that assist administrators in detecting security violations. Some of the well-known host security tools are [47]:

1. **COPS**: a shell script that invokes the collected stand-alone programs that tackle a different security problem with UNIX systems. Problem areas include permission of files, directories, and devices; contents of password and cron files; and

contents of setuid and setgid script files.

2. **Tiger**: a script file that contains a set of commands to scan a UNIX system against a database of known vulnerabilities and security problems.

3. **TCP Wrapper**: a program that consists of small daemon programs which in turn monitor and filter any requests for various services in data communications.

4. **Tripwire**: an integrity checker program that can be used to determine whether stored data has been changed since the last check.

For file system security, the Andrew File System (AFS) mechanism can be used to encrypt files for storage purposes which effectively provides two security properties, confidentiality and integrity for data stored on hard drives [47].

For the whole system security, the host system should be assessed in terms of security measures (to identify the security functionality). From the evaluation standpoint, a host system can be assessed by using a particular security evaluation profile. In fact this research has chosen to use the Common Criteria for security evaluation in information technology for a browser because of its international acceptance. More information on the CC and a security requirements profile for browsers can be found in Chapter 3 and Chapter 4, respectively.

Many implementations addressing security in browsers have been developed to build a secure environment for them. In Janus [25] a tracing facility watches an application process and intercepts system calls. A configuration file is used by Janus to determine whether or not the system call is allowed to be made.

Using the Janus approach, another implementation has built a secure browser by assigning a sub-user id, which in turn determines what resources an object is allowed to access. In this implementation, all downloaded data is recognised as an object and is assigned with a sub-user id [55].

However these implementations are limited by the underlying operating system (available only for Unix platform). These implementations use a process protecting mechanism (providing multiple address space, allows trapping of system calls, and interposing proxies where necessary [25]), which is not available in MS DOS and Microsoft Windows.

2.5 Other Related work

A relevant work addressing overall security in browser is produced by Ioannidis et al and they call their implementation *a secure browser* [55] [56]. Major concern of the secure browser is user privileges in terms of accessing system resources. Therefore, a user is assigned with a new id, namely a "sub-user id" with minimum privileges restricting resource access. When a process initiated by the downloaded data requests access to system resources, the system takes the sub-user id as the process owner and therefore the system rejects the process's request. Using a sub-user id, the risk of malicious data attacks is low. Nevertheless, this browser cannot solve problems occurring during transmission such as eavesdropping and alteration (TLS could be used to prevent these threats).

While security in browsers has been addressed elsewhere, security in additional browser's features, such as scripting languages (for example Java scripts) and helper applications or plug-ins (e.g., Adobe Reader), has received the most attention and a number of different methods have been proposed [69, 25]. The script codes and helper applications always have the right to execute downloaded data while the data is in a processing state and that exposes a serious security issue. The common solution to prevent this occurrence is limiting their functions and separating them. This is achieved by creating a predefined database table that includes only permitted functions as discussed in the followings.

Anupam and Mayer create a safe interpreter for scripts by using a "padded cell" implementation. The script downloaded is isolated in the safe interpreter to prevent it from executing any unsafe commands that could compromise the system security. The safe interpreter also implement an access control mechanism to be used for the downloaded files within the script's own context [69]. For example, the data of the downloaded file should only have a read access. The interpreter isolates contexts from each other, so a user could send some information back to a specific server without being accessible by a script in a different context. The safe interpreter also disjoints multiple windows of a browser; therefore a malicious program in one window cannot access other windows, and thus cannot compromise the security of the browser. However, if scripts in different contexts require mutual access (e.g., an application running

in multiple windows requires the ability to access data in different windows), it must be allowed in trusted contexts.

While Anupam and Mayer build a safe interpreter for scripts, Goldberg et al create a secure environment to contain untrusted helper applications [25]. Since the data downloaded could have been created by an adversary and the helper applications are usually too complex to be bug-free [25], Goldberg et al create a "dispatch table" to reduce the risk of a security breach by restricting the program's access to the operating system. The table contains a list of system calls, associated values and functions, and is consulted when a helper application makes a system call. The decision on whether to allow or deny the call is made according to the table. Again, this implementation cannot solve the transmission problems.

2.6 Summary

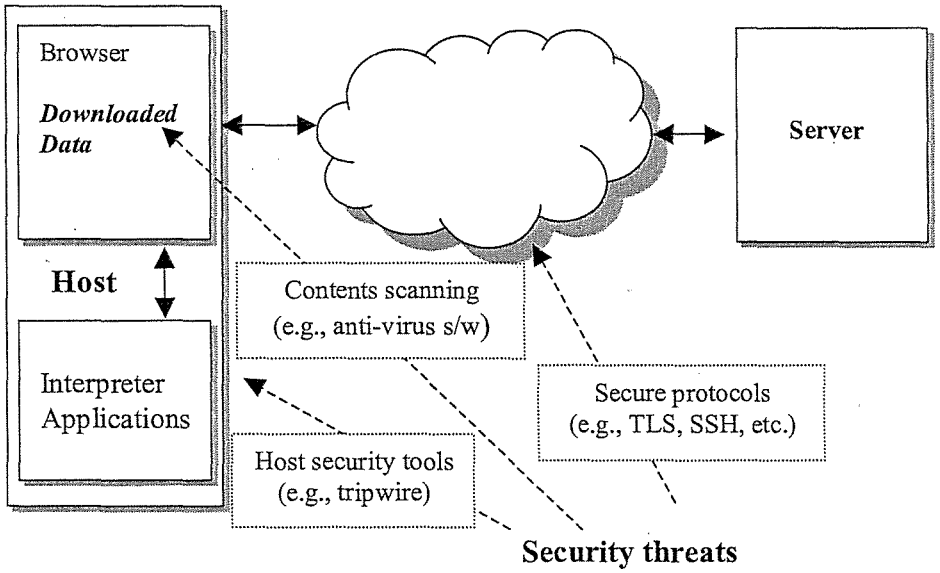

Figure 2.4: A browser, threats, and countermeasures

Browsers were invented with the aim to provide access to data stored on servers. Moreover, browsers provide a number of facilities such as electronic mail, electronic news group, web-based database access, telnet interactive login, to name a few.

Figure 2.4 illustrates a browser with its communicating parties, its threats, and countermeasures for those threats. According to Figure 2.4, threats in relation to browsers functionality are countered by security enhanced tools. Note that these security tools are limited to specific areas and are implemented by unrelated groups that may lead toward incompatibility in terms of cooperation.

When security issues in browsers are addressed, it is necessary to know how much security is provided by the browser. Thus it is required to evaluate browsers with a security measure. Hence, the next chapter inspects an evaluation mechanism (the Common Criteria), for determining whether or not browsers meet the security requirements of their users.

Chapter 3

The Common Criteria for Security Evaluation

Computers are playing an increasingly dominant role in every aspect of modern life. Computers were used firstly by the military, followed by research, business, and so on. Today, electronic communications and transactions are effectively utilising the power of globalised interconnection (the Internet).

Networking was created in research-oriented environments with the goal of free and open exchange of information [9]. In other words, the network pioneers were more interested in sharing information than securing it. Networks have grown dramatically: four computers were linked to form the first network and millions of computers are now connected across the world in what is called "the Internet". Today, organisations rely heavily on computer networks, thus security has become an important issue.

Computers with Internet access serve as a basic platform to complete daily functions (e.g., electronic banking, shopping, gambling, auctions, conferencing, etc) and are widely available (Internet cafes, libraries, etc). Internet technology is evolving from dial-up connections to permanent, high-speed connections. Therefore sensitive data (e.g., authentication data, information of the connected network) is increasingly exposed, heightening the need for protection.

The security level of a computer product can be determined by measuring it against security evaluation criteria. Evaluation can be used to determine whether a product or system meets security requirements. For evaluation people are moving towards the in-

ternationally recognised standard, the Common Criteria (CC), which is currently used in U.S.A, Europe, Australia, and elsewhere [46]. Due to that international acceptance, this research has used the CC to address security requirements for browsers.

The chapter begins by discussing the need for security evaluation of browsers in Section 3.1. Then Section 3.2 outlines the history of the CC. In Section 3.3, the CC's constructs are discussed, with which customers can express their security requirements for a particular computer product. The chapter then includes a discussion on the CC Relationship to Risk Analysis. Section 3.5 addresses the limitations of the CC approach and Section 3.6 provides the summary of the chapter.

3.1 The Need for Evaluation

Computer security evaluation is a process that measures the security level of a computer product by using a security measurement mechanism. In other words, evaluation justifies the level of the security claimed by a computer product. In common with other computer products, browsers are subject to threats. These threats come from the system itself, users, or the environment [9]. System threats come from hardware (e.g., hardware failure) and software (e.g., flaws or configuration errors). User threats occur in three ways: unauthorised access; unintended error; or privilege abuse. Environmental threats comprise natural disaster, power failure, unsatisfactory location, etc. These threats need to be mediated. The common approach currently used is to characterise the level of security required, and to ensure that the security measures used consistently support these desired security characteristics. Since security evaluation can measure the security level of a particular computer product, evaluation is believed by this research as the best way to identify the level of security in browsers.

Organisations require browsers with a degree of security to suit the nature of their business (e.g., while browsers used in a military agency may need the highest degree of security, moderate security would be suitable for a research centre in a university). Users in both environments desire evidence that the chosen security mechanisms meet their requirements. Evaluation results can help customers to determine whether browsers are suitable to use for their intended work.

Organisations cooperate for many reasons, for example a joint project, to increase

business availability, etc. Some organisations may choose only to interact with other organisations that use evaluated products (use of evaluated products is mandated in some government applications).

The alternatives to evaluation include risk assessment process, compliance with a quality assurance standard, and functionality testing. Organisations use risk assessment to identify key system assets and the threat likelihood (wherever possible) against each asset, determine the consequence/harm profile against each threat, and calculate the current risk for each asset. The acceptable level of risk for each asset/threat pair is then determined, and the priority of the associated countermeasure is established in the process [35]. ISO 9001 is the international standard for quality management systems. It includes eight components but only five are important for compliance: quality management system, management responsibility, resource management, product realisation requirements, and measurement, analysis and improvement requirements [19]. Functionality testing is to verify that the product performs as expected and documented [61]. This type of testing is beneficial to product developers who are creating a new product or an existing product which has undergone significant enhancements or changes in capabilities. The differences between evaluation and the alternatives are that the ISO 9001 and the risk assessment focus on the overall security of the organisation, while functionality testing is the product developer driven assessment.

The advantages of having an evaluated browser are significant. Users know exactly what the level of security is in their browsers and that inspires confidence in its performance. Evaluation guards against unjustified claims regarding security by individual vendors or organisations. In other words, evaluation provides justification of the claimed security characteristics of the browser. However, IT product vendors take advantage of evaluation by using the evaluation result. For instance, this has been done to promote sales of the Windows NT and Windows 2000 operating systems [49].

However, an evaluation rating has problems in comparing and representing. As the evaluation investigate the IT product to determine whether the product actually meets the requirements specified in its protection profile, the level of the assurance is varies according the profile. If the profile is written for low level security measure, for example, achieving high level assurance doesn't necessarily represent that the product is secure. Another drawback is that the same assurance level resulting from the use

of different profiles are actually different. Therefore, the users of the profile must understand these factors so they can make use of the evaluation result in an appropriate way.

3.2 Background of the CC

When people were discussing security in information technology in the 1970s, they accepted a need for an appropriate level of information security [9]. People in those days solved that problem by developing security models (e.g., Bell La Padula) [4]. To achieve the required level of security, the information security professional has the task of measuring the security of the systems and products they acquire. The problem is "how can we determine what level of security is offered?" The adopted solution to this question was to establish a set of criteria and an evaluation method operated by an independent body.

In the early 1980s, National Security Agency (NSA) developed the Trusted Computer System Evaluation Criteria (TCSEC), also known as "Orange Book" [9]. A decade later, various countries developed evaluation criteria on the basis of concepts of the TCSEC which are yet more flexible and adaptable to the evolving nature of IT [46]. In 1991, the European Commission, in a joint development of France, Germany, the Netherlands, and United Kingdom, undertook its own initiative and produced a set of Information Technology Security Evaluation Criteria (ITSEC), which contained assurance criteria without any specific functionality. In 1993, the Canadian Trusted Computer Product Evaluation Criteria (CTCPEC) were published. At the same time, NIST and NSA jointly developed the draft version of Federal Criteria for information technology security (FC).

The International Organisation for Standardisation (ISO) had begun developing an international standard evaluation criteria in 1990. The task was assigned to Working Group 3 (WG3) of Sub-Committee 27 (SC27) of the Joint Technical Committee 1 (JTC1) [58].

In June 1993, CTCPEC, FC, TCSEC, and ITSEC joined their activity to form a single set of IT security criteria under the CC project. This project cooperates with the International Organisation for Standardisation (ISO) and contributed several versions

of the CC to WG3 and these versions became various parts of the ISO Criteria version 1.0 in January 1996. The completed version 2.1, known as the International Standard ISO 15408, was published in August 1999 [45].

Many other countries are becoming involved with the CC due to the development of mutual recognition. Mutual recognition allows evaluations in one country to be accepted in other countries lowering costs for developers and consumers. Costs for evaluation before the CC are more prohibitive as different schemes (ITSEC, TCSEC, etc) placed different requirements on developers. The CC enables standardisation of evaluation which in turn allows mutual recognition. The current CC Version 2.1 supports a mutual recognition for Evaluation Assurance Level (EAL) from 1 to 4 between 12 nations [13]. New nations are preparing to join but not all intend to conduct evaluations (e.g., at this stage Japan will rely on evaluations carried out by other nations).

Australia is involved in the CC project with its Defence Signals Directorate (DSD). The Australasian Information Security Evaluation Program (AISEP) has been accepted as a full member of the Common Criteria Mutual Recognition Arrangement [28]. The aims of the CC project are summarised by quoting from the CC [46].

> *"This standard will permit comparability between the results of independent security evaluations. It does so by providing a common set of requirements for the security functions of IT products and systems and for assurance measures applied to them during a security evaluation. The evaluation process establishes a level of confidence that the security functions of such products and systems and the assurance measures applied to them meet these requirements."*

3.3 The CC's Constructs

The CC defines three types of constructs: package; protection profile (PP); and security target (ST) [46]. The package (e.g., an Evaluation Assurance Level) is a set of requirements that meets security objectives of IT products and systems. A protection profile is a complete set of functional and assurance requirements to address an identified set of security objectives [58]. PP includes reusable sets (a reusable definition of product security requirements that are known to be useful and effective [45]) and statements of "wants and needs" for security objects. ST is a set of security requirements provided by product developers, perhaps in response to a PP of a particular computer product.

Customers use a PP to describe their security requirements whereas developers use an ST to describe the security characteristics of their products. Hence, PPs are intended for IT consumers and STs for developers of IT products and systems.

Both PPs and STs address security requirements of a particular product or system operating in a defined environment.

3.3.1 Protection Profile

Protection profile states a security problem rigorously for a given set or collection of systems or products, known as the Target Of Evaluation (TOE) [45]. It also specifies security requirements to address that problem without defining how these requirements will be implemented. Therefore, a PP may be used to define a standard set of security requirements, with which products may claim compliance. A PP can apply to a particular type of product such as operating system, database, smart card, firewall or a set of products grouped together such as a system.

There is a guideline for production of PP available. The ISO released a draft version of guidelines for PP and ST in January 2000 [18] and the guide is intended to be fully consistent with ISO15408. The guide gives details of how to identify and specify the standard components of PPs such as the assumptions, threats, security objectives, security requirements, and rationale. There are reusable combinations of functional or assurance components called "packages". The purpose of a package is to reduce the cost and the work load of PP development. The guide also includes a checklist for PPs, STs, and examples for PP components. Furthermore, it includes sample security functional requirements for cryptography, and sample protection profiles for firewall, database, and trusted third party.

The CC describes the requirements for a protection profile. There are a number of steps, in accordance with the CC, to construct the structural outline of PP documents [2]. First of all, Target Of the Evaluation (TOE) must be described by type, scope, and operational environment. Security environment is then identified in three sections: security usage assumptions; organisational security policies; threats to security. Given the security environment, the PP developer will be able to determine the security objectives for the TOE, and the environment. Knowing the security objectives, functional security requirements for the objectives can be defined. Application

notes (additional supporting information) and rationale (the claims of the PP's completeness and cohesiveness) are optional parts of the PP as described in the CC. Figure 3.1 portrays the content of the PP.

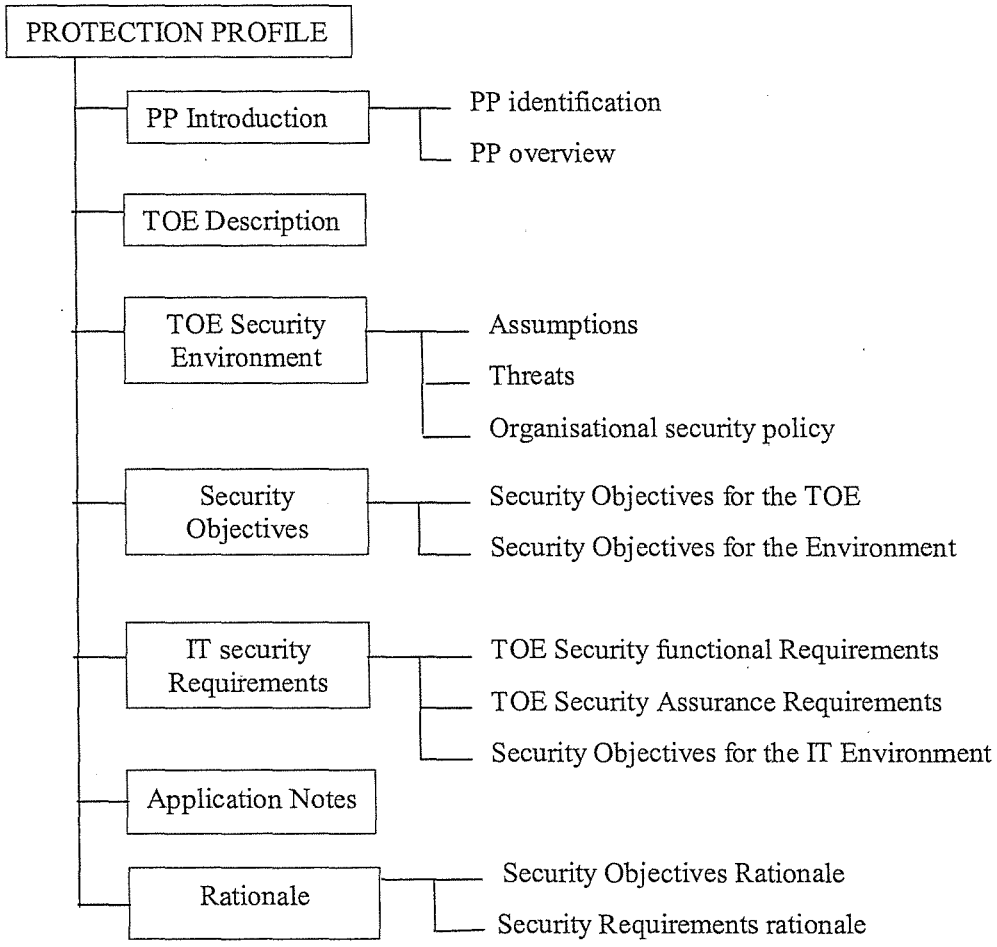

Figure 3.1: Components of the PP

3.4 The CC Relationship to Risk Management

There are many common themes in the CC and in risk management (e.g. both involve identifying assets and threats). However, their scope is not the same, as risk management has to cover the overall security requirements for the organisation (e.g., IT products, personnel, environment, etc) whereas the CC is more technical in focus

and covers only IT products and systems. The outcomes of the processes are different in usage as well. Risk management is performed to produce a security plan for the organisation and the CC is used to determine the assurance level of IT products and systems. Organisations own and control their risk analysis process whereas CC results are product specific and not likely to be controlled by the organisation. The CC can therefore not replace risk management but rather they complement each other.

3.5 Limitations of the CC and PP

The CC does not address organisation policies, assumptions, the administrative and legal framework, qualities of cryptographic algorithms, etc [46]. The use of the evaluation results in product or system accreditation is also outside the scope of the CC.

An evaluated product is not necessarily a secure product. Evaluation only addresses conformance to a PP. Therefore the consumer must be satisfied that the PP meets their needs. In order to be fully aware of the benefits of an evaluated product the consumer must completely understand the goals of the PP. While this may not necessarily be a limitation of the CC, it is an unavoidable limitation of evaluation in general.

The security functional and assurance requirements provided in the CC were not designed to be complete. Therefore authors of PPs and STs may introduce new security functional and assurance requirements as necessary. Another limitation is that when new features are added into a computer product, a new PP for the product must be written. This is a general problem associated with the scope of these types of documents (PPs and STs).

3.6 Summary

The CC is oriented towards specification and evaluation mechanism for IT products and systems, and permits comparison between the results of independent security evaluations. The CC is an international standard which is being increasingly adopted by vendors and customers.

The assurance level defined through evaluation inspires confidence in performance of the product. The security level of products is measured in a standard way, so that customers and product vendors can interpret the result.

Even though browsers are necessary tools for every business as they are used to connect servers and download sensitive information, there was no PP for file downloading (browsers) available at the time the research was started. In the next chapter the need for a PP for file downloading (browsers) is highlighted. Having a PP for file downloading (browsers), one can at least measure a security level of a program to determine whether the program is suitable for its intended purpose or not. In order to obtain the experience of developing a PP for browsers, the research used the CC's idea to produce a *security requirements profile* for browsers, and the experience of developing a profile is discussed in the following chapter.

Chapter 4

A Security Requirements Profile for Browsers

As was discussed in Chapter 3, the Common Criteria (CC) introduced the Protection Profile (PP) in order to identify essential security features in particular computer hardware and/or software products [18]. A PP is one way users can express desired security properties for their nominated information technology products. The PP does not, however, dictate how these requirements should be implemented. This allows computer users (who do not need to know how to implement computer products) to compose a protection profile in order to evaluate security in the chosen computer products.

At the time the research was started, there was no protection profile written for browsers (The U.S. Government and industry have produced a PP for Web browsers on April 21, 2001 [24]). Other relevant PP are the "Rudimentary Web Server Protection Profile" addressing security in web servers, developed by the Information Security Research Centre and "Web Server Protection Profile" issued by U.S. Government and industry [26].

In order to fulfil the needs for secure browser evaluation, the research had intended to compose a protection profile for browsers. Producing a protection profile is a time-consuming and complicated task. It requires large amounts of effort and resources. For example if a PP for browsers is going to be produced, it requires PP authors who should not only have knowledge in browser security, but also have an understanding of the PP concept and language. In addition, it also requires a group of people who

realise the usefulness of the PP, and they must devote a considerable amount of time to oversee and give the authors advice during the PP development process (involving a number of revisions to ensure that the PP maintains consistency and completeness). This research was conducted for a Masters degree and the available resources were limited (two supervisors and six months for developing a profile for browsers). Hence, it was decided that rather than undertaking the arduous task of developing a PP, it would be more beneficial to focus efforts on a less rigorous analysis which borrows some of the methodology used by the protection profile. To highlight this distinction the profile produced by this research is referred as a *security requirements profile (SRP)*.

This chapter contains a summary of the browser SRP. It discusses the approach used, the sequence of processes for the SRP development, and the selection criteria for the chosen security components of the SRP. Sections in this chapter follow the order of contents of the standardised PP document defined by the CC as in the previous chapter. Each section outlines what considerations have been taken into account and how component selections are made. The SRP is used later in the research to evaluate file transferring protocols used by a browser. The security requirements profile has been attached as Appendix A. The chapter also contains overview of the web browser protection profile (WBPP) developed by U.S. government and industry, and the comparison of the WBPP and the SRP.

Section 4.1 describes the browser by focusing on the browser's general functionality. Section 4.2 is then concerned with the security environment of the browser: it is important for a browser to reside in a secure environment where its security will not be compromised. In Section 4.3, user desired security features are depicted as objectives of the SRP. This is followed by Section 4.4 which lists security functionality components with regard to the objectives listed in the previous section. Section 4.5 discusses the reasons behind the chosen security assurance level for the SRP. The security assurance level describes the security functions that must be implemented in the browser to operate securely. Section 4.6 covers the rationale for the SRP in order to demonstrate that the SRP is complete and consistent. The chapter then continues to include an overview of the web browser protection profile and the comparison of the profile and the SRP in Section 4.7. The chapter completes with conclusions in Section 4.8.

4.1 Description of the Browser

The browser is a software application used to find files on a (remote) server and direct the server to fetch and transmit the requested files as appropriate. The browser provides users services such as making a connection to the requested file server, making requests for transmission of the desired file, and delivering the obtained files to an appropriate program on the requesting host for interpretation.

Browsers have been implemented without some security functions such as user identification and authentication. Browsers therefore make use of some security mechanisms provided by the underlying OS but there are still some security functions which cannot be obtained from the OS. The lack of the security functionality in early browsers made users seek additional security software with some essential security properties. The significant example for this would be connection level protocols, for example TLS protocol, used to attain integrity of transferred data. Now almost all browsers are distributed with a support for TLS. For these supplementary applications/protocols, readers are referred to Section 2.4.

The first section of the SRP is a statement describing the browser and the security problems encountered by the browser. It is necessary to have a precise description of browsers because later in this chapter security objectives are determined to suit the description provided here. In the CC this description is called the "Target of Evaluation". Note that in Chapter 2 the definition of a browser has been outlined in general and the description given here specifically focuses on security functionality rather than other aspects such as usability.

Figure 4.1 illustrates how a secure browser operates by showing a communication sequence taking place between a browser and a server, and showing interaction between a browser and an interpreter application. When the browser starts to download a file from a server, it first initiates a particular network protocol to communicate with the server. After a communication channel is successfully setup, the secure browser (proposed by this research) determines whether to proceed with downloading or to stop the communication according to the MIME type that is included in a HTTP header provided by the server. If the file type is permitted, downloading continues and the

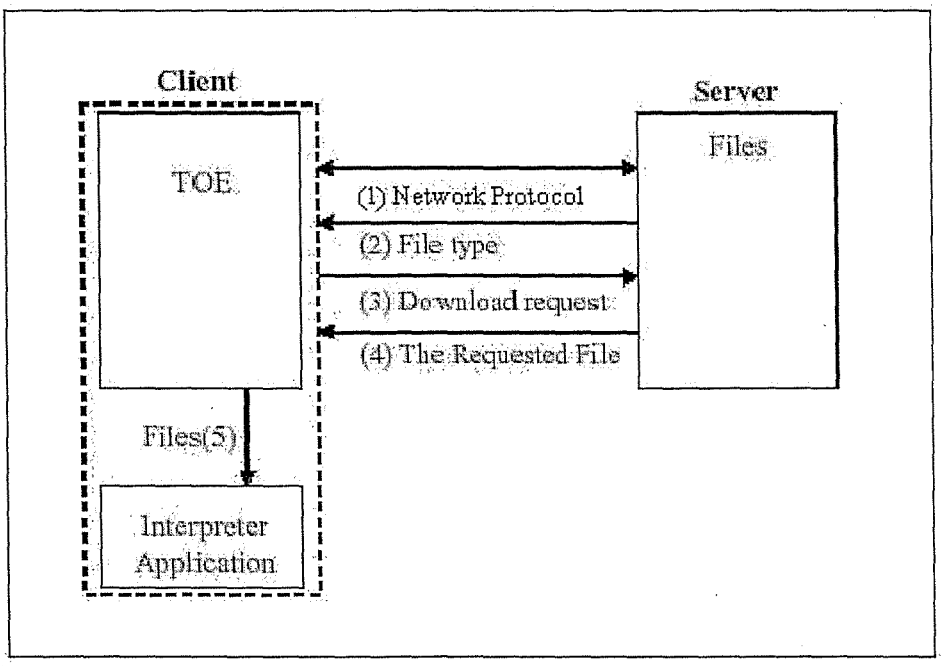

Figure 4.1: The browser architecture

browser will pass the file to an interpreter application.[1] In Appendix A Section A.2, a browser is officially described as follows:

> *The browser is a software application used to find files on a (remote) server and direct the server to fetch and transmit the requested files as appropriate. It provides user services such as making a connection to the requested file server, making requests for transmission of the desired file, protection of transmitted data, and delivering the obtained files to an appropriate program on the requesting host for interpretation. Figure 4.1 depicts the logical interaction between the client (with the browser and an interpreter application) and a server. It proceeds with the request for downloading of the file only if the file type is a permitted file type. It is configurable by the user or an administrator to prevent the download of specified file types.*

Having officially described the browser, the next task is to investigate a secure environment where the browser can operate without being compromised. The next section describes the security environment for browsers.

[1]Note that in some instances the interpretation functionality may be built-in to the browser. For example, browsers interpret HTML internally.

4.2 Browser Security Environment

Even if browsers have strong built-in security features and have made use of protection mechanisms provided by the underlying operating system, it is not guaranteed that they will remain in a secure state. This is because such security features and protection can be affected by an insecure environment that includes browser users, the communicating servers, and so on. For example, if a user changes the browser's configuration to cancel the requirement of a secure connection and connects to a malicious server, then the security of the browser is compromised. Thus, the environment where browsers operate should be taken into account when security of browsers is addressed.

According to the CC authors, the security environment for browsers can be identified by using three properties: secure usage assumptions, threats to the browser security, and security policies applied by the organisation that utilises the browser. The issues in security environment define the underlying security issues to be addressed when security objectives are identified in a later section. Reader can find a formal script of the browser security environment in Section A.3.

4.2.1 Secure Usage Assumptions

It is necessary to outline secure usage assumptions which inform the browser's implementers and administrators what is necessary to establish and maintain a secure physical and IT environment for the secure operation of the browser. With browsers, secure usage assumptions are concerned with user authentication and access control mechanisms. Given the fact that the browser operates on top of the operating system (OS), the browser's authentication mechanisms cannot stop malicious users accessing the browser if they have been successfully authenticated by the operating system. This practical consideration led to the decision that the browser is to rely on the OS's authentication mechanisms for the purpose of access control to the browser application itself. However, the browser can use IPsec (part of the OS and IP based authentication) or its own authentication functionality for the purpose of authenticating servers with which it communicates (see Section 4.3).

To maintain a browser in a secure state, there must be a reliable environment providing necessary security measures to the browser and the browser itself must use

security mechanisms in all areas of its operations. Accordingly, a detailed list of the assumptions of browser functionality and the surrounding environment is produced and the list can be found in Appendix A - Section A.3.1. They are summarised below:

- Access: All access to the browser and its data is controlled by the Operating System (including audit data).
- Storage: The Operating System provides secure storage for the browser itself and its data (including audit data).
- Authentication: The Operating System will authenticate a user prior to granting access to the browser.
- Identification: The Operating System provides the browser with all required user identification information.
- Malicious file: Downloaded files are checked by the Operating System to determine if they contain malicious data.
- Administrator: System administrators manage the browser and the information it contains, and can be trusted not to abuse their privileges.
- Server: The servers, with which the browser interacts, are regarded as trustworthy, especially for conveying and maintaining the correct file type, maintaining confidentiality and integrity of the file content.
- Operating system: The Operating System securely controls all information exchange with the browser.

Donaldson [18] states that assumptions should include intended usage aspects, connectivity aspects, personnel aspects, and environmental protection aspects. In practice, it seems virtually impossible to list all the assumptions in the first attempt. Hence, the listing of assumptions has been reviewed throughout the development of the SRP, and extended with additional assumptions. This happens especially when the rationale section is constructed. (The rationale contains arguments detailing how the security objectives encounter the identified threats.)

4.2.2 Threats to Security

Threats to a particular computer product should be identified through the normal "Risk Analysis" process [18]. The CC does not provide a framework for risk analysis and it is up to the author of the profile to decide how the threats are identified. However, the CC does provide the general principles involved in identifying threats; and they were employed in the development of this SRP.

In Chapter 2, threats for browsers have been identified under three areas: connection threats, threats to data, and threats to the host. Here the threats discussed in Chapter 2 are reiterated for the purpose of defining the SRP.

Following the CC, a threat, or undesirable event, has three components: an asset, a threat agent, and a method.

Asset: Information or resources to be protected by the countermeasures of the browser. The assets are not only of value to the owner but also potentially to the threat agents who seek to compromise the browser.

Agent: Either human or software application, their activities violate the security level of the assets.

Method: Possible attack methods in relation to potential vulnerability of the assets and capabilities of the attacker.

Table 4.1 represents the threat agents, methods of attack, assets, and the result of their interrelation. The threats outlined in the table are mainly caused by human users, either authorised users (normal users or privileged users i.e administrators) or intruders. Unauthorised users can access a browser and change a browser's configuration in order to download a file of illicit file type. If this unauthorised access occurs during data transmission then inaccurate data could be transferred. Even if browsers operate securely, it is possible to download inaccurate data if the connected server is spoofed or untrustworthy and a file's content and/or file type are changed. Authorised users, including normal users and administrators, would intentionally or unintentionally change the browser's configuration and this may allow users downloading a file with an illicit file type. Recognising that errors are always possible, it is accepted that both normal users and administrators may commit errors during their interaction with the browser.

Asset	Threat Agent	Method	Consequence effects
Data (C, I)	Unauthorised users	Unauthorised access	- loss of C (disclosure of the data) - loss of I (modification of the data) - Compromise browser security (download a file with illicit type)
Data (I)	Authorised users	Unintentional activity	- Compromise browser security (download a file with illicit type)
Data (I)	Authorised users	Intentional activity	- Compromise browser security (download a file with illicit type)
Data (C and I)	Software	Undiscovered flaw	- Compromise browser security (download a file with illicit type)
System (A)	Hardware	Failure (e.g., power supply)	- loss of A (interruption in downloading)

Table 4.1: Interrelation between assets, threat agents, methods, and effects.
(C = Confidentiality, I = Integrity, A = Availability)

Additionally, the browser can be implemented with undiscovered flaws which may allow users to compromise security by actions, such as downloading files with unpermitted file type. Thus, flaws in software (the OS and browser itself) can compromise security of the browser which in turn allows users to download disallowed file types. Threats defined in the SRP are included in Appendix A Section A.3.2.

4.2.3 Organisational Security Policies

According to the CC, any relevant organisational security policy is required to be included so that security objectives can be clearly identified [46]. However, the CC allows the statement of policies to be omitted if the security objectives are derived entirely from the threats [18] and this is the case in this work. It is necessary to note that any security issues exposed under the threats section should not be included as a policy, because it will simply be a restatement of a threat. There are certain circumstances where profile authors must specify policies, for example if the browser will be

used by a specific type of organisation. A good example that cannot be a threat is "an organization's security policy in regard to security audit" [18].
With regard to the security in browsers, there is only one requirement of the organisational policy. That is: *"Users shall be held accountable for their actions"*. By having such a requirement, the users can not perform any illicit action without it having been recorded along with their identity.

4.3 Security Objectives

In Section 4.2, the security environment was discussed in terms of assumptions, threats, and policies. Security objectives can be effectively identified by addressing these assumptions, threats, and policies [46].

Security objectives represent the security goals that the users desire the browser to achieve. These objectives are grouped based on their affiliations with either the browser itself or the browser environment. Section A.4 in Appendix A shows security objectives for the SRP.

4.3.1 Security Objectives for Browsers

Traditionally, IT security is concerned with confidentiality, integrity, and availability [9]. With browsers, availability is omitted, as it is not a major issue in the circumstance where the browser operates and access to a browser is controlled by the OS. However, integrity is a high priority since the browser must accurately obtain data from remote servers. Another high priority is confidentiality which may be required by some organisations (e.g., government, bank, etc) but not all users of the browser. Servers also need to authenticate browser for the purpose of access control.

Five objectives are identified in the SRP and their intentions are as follows:

1. *Authentication*. The means for providing an authentication mechanism to verify the identity of browser users by servers and vice versa. User and server authentication are required in establishing a secure connection between a browser (user) and a server.

2. *Confidentiality*. The means for ensuring that data or files can not be illegitimately obtained during transmission. Confidentiality is important because browsers

carry sensitive data (e.g., financial data, personal information, or national security information).

3. *Integrity*. The means for ensuring the integrity of downloaded files and data is maintained during transmission. Integrity is wanted because accurate data is as imperative as confidentiality in downloaded data.

4. *Access restriction*. The means for limiting access to the browser and downloading only permitted file types. This is to stop downloading dangerous file types (e.g., executable) in a proactive approach. Limitations are imposed by the administrator who can also override those of normal users.

5. *Audit*. The means for recording security relevant events to hold individual users accountable for the actions they perform. This objective aims to log security relevant events providing user accountability.

4.3.2 Security Objectives for the Environment

Without an appropriately secured environment it would not be possible for the browser to achieve its security aims. The browser environment includes not only interacting IT products or systems but also the human users working with the browser. Specifically, they are the underlying operating system, users and administrators, the connecting server, security on the browser installation and maintenance, and services that provide other necessary mechanisms for the browser operation. The following summarises the security objectives for the browser environment.

1. *Operating system*. The operating system should itself have an appropriate security level and provide security relevant information to the browser, for example access to browser data, secure storage for browser data, authenticating users before granting access to the browser, providing user identity information to the browser, and checking downloaded data for malicious data.

2. *Trained administrators*. Administrators must be adequately trained to manage and operate the browser.

3. *Servers*. Servers must be trustworthy.

4. *Browser operation.* The browser is designed, installed, implemented, configured, managed, and operated in a secure manner.

4.4 IT Security Requirements

The security requirements define "security needs" that the browser has to address. The IT security requirements are described in terms of the security functional requirements *(SFRs)* for browsers, security assurance requirements *(SARs)* for the browsers, and security requirements for the IT environment. The CC defines SFRs as the requirements for security functions provided by the TOE in order to achieve security objectives for the TOE. Examples of security functional requirements are requirements for authentication, access control mechanism, and security audit. The CC describes SARs as a specification of a strength level of the SFR that is consistent with the defined security objectives. For example, if the security function is a secure authentication mechanism using password, a SAR may specify that the minimum length of the password is coherent with the identified security objective for authentication. The security requirements in the SRP are defined using the catalogue of functional and assurance components detailed in the standard ISO/IEC 15408 Part-2 and Part-3, respectively [46].

4.4.1 Security Functional Requirements

The security objectives identified in Section 4.3 must be met by the security functional requirements and this can be achieved by selecting them appropriately from the list defined by the CC. If a requirement needed for a particular profile is not in the CC defined list, then the PP's authors can write the requirement to be included in the PP (but they must follow the CC's defined taxonomy). There are two types of security functional requirement in the CC:

- Principle security functional requirement - directly satisfy the objectives
- Supporting security functional requirement - indirectly satisfy the objectives

There are several stages involved in selecting SFRs. Some stages are necessarily iterative but overall this makes the process both complete and efficient. In the first stage, each objective must be addressed by one or more principle SFRs. At the second

stage, the process is iterated in order to select supporting SFRs that will provide more security functional components to the objectives.

The SRP is completed successfully with the security functional requirement components extracted from the CC.

The security functional requirements identified by this work to cover the browser security objectives are listed in Table 4.2 and these SFRs are described using the CC's methodology in Section A.5.1 of Appendix A. In order to choose the appropriate SFRs for each security objective, a major function to encounter the objective was identified first, and then suitable SFRs were searched in the SFR catalogue of the CC. For example, the audit recording was identified as a major function for the O.AUDIT objective, since it requires the TOE to provide the means for recording security relevant. The research then selected the SFR components such as FAU_GEN.1 to generate an audit record providing the required audit information, FAU_GEN.2 to link audit events and the users who cause the events, and FPT_STM.1 to obtain an accurate time for each security event. The security functional requirement components and their supported functions are summarised below.

Class	Functional Component
Security audit	Audit data generation
	User identity association
	Reliable time stamps
User data protection	Subset information flow control
	Simple security attributes
Identification and Authentication	User identification before any action
Trusted path/channels	Trusted channel

Table 4.2: The Security functional Requirements and their components

1. *Security audit.* Audit data generation defines the auditable events and generates an audit record providing the required audit information. In the record, user

identity is used to link audit events and the users who cause the event. Reliable time stamps then provide an accurate time that is recorded along with each audit event. The reliable time is critical for audit records as they assist administrators to detect the sequence or/and pattern of the security related events. By having such reliable audit records including associated users, administrators are able to detect the misuse or unauthorised activity of the users.

2. *User data protection.* Information flow control ensures that all user requests are subject to the identified security policy for file downloading. This policy allows users to download files with file type permitted by the administrator. Security attributes support the information flow control by restricting all users to the rules outlined in the SFR with regards to user identity, the file type and whether a secure connection for communicating the file is required. In short, the security attributes and rules for the information flow ensure that the file transmissions take place in accordance with the defined organisational security policies.

3. *Identification and Authentication.* User identification before any action ensures that users are identified before any other action is taken by the browser. Even though it is assumed that users are identified by the OS before they use browser, this requirement is necessary to include in SFRs because user identities are used in the secure audit SFR to associate user identities with the security events.

4. *Trusted path/channels.* A trusted channel provides a communication channel separated from other channels and assures the identification of connecting servers, while a trusted path between the browser and the user ensures that communication between them is protected from modification by or disclosure to untrusted applications. Using the trusted path/channels, the browser can securely identify user and also authenticate a server, thus a level of trust is built for the data downloaded from that server.

4.5 Security Assurance Requirements

The SARs describe how well the security functions are implemented in the browser (as compared to the security functional requirements which describe the security functions

the browser must provide). The selection of the SARs involves considering several factors. The major considerations are:

- *The value of the assets*. The greater the value of the assets, the higher the level of evaluation and thus the higher level of assurance is required.
- *Technical feasibility*. It must be practical to achieve specific assurance components.
- *Costs and time for development and evaluation*. The cost and time should be reasonable.

The security objectives also impact on the security assurance requirements needed for the browser. For example, if an objective states that the browser should be resistant to the attackers, then the security assurance requirements must have a component requiring such resistance to be demonstrated. Another example would be if the organisational policies determine the level of assurance of the browser to be used for their business, then the profile must have security assurance requirements in that assurance level. However, this policy determination can't be taken into account for the SRP as it has been developed with no intention to use in a specific type of business.

In general, the SARs catalogue starts from Evaluation Assurance Level (EAL) 1 to 7. The PP authors can select a desired EAL level from the predefined EAL list. The CC claims that the greater assurance is obtained from the greater evaluation effort. Thus, the goal is to apply the minium evaluation effort for the necessary level of assurance. The evaluation effort increases along with the portion of the IT product, the finer level of the design and implementation, and the structured formal manner. Therefore, there is a significant barrier to achieving higher assurance levels.

This SRP has been developed for circumstances where a moderate level of security is required. Thus for the security requirements profile, Evaluation Assurance Level 3 (EAL3) has been selected as that level includes high level design and configuration management control, testing for all parts of the browser, and documentation of the browser. The assurance classes and their assurance components included in the EAL3 (as described in the CC) are listed in Table 4.3 [46]. More information for SARs can be found in Section A.5.2 in Appendix A.

Assurance Class	Assurance Components
Configuration Management	Authorisation controls
	Browser CM coverage
Delivery and Operation	Delivery procedures
	Installation, generation, and start-up procedures
Development	Informal functional specification
	Security enforcing high-level design
	Informal correspondence demonstration
Guidance Documents	Administrator guidance
	User guidance
Life Cycle Support	Identification of security measures
Tests	Analysis of coverage
	Testing: high-level design
	Functional testing
	Independent testing - sample
Vulnerability Assessment	Examination of guidance
	Strength of browser security function evaluation
	Developer vulnerability analysis

Table 4.3: Security Assurance Requirements for EAL3

4.5.1 Security Requirements on the Environment

The requirements for the browser's environment are necessary for various reasons. One reason is because the browser will entirely depend on the underlying operating system or hardware devices for security operation in some way. Thus the requirements for that associated software and hardware should be addressed under this section. The CC states that the components in its Part 2 may not be appropriate to express the security

requirements for the environment. In this case, authors are allowed to compose their own components but they must provide justification for any deviation from the CC components.

The assurance in any security requirements provided by the IT environment must be the same or above to the security requirements provided by the TOE. For example, EAL3 set for an access control functionality would be undermined if the identification and authentication functionality provided by the OS is EAL2. Since EAL3 is chosen for browsers, the assurance requirement for IT environment is to be assured to EAL3.

This profile does not need any new security requirement components and this section in the profile was successfully completed by using the components of the CC catalogue.

The following requirements are selected as they are needed by the browser but are performed by the OS with which the browser interacts. It is necessary that the OS manages all of the configuration files and access to those files securely as the browser security depends upon this. The following subsections outline the security requirement components required for the browser SRP.

4.5.1.1 Identification and Authentication

Identification and authentication ensures that users are not allowed to perform any actions before they are successfully identified by the underlying OS. Thus only legitimate users identified by the OS can access the browser.

4.5.1.2 Security Management

Management of security attributes enforces a security policy that restricts access to the security attributes of the configuration files to authorised users. Security attribute initialisation is therefore required to enforce the security policy to provide either initial values or default values (if no initial values) for the browser and its associated files. Additionally, security roles need to be maintained (i.e., administrator, users) and users must be associated with the roles. In summary, setting a security attribute (at least) to the configuration file (and that can only be changed by a user with an appropriate role) contributes to the security of the browsers.

4.6 Rationale

The rationale section contains arguments demonstrating that the profile is complete, consistent and reasonable. It is achieved by verifying that the browser has provided effective security measures within its security environment [18][46]. Furthermore, the rationale corroborates that the security functional requirements and assurance requirements in the profile meet the security objectives and covers all aspects of the security environment. The rationale is briefly discussed in the following three subsections but the comprehensive discussion is given in Section A.6.

4.6.1 Security Objectives Rationale

Security objectives have been identified with regard to security environment defined in terms of assumptions, threats, and organisational policies. If it is possible to show that the objectives have covered all security needs as specified in security environment, then the security objectives rationale has been well achieved. In the SRP, all security environmental factors identified are addressed along with associated objectives (see Table A.4 and A.5). Thus it proves that the SRP is complete and consistent.

4.6.2 Security Requirements Rationale

The research has demonstrated that the selected security requirements are sufficient and necessary to meet the security objectives identified (see Table A.6 and A.7). Hence, it is appropriate to say that the SRP is complete and consistent. Take note again that one security objective can be countered by a number of security requirements and vice versa.

4.6.3 Security Assurance Requirements Rationale

This rationale explains the reason for the chosen assurance package EAL3 for this Profile. The major considerations are that the browser requires a moderate security level and that EAL3 is deemed achievable. As discussed in Section 4.5.1, EAL3 set for an access control functionality would be undermined if the identification and authentication functionality provided by the OS is EAL2. Thus, the underlying operating system must also have a EAL level at least that of the browser.

In summary, the underlying operating system shall need to have the EAL3 as the browser relies on the operating system for numerous important functions such as user authentication, access control, etc.

4.7 Overview of the Web Browser Protection Profile

The work on the security requirement profile (SRP) was started in the middle of 2000 and completed the SRP in March 2001. "A Web Browser Protection Profile (WBPP)" developed by U.S. government and industry become publicly available in early May 2001 [24]. Therefore the SRP's authors were not aware that there was a profile being developed for browsers at the time the research was started. However, the U.S government's work on a PP for web browsers demonstrates that browser security is an important issue. Thus, here a brief discussion of the WBPP is given, and similarities and differences between the WBPP and the SRP are identified.

As the WBPP is developed for the U.S. government, the description of the browsers given in the PP emphasises that a browser must be a "A PKI-enabled secure web browser compliant with the Global Information Grid (GIG) policy IA6-6510 and the Information Assurance Technical Framework (IATF) Forum documentation while communicating through the Hyper-Text Transfer Protocol - Secure (HTTPS) ... The browser may support the execution of mobile code (e.g., JavaScript or ActiveX) in an isolated secure manner".

In this section, some key components of the WBPP are briefly discussed. They are: security environment for browsers, security objectives for the TOE and its environment, security functional requirements, and security assurance requirements. The comparison of the WBPP and the SRP is then discussed in detail.

4.7.1 Security Environment for Browsers

The security usage assumptions in the WBPP describe security aspects of the environment which includes personnel, physical, and connectivity aspects. The assumptions made in the PP are: administrators follow guidance for the configuration and maintenance of the TOE; users utilise information for intended purposes; network connections are secured; data is non-malicious; plug-ins are not part of the TOE; certificates

determine the role permissions; and time synchronisation with the OS.

For threats, the WBPP is concerned with human users, downloaded data, and browser functions. The threats coming from human users include errors made by administrators, user data viewed by other users, browser information obtained by unauthorised users, and unauthorised functions performed by users. Threats in downloaded data include malicious code and active content operations, and downloaded data accessing system files. The browser can also unintentionally or intentionally misrepresent security relevant information.

The PP then imposes seven policies which aim to protect browsers' sensitive data. The policies address user accountability, information accessibility and availability, guidance for installation and usage, data integrity, user training, and physical access control.

4.7.2 Security Objectives

The PP defines a list of security objectives for the TOE in order to counter identified threats and/or comply with any organisational security policies identified. The security objectives are identified in the following sections.

4.7.2.1 Security Objectives for the TOE

The WBPP includes a number of objectives that address security issues in relation to browser users and their data, the downloaded data, secure sessions, and the connecting servers. Objectives regarding browser users and their data include restricting users' actions prior to being authenticated, maintaining security-relevant roles and the association of users with those roles, maintaining the user attributes for each active session, providing evidence by users for identification and authentication processes to support accountability by the destination servers, and separating user data (e.g., history, profiles, cookies, and cache) from other user data. Objectives which aim to address the downloaded data include controlling active content downloaded from a web server, and displaying and labelling the information according to its content as received from the destination server. Objectives identified for secure sessions are recovering browser automatically to a consistent and secure state, and terminating an inactive secure session. For servers, one objective is defined and this is validating the connected destination

server.

4.7.2.2 IT Security Objectives for the TOE Environment

In this section, the WBPP identifies three objectives: defining the necessary settings for each cryptographic operation to establish and maintain a secure state of the TOE; using appropriate cryptographic services in order to authenticate the browser user to the server; and operating with other network devices to protect transmitted data.

4.7.2.3 Non-IT Security Objectives for the TOE Environment

Non-IT security objectives for the TOE Environment in the WBPP are: training authorised administrators for the establishment and maintenance of sound security policies and practices; procedures for browser delivery, installation, and managing in order to maintain the system security; and the protection of computing resources and conflict resolution in a multitasking environment.

4.7.3 Security Functional Requirements

Based on the security objectives, the PP selected four classes of security functional requirements: Cryptographic Support, User Data Protection, Identification and Authentication, and Security Management.

4.7.4 Security Assurance Requirements

The profile has chosen EAL2 for security assurance requirements and EAL2 is defined by the CC as "structurally tested".

4.7.5 Comparison of the WBPP and the SRP

The TOE description in both profiles is very similar. However, the rest of each profile has a number of differences to the other and they are discussed in this section.

For the security usage assumptions in the security environment section, the SRP claims that browsers rely on the OS in terms of access control for browsers and their data, user identification, and user authentication. However, the WBPP did not contain an assumption with regard to access control, identification, and authentication to be provided by the OS. The SRP addresses some assumptions made by the WBPP in other

sections of the SRP, for example the assumption "the TOE will rely on the operating system for the purpose of a timekeeping mechanism" is addressed by security audit class in security functional requirements.

Both profiles identify threats very closely except for "the alteration of file type" addressed by the SRP and "misrepresenting security relevant information" raised by the WBPP. The SRP's concern of file type alteration is covered by an assumption in WBPP. The assumption in WBPP presumes the protocols used for network connections protect information (by physical protection and encryption (ssl)) from disclosure and modification during transmission. An example for misrepresenting security relevant information, as described by the WBPP, is that a browser displays a padlock as if SSL is enabled but no actual SSL takes place, and this tricks users into sending sensitive data in clear form. As discussed in Section 2.3.1, this is caused by administrators and browser implementers who lack good implementation practices and knowledge of cryptography. Thus SSL is an adequate secure connection protocol if it is configured properly and is used in conjunction with a secure operating system. For this reason, the SRP's authors do not identify this threat in the SRP.

For Policy, both profiles include a policy for user accountability while the WBPP identified six more policies aiming to protect browsers' sensitive data. Unlike the WBPP, the SRP does not identify policies for information availability, information accessibility, training users to use the browser properly, establishing guidance for installation and use of the browser, retaining integrity of the browser information, and controlling physical access to the host system. The reason for not having these policies is that these issues have been addressed elsewhere within the SRP. Unlike the WBPP, the SRP does not identify policies for information availability, information accessibility, training users to use the browser properly, establishing guidance for installation and use of the browser, retaining integrity of the browser information, and controlling physical access to the host system. The reason for not having these policies is that these issues have been fully or partially addressed elsewhere within the SRP. The policies for information availability, accessibility, retaining integrity of the browser information are addressed by an assumption for access to a browser and its data, which assumes that all access to a browser and its data is controlled by the OS. The policy for guidelines is covered by a threat, which notes that administrators may commit errors in

installation, configuration, or management of the TOE, compromising security. While the policies for physical controls and training users are not explicitly addressed in the SRP, they are partially covered by security requirements for environment, operating and training, respectively.

Security objectives for the TOE in both profiles are very similar but each has extra objectives that are not in the other. The SRP defines an extra objective for recording audit data. The WBPP includes additional objectives for limiting access of active content, recovering the browser to secure state, and terminating an inactive session. The objective for limiting access of active content is not addressed in the SRP, but the SRP contains objective for restricting file type which will not allow downloading malicious data. Despite the different definitions of these security objectives listed in the SRP and the WBPP, the same SFRs are chosen to meet these objectives. The SRP does not define objectives for recovering the browser to secure state and terminating an inactive session, because even though these objectives aim for a good security practice, they do little to improve security of the browser and its data as these functions are already supported by the underlying OS.

Security objectives for environment in both profiles have focussed on different issues. The SRP requires the OS to have an appropriate security level (since it support user identification and authentication, and access control mechanism), but it is not addressed by the WBPP even though it is crucial for browser security. The SRP also requires the browser to be designed, installed, maintained, and operated in a secure manner, and that are listed in non-IT security objective for environment in the WBPP. The SRP defines the browser to have a mechanism for user accountability while WBPP addresses this as objective for TOE. Training for the browser administrators is addressed in the environment section by the SRP, but the WBPP prefers to place it in the policy section. Thus it appears that the WBPP considers this requirement is critical and must be enforced at organisational level (SRP simple expects this objective from the TOE environment). The SRP requires the browser to interact with trustworthy servers, but the WBPP does not have any objective for this issue. The WBPP identified objectives for environment in relation to the cryptographic mechanism and network services, which are covered by the SRP under the objective for designing, installation, and maintaining of the browser.

For non-IT security objectives, the SRP defined none and the WBPP has four objectives which are however addressed in the SRP under security objectives for environment.

In security functional requirements section, both profiles choose user data protection, identification and authentication classes, protection of the TOE security functions, and trusted path/channels. The additional class in the SRP is security audit. Because the WBPP is completed without identifying secure audit, it seems that the underlying OS is providing it. The WBPP also has some additional classes such as cryptographic operation class, which is considered by the SRP as part of the trusted path/channels (the TLS function), and security management class which is included in security requirements for the IT environment section in the SRP.

The Evaluation Assurance Level (EAL) selected for the SRP and the WBPP are different, as the SRP has chosen EAL3 and the WBPP has selected EAL2. The CC part 3 explained that "EAL3 is applicable in those circumstances where developers or users require a moderate level of independently assured security, and require a thorough investigation of the TOE and its development without substantial re-engineering. EAL2 permits a conscientious developer to gain maximum assurance from positive security engineering at the design stage without substantial alteration of existing sound development practices". The SRP's authors believe that EAL3 is achievable as it requires a moderate level of assured security, and that a thorough investigation of the browser and its development can be obtained by people with general computer security knowledge.

As a summary, both profiles have been developed with similar security concerns. The major differences between them are the SRP raises the auditing security issue while the WBPP discusses about the security issue on active content which is covered by file type restriction in the SRP, and the inactive secure session which, the SRP's authors assume, has little impact on the browser security.

4.8 Summary

This chapter has described the considerations undertaken over the period of development of the SRP. The SRP has entirely focused on security in file downloading in terms of confidentiality and integrity. The Profile has five objectives addressing au-

dit trial, access restriction, authentication, confidentiality, and integrity. The rationale section demonstrates that the chosen security components from the CC comply with the identified objectives. It is worthwhile to reiterate that the underlying operating system shall have an equivalent or higher security level (Security Assurance Requirements level in the CC's terminology) with the right SFRs to support browsers, since the browsers make use of its identification and authentication mechanism, access control to the browser itself, and its data in storage or transit between two applications (e.g., browser and email application). The complete SRP has been included as Appendix A. A comparison of the SRP with the more recently released "Web Browser Protection Profile" [24] was performed.

In the next chapter, this SRP is used to assess current browsers and to determine if they have the security requirements identified in here.

Chapter 5

Assessment of Current Browsers

In Chapter 2, browser security in terms of the services they provide was highlighted. The SRP for browsers was then composed in Chapter 4 to address security problems identified in Chapter 2. This chapter contains a discussion on how much security today's browsers are providing by checking against the security requirements identified in the SRP for browsers.

Chapter 2 addressed browser security problems in three components: connection, downloaded data, and host security. These three aspects affect each other if security in one area fails. For example, if connection security is compromised by an attack, then data can be modified during transmission. Thus downloaded data security fails and consequently host security is affected by the compromised downloaded data, for example if there is a virus in the file. Therefore it is important that security is deployed in all components of browsers. Since browsers initialise a new connection to servers and send the servers requests for files to be transferred, the research examines the file transferring protocols used by browsers. The analysis of HTTP over TLS, the principal file transferring protocol used on the web, has been included in this chapter while the findings for some file transferring protocols such as File Transfer Protocol (FTP), Secure Copy (SCP) over Secure Shell protocol (SSH), and File Transfer, Access, and Management protocol (FTAM) are included in Appendix B.

The SRP outlined the security objectives for browsers such as authentication, confidentiality, integrity, access restriction, and audit, as developed in Chapter 4. The SRP consequently identified security requirements in order to achieve the security objec-

tives outlined. These security requirements are:

- *Identification and authentication:* users are identified before any other action is performed by the browser.
- *Trusted path/channels:* browsers use a communication path between themselves and a server that is logically separate from other communication paths to ensures the identification of connecting servers, and to protect the transmitted data from alteration or disclosure.
- *User data protection:* restricting all users to the rules outlined in the organisational policy with regards to user identity, the file type and whether a secure connection for communicating the file is required.
- *Security audit:* to record security relevant events to hold individual users accountable for the actions they have performed.

The SRP is then used in evaluating modern browsers in order to see if they have sufficient security measures suitable for downloading sensitive data. For evaluation purposes, the research have chosen Netscape and Internet Explorer as they are the most popular commercial browsers, Opera for its reputation for speed and less resource usage, and the Lynx text browser for its recognised simplicity. Since browsers make use of user identity and access control provided by the OSs, the research also selected two current major operating systems, Windows 2000 and RedHat Linux version 7.3, for evaluation with respect to these two features.

This chapter is organised as follows. Section 5.1 discusses the requirements of the OS in supporting security in browsers. Since browsers operate on top of the underlying OS, this section ensures that the evaluated OSs have adequate security measures to support browsers. The following section, Section 5.2, analyses security features in connection protocols used by browsers, e.g., HTTP and HTTP/TLS. Section 5.3 then explains how auditing is implemented in current browsers and what they still have to improve for obtaining audit data as required by the SRP. In Section 5.4, the issue of restricting file access is discussed in terms of users downloading files with administrators permitted file type and users accessing downloaded files with correct permissions. Finally, the conclusion in Section 5.5 summarises the evaluation.

5.1 Operating System Responsibilities

As mentioned in Chapter 4 (Section 4.2), the underlying OS is responsible for user identification, authentication, and access control. Section 4.2 continued to explain the reasons for relying on the OS for these requirements. In short, browsers make use of the OS's identification, authentication, and access control mechanisms for two reasons. Firstly, they are major tasks of the OSs, and secondly if the OSs fail to stop an intruder, there is no way for a browser to protect itself at all. In fact, as explained in Section 4.5 that the operating system must have the same security level or higher as the browsers, since the overall security level of a browser is the minimum security level of any cooperating application which includes the operating system.

The OS can provide browsers with user ID, time, and secure storage. Modern operating systems allow programmers to use system API calls such as "getuid", "time", and "open" to create a file. Even though the research does not recommend identifying and authenticating users again after they log in, a secure browser needs to utilise these system calls for creating audit data. In Chapter 6, how a prototype for a secure browser uses the system calls to generate audit data is discussed.

The research identifies the extent of user identification, authentication, and access control mechanisms (part of user data protection) used by the current major OSs. The findings are outlined in the following sections.

5.1.1 User Identification and Authentication

The research analysed how user identification and authentication is performed in the selected OSs, namely Windows 2000 for the Windows platform and RedHat Linux 7.3 for the UNIX platform. In general, both OSs prompt users to supply account information (user name and password) in the first stage of logon, then the OSs verify the account information and provide access to the computer if the information is correct. Windows 2000 provides local authentication by using local account information, and so does Linux 7.3 by using passwords stored in the /etc/passwd file. They also provide central authentication mechanisms such as Kerberos[1] in Windows 2000, and

[1] An Internet standard security protocol for handling authentication of user or system identity. With Kerberos V5, passwords that are sent across network lines are encrypted. Kerberos V5 includes other security features, e.g., Key Distribution Centre (KDC) service that issues Ticket-granting ticket (TGT)

NIS+[2] and Sesame[3] in RedHat Linux 7.3. In short, both Windows 2000 and RedHat Linux provide a number of industry-standard authentication mechanisms, including user name and password, smart cards, single sign-on secure authentication (Kerberos, Sesame), and biometric authentication mechanisms.

Regardless of the method used to verify user identity, Windows 2000 consistently uses Active Directory to look up the identity presented by the authentication mechanism. If authentication is successful and an account is found for the user, Windows 2000 provides the user with a set of credentials that can be used throughout the network to access resources. On the other hand, RedHat Linux uses Pluggable Authentication Modules (PAM), which allows the system administrator to authenticate users by using PAM-aware applications without having to recompile authentication programs.

In summary, the two major OSs, Windows and Unix offer many user identification and authentication mechanisms to choose from. For example users can deploy a simple password authentication or more sophisticated mechanisms such as biometric authentication.

5.1.2 Access Control Mechanism

Generally, OS access control is performed by granting access attributes to owner, group, or everyone. The granted access attributes are stored in the Access Control Lists (ACLs) which are read by the OS to determine whether to allow or deny access to a file or a directory. ACLs are also used when a process (started as soon as a user runs a program) tries to access some system resources. The decision on whether to allow or deny access to the system is made based on the privilege of the user who starts the program, for example a program run by a normal user cannot access a password file but the program run by administrators can.

In Windows 2000 and RedHat Linux 7.3, if the same attributes are required by many users, administrators can put the users into a group and then grant it with access attributes. With both OSs, a user can be a member of multiple groups.

---

in order to obtain Ticket-granting service (TGS) tickets that allow users to authenticate to services in a domain [31].

[2]NIS+ is a database containing users account information and related authentication data [41].

[3]SESAME technology offers sophisticated single sign-on with added distributed access control features and cryptographic protection of interchanged data [52].

Windows 2000 offers five levels of permission - Full control, Modify, Read, Read and Execute, and Write - to be assigned on files and directories. Windows 2000 also allows administrators to assign different permissions to different users and groups for a particular file. For example, administrators can allow one user to read the contents of a file, allow another user to make changes to the file, and prevent all other users from accessing the file. Hence, Windows 2000 provides a simple, efficient way to set up and maintain security on browsers and their related data by assigning only the necessary access attributes to the browsers and the related file for an individual user or a group.

With RedHat Linux 7.3, the access control mechanism is simple because it defines only three access attributes - read, write, execute - for three types of account such as the owner of a file or a directory, groups, and everyone. Thus, by default RedHat Linux 7.3 supports ACLs with less granularity than Windows 2000. However, there exist ACLs utilities called "setfacl" for RedHat Linux 7.3 with functionality similar to that of Windows 2000 [23].

Even though RedHat Linux 7.3 has only three access attributes (read, write, execute), it has similar functions to the other access attributes of Windows 2000. For example, "Delete" in Windows 2000 is part of "Write" attributes in RedHat Linux, and the same functions of "Change permission" and "Take ownership" in Windows 2000 are done in RedHat Linux by using "chmod" and "chown" commands respectively. For directories, RedHat Linux's "Write" attribute carries the same effect as "Add" in Windows 2000 and "ls" command in RedHat Linux produces the same result as in "List" attribute in Windows 2000.

Table 5.1 shows access functions and the names of access attributes in RedHat Linux 7.3 and Windows 2000.

The major concern with access control mechanisms comes from the fact that the downloaded data, audit files, and the browsers themselves are protected by these access attributes (permissions) from unauthorised access. If an audit file, for example, is inadvertently set writable to everyone, then the content of the file could be edited by any user and consequently administrators cannot trace how a security problem (e.g., changes in the browser's configuration file) has occurred.

As a conclusion, Windows 2000 can provide finer-grained access control lists but RedHat Linux 7.3 requires an additional ACLs utility in order to support similar fine-

grained ACLs.

Access Functions	RedHat Linux	Windows 2000
Read access	Read (r)	Read
Write access	Write (w)	Write
Executable	Execute (x)	Execute
Delete	Write (w)	Delete
Change permission	Owner only	Change permission
Take ownership	-	Take ownership
Full control	Read Write Execute (rwx)	Full control
Add (for directory)	Write (w)	Add
List (for directory)	Execute (x)	list

Table 5.1: Access type comparisons in UNIX and Windows operating system

5.2 Connection Security

Browsers can use either HTTP by itself or HTTP over TLS to setup a connection with servers. However, there are many weak points in HTTP alone. For example, HTTP supports user authentication but cannot authenticate servers since they can only check the IP address and this can be modified by an attack on servers or during transmission. HTTP also transfers data in clear text, hence it can be monitored by attackers without being detected.

In the light of weaknesses in HTTP, the SRP enforces the use of HTTP over TLS to gain authentication between browsers and servers, and to give downloaded data confidentiality and integrity. The research found that all the evaluated browsers support TLS.

Since TLS uses certificates for peer authentication and to provide a mechanism to achieve data confidentiality, the research investigates browsers further for how and what kinds of certificates are provided. Generally, they all have a built-in database,

which contains many certificates of Certificate Authorities (CA) used by browser to validate server certificates which are signed by a CA. The browser therefore cannot validate certificates signed by a CA whose certificate is not in the database. The standard certificates include all necessary fields such as version and serial numbers, issuer, valid date, and fingerprint.

As the SRP requires using HTTP over TLS, the functionality of the protocol pair and their appearance in the selected browsers are analysed. As a result, the following sections highlight the security features of HTTP over TLS, the overview of TLS in browsers, and the vulnerabilities in TLS.

5.2.1 General Functionality of HTTP Over TLS

The goal of TLS is to provide authentication, data confidentiality, and data integrity. Conventional HTTP server/client applications run on TCP port number 80. If TLS is requested, then the connection is set to port 443 of the server by default [17]. TLS uses public key cryptography to enable mutual authentication between the server and the client, and applies symmetric key cryptography for the bulk data encryption. As HTTP is just a communication protocol, it would not impose any restriction on the file's content type for transfer. Setting security attributes of the file downloaded is also out of scope for HTTP.

How HTTP over TLS initialises a connection between a server and a client is summarised as follows [51].

1. The HTTP client initiates a connection to the server on the default port (443).

2. The client begins the TLS handshake by sending the TLS client "Hello" message to the server. Server identification can be verified at this stage by using the hostname or IP address information presented in the server's certificate message. Checking the client identity is usually omitted by the server, though it is possible to achieve if the client has a certificate.

3. Upon successful processing of the TLS handshake, the client sends the first HTTP request (for file downloading here) that must be sent as the TLS "application data". The protocol identifier for HTTP/TLS is "https" and this usually appears in the URL, for example https://www.qut.edu.au.

4. To close the connection, a valid closure alert must be exchanged before the actual connection is closed. In this case, the session can be reused later. If the connection is closed without sending a valid closure alert, the session must not be reused. The reason for this is the TLS does not understand the HTTP mechanism and cannot therefore determine whether a message has been truncated.

Figure 5.1 illustrates the communication sequences of the TLS protocol between a browser and a server.

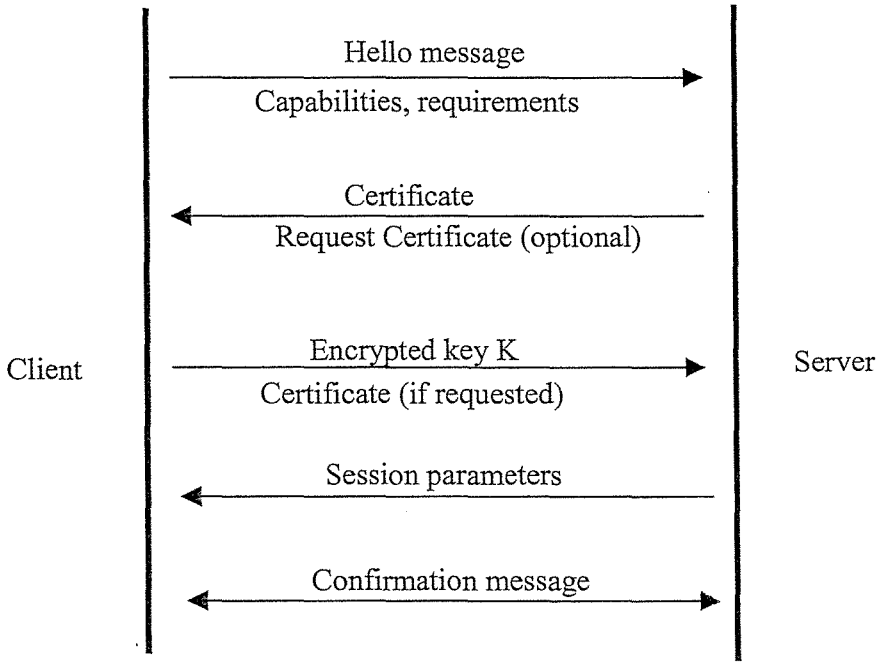

Figure 5.1: Login process of TLS

5.2.2 TLS Overview in Browsers

TLS version 1.0 is included in all the evaluated browsers for use. The TLS protocol, the successor of SSL, was developed by the Internet Engineering Task Force (IETF), and is almost identical to SSL 3.0 except in its new approach to implementation i.e. an open and standards-based solution, using non-proprietary ciphers, improved error

reporting, and use of HMAC (a hash function based message authentication code) instead of MD5 (Message-Digest Algorithm) [8].

TLS checks server's certificates in the first stage of communication with the server. The server's certificate is signed by a CA which is also shown in the certificate. As browsers store a list of root CA certificates which they consider trustworthy, the server's certificate can be verified. If a CA in a server's certificate is not included in the list, then browsers display a warning message and users have to decide whether or not to accept the certificates and continue to setup the connection. Upon users' acceptance of the server certificate, browsers store the CA certificate in the list for future use.

5.2.3 Vulnerabilities in Implementations of TLS

As mentioned above, TLS is the successor of SSL. TLS also inherits the SSL vulnerabilities. Since SSL vulnerabilities are broadly discussed in Section 2.3.1, the following is a brief discussion of the TLS vulnerability. The errors in the TLS implementations introduced vulnerability in private keys which are used for mutual authentication, and in session keys which are used for bulk data encryption. Some incidents in the past showed that the private keys in use can be recovered from memory, and session keys could also be recovered from the server messages responding to a number of structured request messages sent by a client (browsers). Like TLS programmers, the OS programmers also contribute vulnerabilities to TLS by leaving some security holes in the OS, by which attackers can insert malicious certificates into the trusted root CA list. This will result in users communicating with malicious servers. In addition, browsers themselves allow some protocols to display a visual TLS symbol while no actual TLS connection is in use, and this encourages browser users to send sensitive information such as passwords over insecure connections.

As discussed in Section 2.3.1, these problems mentioned above are now rectified, and TLS is, in fact, an adequate secure connection protocol if it is configured properly and is used in conjunction with a secure operating system.

5.3 Audit Trail

Auditing records users' activities with the associated identity and the time of the action. Recording security relevant events helps administrators to analyse how the security problems occurred or have been initiated. By having such records, administrators can eliminate security holes in browsers and identify the responsible users. However, the evaluated modern browsers do not generate audit data and it is left to the underlying OS, which vary in their scope for recording audit data.

Even though the evaluated browsers do not generate audit records, a history file (created for each user as part of a user profile) in IE, Netscape, Opera, and Lynx contain accessed URLs with time stamps. IE and Netscape also record how many times a web page is visited but only Netscape can tell the first visit date and the last visit date for a particular web site. The research is also interested in the users' requested files and the downloaded files from servers, but have found that the history files have no information on them (though Opera records a list of downloaded files in the "download.dat" file).

With RedHat Linux 7.3, browsers can record user name, time, any request, and error messages by using the logging utility. The majority of logging in Linux is provided by two main programs, system log (syslog) and kernel log. The system log utility stores log files generated by various programs, especially the login program (logs all logins and logouts into the system) and the kernel log utility records system resources usages (logs all messages produced by kernel and system program) [53]. However, both Window 2000 and RedHat Linux 7.3 have the audit configuration for every file and it must allow the OS to record every system call. The logging utility can be used to generate a specific log file concerning users' activities with browsers for auditing purpose. The research found that even though the syslog service is used at the discretion of the application, there is no information of user ID, access time, user request for web pages, and successful or fail result of the request in log files.

On the other hand, Windows 2000 offers three types of event log, namely System log, Security log, and Application log. The system log generated by the operating system contains events of system components, for example the failure of a driver or other system component to load during startup. The application log generated by applications contains records of errors with the application developer deciding which events

to record. The security log specified by administrators can record security events such as valid and invalid logon attempts, resource use, and creating, opening, or deleting files. The Event Log service starts automatically when Windows 2000 starts and the log file can be viewed via *Event Viewer* which displays error messages, warning messages, information on operations of an application, and audited security access. By default, application and system logs can be viewed by all users, but security logs are accessible only to administrators. Administrators can restrict access to the log files by using the access control mechanisms described in Section 5.1.2. Security logging that is turned off by default can also be enabled by administrators. However, Windows 2000 does not record users' interactions with applications (they only record changing attributes of files or directories) except if the application implements its own logging. The browsers investigated do not use the Windows 2000 logging facility.

Therefore, the research concludes that the evaluated browsers do not generate the security audit records required by the SRP for browsers.

5.4 Access Restriction

As described in Section 4.3.1, access restriction is one of the desirable objectives because it provides administrators with control over a particular user downloading a specific file type through a secure connection. In order to prevent virus attacks, for example, users should not be able to download executable file types such as files with extension "exe", "ps", etc. Thus, the access restriction mechanism needs information of user identity, restricted file type, and verification on whether a secure connection is required, so it can use these information to determine which files are acceptable to download by whom. The only problem with this is that today browsers have no feature for administrators to restrict file type for downloading, even though restricting file type can increase browser security.

The only available feature in relation to access restriction is a file download option from Internet Explorer which allows users to choose whether or not to have the browser download files. The file downloading option does not apply to FTP files and files displayed by browsers or helper applications. Thus, users can still read and save web pages as long as the browser displays them (with or without a helper applica-

tion). Netscape, Opera, and Lynx offer neither file download options nor features for restricting file type.

In general, browsers save the downloaded file to a temporary location and then pass it to the appropriate program (a helper application) for display. If the helper application is not available then users are prompted to save the file onto storage. So it is impossible to stop users downloading files but it is unlikely that the users are able to view the downloaded file without a helper application. All the evaluated browsers have an option for users for setting file types and the associated applications. However, if the operating system is configured by an administrator so that users cannot install helper applications, then some control over the types of files a user can view is obtained.

In summary, access restriction in the evaluated browsers does not meet "Access restriction" objective of the SRP for browsers.

5.5 Summary

This chapter highlights security features in major browsers and the TLS secure connection protocol. It is noted that none of the selected browsers allow client hosts to log the activity of the protocol itself or to restrict file type for downloading. In all browsers, user identification, access control, and time functions are supported by the underlying operating system. The findings of this analysis are applied in a prototype model for a secure browser outlined in the next chapter.

Chapter 6

Case Study

In Chapter 2, the research investigated the security measures provided by the current browsers, and the findings were used to construct a SRP for browsers (in Chapter 4). Using the SRP, the research was then extended in Chapter 5 to analyse modern browsers and a file transferring protocol (HTTP/TLS) in order to determine what security requirements they support. As none of the browsers analysed offers all security requirements required by the SRP, the research has implemented a prototype model for a secure browser. The Lynx text-only browser was modified to demonstrate that the model is achievable.

This case study involves outlining a prototype model and modification of an existing browser in order to meet the defined prototype model. The prototype model is a framework for a secure browser program that can be used to download data (files) securely from a remote server. The model consists of five security requirements: user identification and access control, cryptographic mechanism, limitation on file type, file attribute, and audit trail. For the source code modification, the Lynx browser is chosen due to its relatively simple features and it is also available as an open source code.

The chapter is organised as follows. In Section 6.1, the desired design requirements are highlighted. Sections 6.2 - 6.6 explain how each component of the prototype design will function and interact with other components and/or the underlying operating system. Section 6.7 contains the rationale of the prototype design. The chapter finishes with a summary in Section 6.8.

6.1 Design Requirements

The prototype model is carefully designed to achieve security objectives identified in the SRP for browsers discussed in Chapter 4 (see also Appendix A for details). The security objectives that are identified in the SRP and which the research has set out to achieve are:

- *Auditing:* to record security relevant events to hold individual users accountable for the actions they have performed.
- *Restriction:* to limit access to a browser and to download the permitted file types only. Permission on file type is imposed only by the administrator.
- *Authentication:* to provide a mutual authentication mechanism to verify the identity of servers by users and vice versa.
- *Confidentiality:* to ensure that data cannot be illegitimately eavesdropped or obtained during transmission.
- *Integrity:* to ensure that the integrity of the downloaded files is maintained during transmission.

With the aim to overcome limitations encountered in the browser analysis in the Chapter 5, the prototype design as shown in Figure 6.1 is built using five core mechanism components: user identification, cryptographic mechanism, file type limitation, file attribute, and audit trail. The components and their functionality details are given in the following sections.

6.2 User Identification and Access Control

The user identification component is necessary because it verifies a user who attempts to use the program. The component communicates with the underlying operating system and validates the user according to the credentials provided by the operating system, which has already successfully identified and authenticated the user during the login procedure prior to allowing the use of the computer.

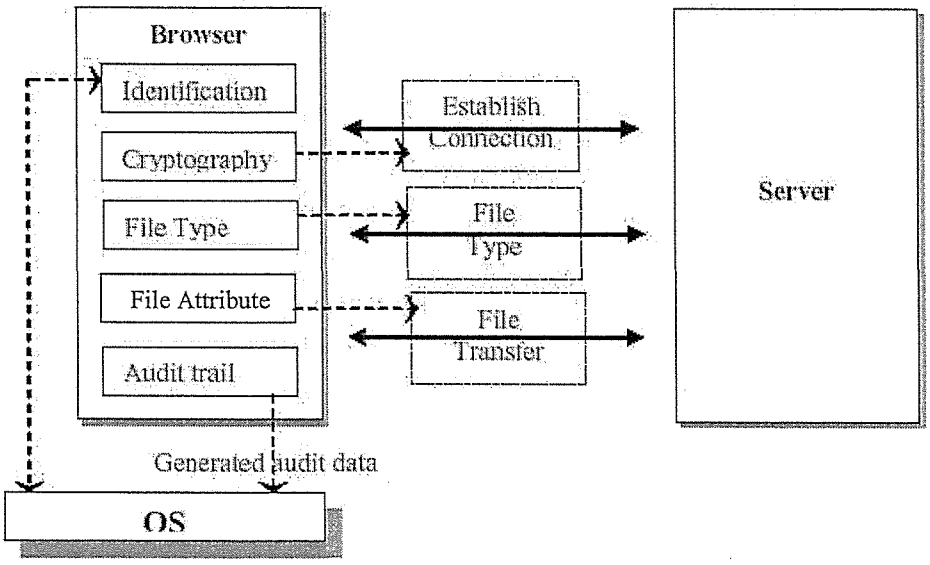

Figure 6.1: A prototype design architecture of a secure browser

Security for this mechanism relies totally on the underlying OS and the reason for trusting the OS is that modern OSs conventionally identify and authenticate users as part of their own access control mechanism, thereby the model shouldn't reiterate this authentication process. These available secure authentication mechanisms are implemented by particular interest groups focusing only on a specific area, and the developed authentication mechanisms allow organisations to choose an appropriate one from the many variants (e.g., from simple password authentication to smart card with public key cryptography). It is assumed that more and more secure authentication mechanisms, particularly in network environments, will emerge since there is much research being undertaken currently in this area.

By default, the Lynx browser can be started by all users of the host computer. Lynx, like other browser applications, relies on the operating system for user identification and authentication. This research does not investigate the necessity of applying user identification and authentication functions when a user starts the browser, because there is not much chance to stop the user who has successfully logged in (by any means).

Overall, the research is satisfied with the functionality of the current operating systems in terms of identification and access control mechanisms.

6.3 Cryptographic Mechanism

This component is used to perform crucial functions such as a mutual authentication between a user and a remote server to ensure that they are both legitimate. It also encrypts data before transmission to ensure that data integrity and data confidentiality are obtained. Functionality of the components is the same as for the HTTP/TLS protocol.

A public key cryptographic mechanism is employed for mutual authentication. For that purpose, both the remote server and the user must have public key certificates to provide their credentials. The certificates can be issued by either a trusted third party or an internal key issue centre of the organisation.

The summary of the mutual authentication and key exchange mechanism are as follows (from TLS):

- The client sends a request for server ID (a public key certificate)
- The server returns its ID and a request for the client's ID
- The client returns its ID and a proposal of a session key for data encryption and decryption
- The server sends its agreement on the session key

Upon receiving a request for file to transfer, the server encrypts the requested file using a symmetric key (agreed to and generated by both parties during the key agreement process), then sends it to the user. This simple single step supports confidentiality of the file so it can only be decrypted using an appropriate key. To achieve data integrity, the server generates a value (e.g., MAC) that is associated with both the contents of the file transferred and the identity of the server. Browsers can compute a value using the same method the server used and compares it with the one the server has provided. If the two values are identical then the file's integrity has been maintained.

For Lynx, there is no need to implement a mechanism for data confidentiality and integrity since TLS can be compiled into Lynx as additional component.

The Lynx browser makes use of cryptographic mechanisms in establishing a secure connection with remote servers. In fact, Lynx provides TLS which extends the security of transmitted data by encrypting data via a public key cryptographic mechanism.

The model only uses certificates for browsers and servers issued by the same internal or trusted third party CAs. The model itself would neither generate certificates for browser nor accept any server certificate from intrusted CAs.This requirement would reduce the possibility of users falling into man-in-the-middle attacks[1], because the certificates used by the client's browser and the connected server are issued by the same CA, thus less likely for the attackers to obtain the valid certificate.

As highlighted in Section 2.3.1, TLS has software flaws and a problem with private key storage. However, implementing a new mechanism to replace TLS or enhancing TLS is out of scope for this research, and the research is satisfied with the method of cooperation between Lynx and TLS. Hence TLS is used in the model for a secure connection.

6.4 Limitation on File Type

This component allows administrators to put constraints on permitted file types for downloading. The administrators are able to compile a list of allowed file types for downloading by selecting them from the built-in predefined list. When users make a request to download files, the model verifies the requested files according to the list and makes a decision on whether to allow or deny.

In addition to determine file type for filtering mechanisms, the model will use the MIME mechanism since it has been used as a standard for file type verification in many data transferring applications (e.g., email) [21]. However, the research is aware that servers check file extensions to determine the MIME type and a file extension can be modified by anyone with an appropriate access right to the file. This problem is overcome by using TLS, the secure connection.

The research has found that there is a mechanism for determining the file type implemented in UNIX environments, known as the "file" command. The mechanism consists of three sets of tests performed in order:

1. Filesystem tests: use system call ("stat" - distinguishes between directories, symbolic links, and normal files) and examine the result.

[1] an attacker, who appears to be a connecting server to a client (browser) and acts as a client to the connecting server, relays data packets between a browser and a server.

2. Magic number tests: check for data in particular fixed formats (numbers stored in a particular place near the beginning of the file).

3. Language tests: at this stage it is assumed that the file is a text file. The test will then look for particular strings to determine the written language. For example, the keyword "struct" indicates a C program.

Future research may investigate the "file" mechanism and find a possible way to implement it in heterogeneous operating system environments, as the mechanism appears to be useful in ascertaining malicious data such as virus files. It is however out of the scope of the current research and this mechanism is currently available only in UNIX environments.

In summary, the research has decided to use MIME for determining file type in this model.

6.4.1 Filtering File Types

Lynx unconditionally allows users to access files in any format (file extensions). However, security for browsers is concerned with file types in relation to their capabilities to access system resources and compromise system security. Therefore the Lynx source code is examined to see if it can be altered to obtain the file type restriction option. The examination shows that browser's requests can be placed in two categories.

1. A user's request for a home page is actually asking for a file which can be any type depending on the server configuration. Browsers won't know the requested file type until they receive a HTTP header from servers indicating contents type (using MIME format).

2. Most requests for file downloading usually end with a file extension which is required for an appropriate file handling process.

The experimentation shows that Lynx can be modified to restrict file type by using two different methods: granting access if users' requested file extension is valid and granting access if users' requested file is a valid MIME file type.

Note that the valid file extension and MIME types for file downloading are set by the administrators as was explained in Section 6.4. The first mechanism works well

if the requested URL shows a file extension while the second mechanism supports file type restriction in more standard means. The implemented two file type filtering mechanisms are explained in detail in the following sections.

6.4.1.1 Granting access to valid file extension

As an experiment, additional code is added into the Lynx source code in order to filter users' requests by checking the file extensions against a list compiled by administrators. The sequence processes involved in checking the file extensions as shown in Figure 6.2 are:

1. User requests URL.
2. The browser checks the file extension against the list.
3. If the requested file is a permitted file type, then the request is sent to the server.
4. If the requested file is not a permitted file type, then the request is discarded and an error message is displayed.

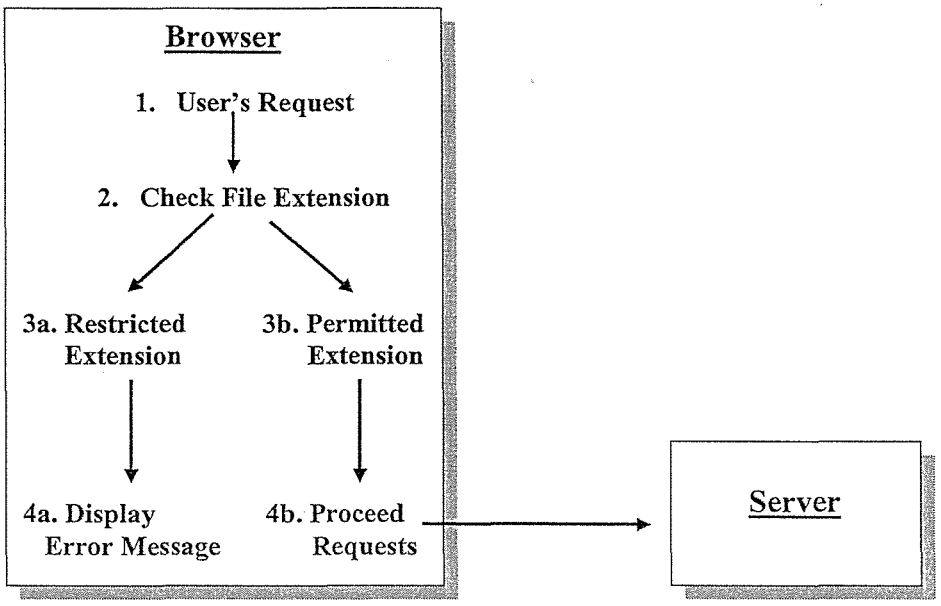

Figure 6.2: File extension filtering processes.

The result of the experiment has shown that the filtering mechanism is effective and efficient for the requests containing file extension. However, some requests do not show extensions and thus give no indication of the file type which can be varied depending on the server-side script. The research therefore implemented the second mechanism which made use of MIME file type in order to filter users' requests. The following section discusses the file type filtering mechanism using MIME specification.

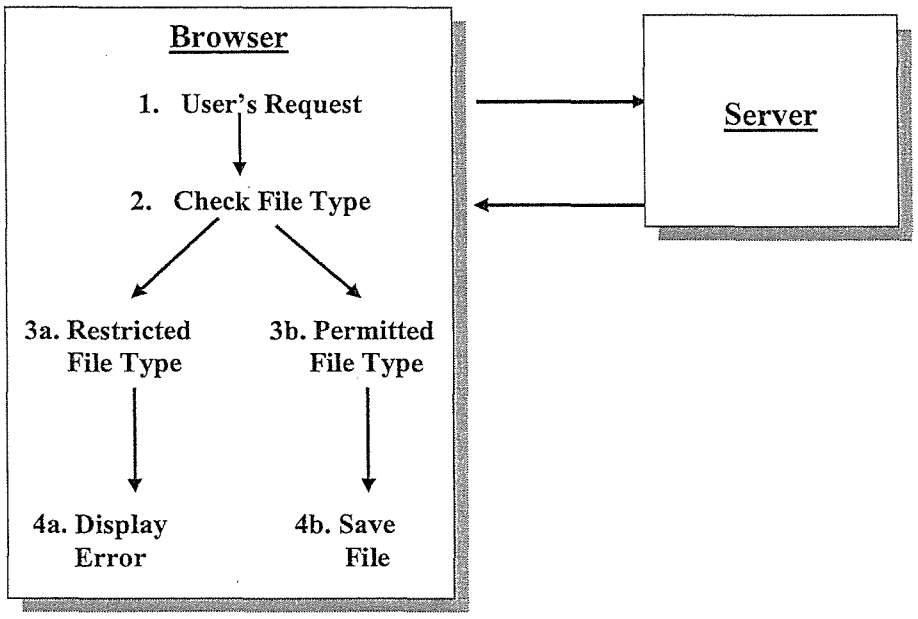

Figure 6.3: MIME type filtering processes.

6.4.1.2 Granting access to valid MIME file type

As discussed in Section 6.4, MIME is used by many Internet applications such as Internet mail services. In general, MIME provides browsers with media type data for identification purposes. The sequence processes for filtering MIME type as shown in Figure 6.3 are:

1. User request URL.

2. The modified Lynx browser sends the request to a server.

3. The server replies with a HTTP header indicating the file type in MIME specific format.

4. The browser checks the file type against the banned MIME type list compiled by administrators.

5. If the requested file type is in the list, the browser displays an error message as shown in Figure 6.4.

6. If the type is permitted, the downloading continues.

This filtering process is tested and found working well. For the testing purpose, the postscript file type was put in the banned list (for example, "application/postscript"). When users tried to access a file with extension ".ps", the request was rejected and a error message (compiled in the messages file of Lynx by the researcher) was displayed as shown in Figure 6.4. However, users successfully accessed any other files with different file extensions, because only ".ps" file extension was not permitted to download in this experiment. The experiment has verified that this method is simple and effective for administrators in restricting users' access on different file types. Therefore, this method is chosen to be used in the secure browser model.

The research also programmed the browser to log every request and its subsequent occurrences such as the request is denied, or downloading is cancelled or completed.

6.4.2 Configuration for Helper Applications

As discussed in Section 6.4.1, browsers use a list of the registered file types and associated applications to determine the downloaded file type. It is important to mention that Windows 2000 and RedHat Linux 7.3 allow only system administrators to modify the registered file types and their associated applications. The model is therefore designed to use the OS's registered file types in finding the associated helper application for a particular file type.

6.5 File Attribute

In Chapter 5 (Section 5.1.2), the research examined the mechanisms for setting attributes in the most commonly used operating systems, Windows 2000 and RedHat

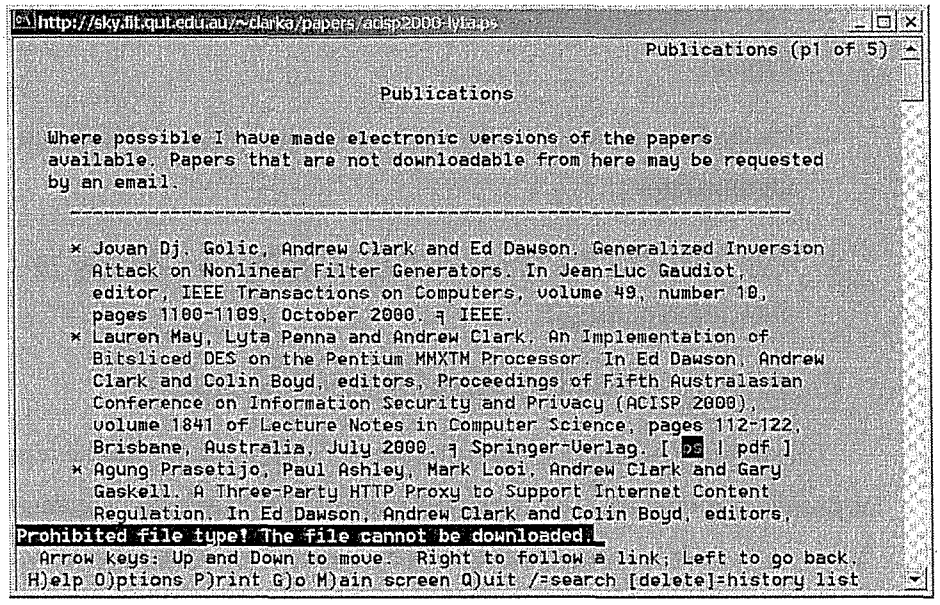

Figure 6.4: Lynx browser displaying an error message.

Linux 7.3. It is noted that RedHat Linux 7.3 has implemented three types of user (owner, group, and others) and each type has three access attributes (read, write, and execute), whereas Windows has created 7 types of access attribute (read, write, execute, delete, change permission, take ownership, and full control) for file and an extra two types (add and list) for directories.

The model has some procedures for requesting and maintaining files and their associated access attributes, which are set by the remote servers. It is noted that access attributes used in UNIX and Windows operating systems (most commonly used OSs) are not compatible or convertible. The model could overcome the problem (only if servers and users use different operating systems) by implementing an option for setting default attributes for files downloaded. For example, file owners and administrators will have "full control" in Windows and "read write execute" in UNIX, but none for others.

By default, Windows 2000 and RedHat Linux 7.3 set full control permission for the file owners (who downloads the file) and read permission for other users for the downloaded files. This means that a user can read other users' files and subsequently confidentiality of the user data fails. The prototype model therefore sets "Full control

for owner and no access for others" as a default attribute by using the "umask" facility , however it can be changed by system administrators. In an experiment with the modified Lynx, the research was able to successfully set the default attributes to downloaded files (e.g, a file owner can read and write but others can neither read nor write). Thus, this component effectively supports security attributes for downloaded files.

6.6 Audit Trail

As discussed in Chapter 5 (Section 5.3), security auditing is necessary because it records users' activities with the associated identity and the time of the action. Administrators can use the audit file to analyse how security problems, if any, occurred or have been initiated, and consequently they can reduce security incidences in browsers and can also identify the responsible users. However, the browser evaluation shows that the analysed browsers do not generate audit data, even though Section 5.3 highlights that the underlying OS (Windows 2000 and RedHat Linux 7.3) have built-in features with which the browsers' programmers can generate audit data and pass it to the OS for secure storage.

To fulfil the much needed audit feature for browsers, the prototype model has a component to generate audit data containing crucial information such as user identification, time, and the activities made. The underlying OS is used to provide user information, accurate time, and a secure storage for audit data. The model records them along with the user requested web pages for viewing or downloading. The audit data is then passed to the operating system for protection and storage purposes.

To demonstrate that a log file can be successfully generated, additional code (for Window 2000; create, open, write, and close file and for Linux 7.3; syslog() utility) is added into Lynx source code to create its own log file. The log file is created the first time the browser is used and a system call is used to write information into the file. Once the file is created, information is added into the file each time a user accesses a web page.

The modified Lynx browser also made use of system calls to obtain user account name, identification number, and time of access. A standard operating system provides user information and system time in response to the system calls "id" and "time"

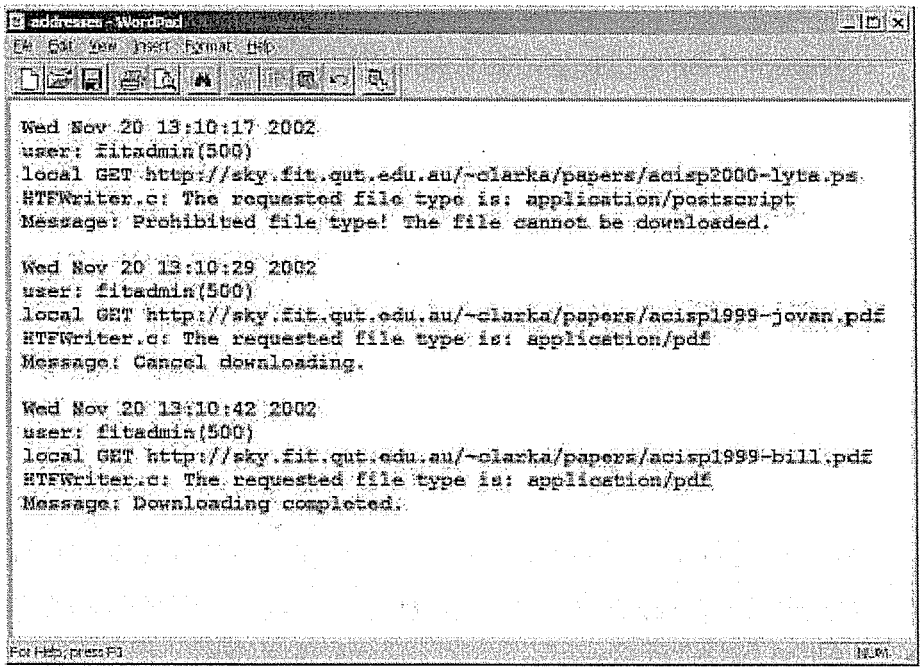

Figure 6.5: The contents of the audit file.

respectively. Since the name of the downloaded file and the visited URLs are also required for audit trail, Lynx is programmed to obtain any request made by users and whether or not the downloading was successful. Thus the modified Lynx browser can successfully intercept all users' requests to record the URL along with user name and time in the log file before it filters the request for file type restriction as described in Section 6.4.1. Note that most browsers create a history file for each user which contains only the visited URL, the web page title, and time of the access. Figure 6.5 shows the contents of a log file generated by the enhanced Lynx browser.

6.7 The Prototype Model Rationale

In Section 6.1, the design requirements is discussed in relation to five security objectives described in the SRP. These design requirements are fully met by the model as shown in Table 6.1 matching the design requirements to the security components of the model.

Design requirements	Model's components
Auditing	Audit trail component
Restriction	User identification component File type limitation component
Authentication	Cryptographic component
Confidentiality	Cryptographic component File transferring component
Integrity	Cryptographic component File transferring component

Table 6.1: Tracing of design requirements to the prototype model's components

6.8 Summary

A prototype design for a secure browser has been discussed with its five components: user identification, cryptographic component, limitation on file type, file transferring process, and audit trails. The prototype model was designed to achieve objectives identified in the preferred SRP for browser (Appendix A). The model also overcomes limitations encountered in the file transferring protocols analysis in Chapter 5. For demonstration purposes, extra components are added into the Lynx browser source code and tested it to prove that the model is achievable.

Chapter 7

Conclusions and Future Directions

This thesis has examined security in browsers which is extremely important. Most users of browsers do not properly understand the risks associated with connecting to, and obtaining data from, a potentially untrusted server on the Internet. Security in browsers was addressed in terms of downloaded data confidentiality and integrity (obtained via authentication), auditing both data transfers and users' activity, and restricting the downloaded file type.

To properly discuss security in browsers, it is necessary to understand what motivation leads toward the browsers development and what services are available in browsers. As browsers have been implemented for accessing information within an organisation, browsers themselves and the services they offer have little security for users. That was fine in those days when computers with a few services were only used for research and military applications. It has however become a major problem now that computers are accessible by everyone, and more services are embedded. The most useful and perhaps dangerous service is the file handling service where a downloaded file is immediately transferred to an appropriate application for further processing.

Browser security requirements can be determined by identifying threats, vulnerabilities, and countermeasures. Threats exist in the connection between clients and servers, in downloaded data content, or on the browser's host. Threats in the connection come from the TCP/IP protocol suite implemented without security considerations in the first place. Since servers are meant to provide the information requested, threats to file content are mostly handled by browsers. Host security is also important because

anyone who administers a host can control the installed browser as well. Many incidents have occurred in the past proving that vulnerabilities exist in browsers. Many of them are caused by flaws in programming. Currently patches for these problems are available but not all administrators apply them in a timely fashion. There are many additional security tools providing host security and connection security, however these solutions can increase the complexity of the problem due to the interaction between the browser itself and additional security tools.

This research used the international security evaluation mechanism, the Common Criteria to assess security in browsers. Following the CC approach, a security requirements profile for browsers was developed.

Browsers utilise a number of protocols and the core protocols among them are the file transferring protocols, the accountable protocols for a browser's main functions. As their functionality and security impact browser security, an analysis of the common file transferring protocols - HTTP, FTP, SCP, and FTAM - was conducted. The research shows that none of these protocols (without TLS) meets the security requirements identified in the SRP, such as audit data generation, user data protection, trusted channels.

However, the thesis includes a prototype model that realises a secure browser. The prototype model was implemented using the Lynx browser source code that is publicly available. With modifications, the browser can be made more secure.

This work has focussed on the browser end and can be extended to the server end, since a secure server would enforce the security requirements defined in the secure browser model. For example, a secure browser may insist on the use of public key cryptography for connections, which can be realistic, only if the connected server supports such a feature and communicates with the client in that particular way.

There is more work required in the area of determining the type of the downloaded file. Currently the MIME mechanism is used to specify file type. The research found that MIME makes use of file extension in file type determination and file extensions are recognised when proprietors register them with IANA [1]. Once a file type has been registered with an extension, all applications acknowledge any file with such extension as it is, even though file extensions can be altered easily.

It is possible, and maybe difficult, to record a file type in relation to its format

as every file has its own way of formatting. To make it simple and effective, the research suggests that a file type is encoded and inserted in a file as a water mark, hence applications can read the code whilst alteration is infeasible. This can be an area for future research.

As has been demonstrated in this thesis, a secure browser can be implemented by modifying current browsers, and if commercial browsers would follow this simple innovation, both users and service providers, in other words clients and servers, can benefit from this research.

Appendix A

Security Requirements Profile for Browsers

Version 1.0a

March 2001

Conventions and Terminology

Conventions

The notation, formatting, and conventions used in this profile follow those given in version 2.1 of the Common Criteria for Information Technology Security Evaluations. The Common Criteria documents may be obtained on-line from the International Common Criteria Project home page (www.commoncriteria.org).

Terminology

In this section, terms used within this profile are defined that are additional to the terminology given by the Common Criteria version 2.1 - Part I.

- **Secure channel:** a communication channel that provides security for data transmitted via that channel.
- **File type:** format of data in a file, the TOE decides this based on the extension of the file.
- **Interpreter application:** the application to which the TOE forwards a downloaded file in order for the file to be processed or displayed.
- **Security relevant action:** actions that are regarded as affecting TOE security, for example, obtaining access to the TOE or its data.
- **Security level:** the selected security assurance level. (e.g., evaluation assurance level - EAL level)
- **User identity:** identity of the user who wishes to use or is using the TOE facility to download a file from a remote server.
- **Object identity:** the name of the requested file to be downloaded.

A.1 Introduction

A.1.1 Identification

Title: Security Requirements Profile for Browsers

Authors: Nwe Nwe Hlaing, Rose-Marie Henderson, and Andrew Clark,
Information Security Research Centre,
Queensland University of Technology,
Brisbane 4001, Australia.

CC Version: 2.1 Final

General Status: Draft

Registration: -

Keywords: file downloading, remote server, file type

A.1.2 Security Requirements Profile (SRP) Overview

This profile defines the security requirements and assurance requirements for a program (software application) used to download files from a remote server. The program downloads a particular file in response to a user's request provided the type of the file is allowed. In addition, this PP specifies security requirements for forwarding the downloaded file to an appropriate program which in turn processes or displays the file.

A.1.3 Related Protection Profiles

This profile could be used in conjunction with the "Rudimentary Web Server Protection Profile" to in addition allow access to local files from remote locations. The TOE described in the SRP relies on the secure operation of the underlying Operating System. Two PPs relating to Operating Systems are the Controlled Access Protection Profile (CAPP) and Labeled Security Protection Profile (LSPP) although these do not cover all functionality required to support TOE operation.

A.2 TOE Description

The TOE is a software application used to find files on a (remote) server and direct the server to fetch and transmit the requested files as appropriate.

The TOE provides user services such as making a connection to the requested file server, making requests for transmission of the desired file, protection of transmitted data, and delivering the obtained files to an appropriate program on the requesting host for interpretation. Figure 1 depicts the logical interaction between the client (with the TOE and an interpreter application) and a server. The TOE proceeds with the request for downloading of the file only if the file type is a permitted file type.

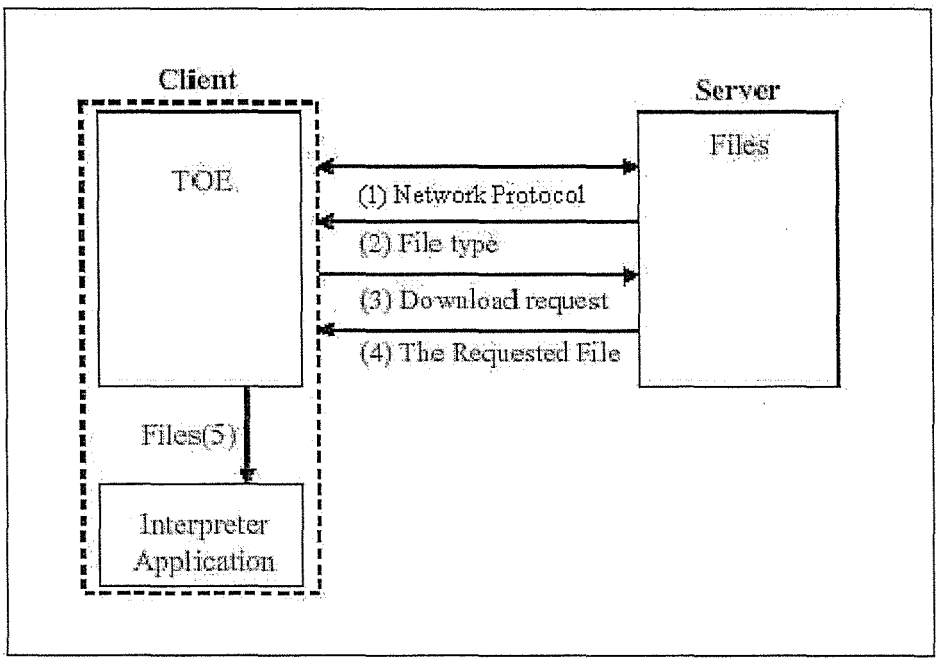

Figure A.1: The browser architecture

The TOE is configurable by the user or an administrator to prevent the download of specified file types. It is assumed that the remote servers have the necessary mechanisms for designating a file's type and conveying this information to the TOE. The TOE is able to set up and maintain a secure channel between the TOE and remote servers in order to protect the transmitted data. This requires the co-operation of the remote server and it is therefore necessary that remote servers, with which the TOE communi-

cates securely, supports this ability. The requirement for a secure channel between the TOE and remote servers is a configurable option by users and the administrator but the administrators configuration will override a users configuration. The administrator's configuration shall override the user's configuration for both file type restrictions and secure channel requirements. Existing protocols such as FTP and HTTP have limitations which are addressed by this TOE. For example the ability for administrators to place restrictions on the file types for downloading (in a configuration file) and mandating secure channels for selected remote servers. This also helps avoid accidental access to files or downloading by users without proper security controls being used.

The SRP is applicable to programs that enable users to remotely obtain files from a server, which responds with data that can be encapsulated in a file and interpreted by an application familiar with the format of that file. The SRP is not intended to cover either the underlying operating system or interpreter applications. These aspects will be separately covered by associated PPs and the TOE will rely on such services being in place. Note that existing PPs relevant to this SRP have been outlined in Section 1.3.

The TOE takes human users as active entities, which request a process that initiates a communication between the TOE and a remote server. The users are identified and authenticated by the operating system before they are given access to the TOE. The operating system is also responsible for controlling access to the configuration data of the TOE. Administrators are able to manage a configuration file for the TOE, which shall meet an organisation's security policies. Users may limit their own personal configuration file but the configuration file of the administrator overrides any user configuration.

A.3 TOE Security Environment

A.3.1 Secure Usage Assumptions

- A.ACCESS: All access to the TOE and its data is controlled by the Operating System (including TOE audit data).
- A.STORAGE: The Operating System provides secure storage for all TOE data (including all TOE audit data).
- A.AUTHN: The Operating System will authenticate a user prior to granting access to the TOE.
- A.IDENTITY: The Operating System provides the TOE with all required user identification information.
- A.MALICIOUS: Downloaded files are checked by the Operating System to determine if they contain malicious data.
- A.ADMIN: System administrators manage the TOE and the information it contains, and can be trusted not to abuse their privileges.
- A.SERVER: The servers, with which the TOE interacts, are regarded as trustworthy.
- A.OSCNTRL: The Operating System securely controls all information exchange with the TOE.

A.3.2 Threats to Security

- T.ABUSE: Authorised users may perform actions that are not permitted.
- T.ADMIN: Administrators may commit errors in installation, configuration, or management of the TOE, compromising security.
- T.AUDIT: Users may perform undetected security relevant actions.
- T.CAPTURE: A file or data transferred between a remote server and the TOE may be exposed and captured by an illegitimate user.

- T.INTG: Modifications during transmission may lead to alteration of the file type or the transmitted data.
- T.SPOOF: A server may be spoofed.
- T.SWFLAW: TOE software flaws may allow users to download files of illicit file type or result in the TOE forwarding the downloaded file to an inappropriate application.
- T.CRASH: Human error, a failure of software, hardware or environmental support (e.g. power failure) may cause an abrupt interruption to TOE operation.

A.3.3 Organisational Security Policies

- P.ACCOUNT: Users shall be held accountable for their actions.

A.4 Security Objectives

A.4.1 Security Objectives for the TOE

- O.AUDIT: The TOE must provide the means for recording security relevant events to hold individual users accountable for the actions they perform.
- O.RESTRICT: The TOE must limit access to authorised users and to download of files of permitted files type and in accordance to configurations requiring secured exchanges with remote server where any limitations imposed by the administrator override those of all users.
- O.AUTHN: The TOE must provide an authentication mechanism to verify the identity of servers with which it interacts.
- O.CONFD: The TOE must ensure that data or files can not be illegitimately obtained by the unauthorised person during transmission.
- O.INTG: The TOE must be able to indicate that the integrity of downloaded files and data is maintained during transmission.

A.4.2 Security Objectives for the Environment

- OE.OS: The Operating System with an appropriate security level must provide security relevant information and services to the TOE (such as access to TOE data, secure storage for TOE data, authenticating users before granting access to the TOE, providing using identity information to the TOE, and checking downloaded data for malicious data).
- OE.TRAIN: Administrators and users must be adequately trained to manage and operate the TOE and maintain TOE security.
- OE.SERVER: The TOE must only be allowed to interact with trustworthy servers.
- OE.OPERATE: Those responsible for the TOE must ensure that the TOE is installed, configured, managed, and operated in a secure manner.
- OE.SUPPORT: Those responsible for services external to TOE, but necessary to TOE operation, must maintain continuity of these services.
- OE.ACCOUNT: An audit mechanism must be in place to provide user accountability.
- OE.OSCNTRL: The Operating System must secure and control all information exchanges between itself and the TOE.

A.5 IT Security Requirements

This section outlines the security requirements selected to meet the objectives identified in Section 4. The security requirements are Security Functional Requirements (SFRs) selected from Part 2 of the CC or Security Assurance Requirements (SARs) selected from Part 3 of the CC.

A.5.1 TOE Security Functional Requirements

Table 1 outlines the SFRs selected to meet the TOE objectives. All dependencies are met excepting the dependency on FMT_MSA.3. A discussion of dependencies is included in Section 6.2 Security Requirements Rationale.

Class	Functional Component	Dependencies
Security audit (FAU)	FAU_GEN.1 Audit data generation	FPT_STM.1
	FAU_GEN.2 User identity association	FAU_GEN.1 FIA_UID.1
User data protection (FDP)	FDP_IFC.1 Subset information flow control	FDP_IFF.1
	FDP_IFF.1 Simple security attributes	FDP_IFC.1 FMT_MSA.1 FMT_MSA.3 FMT_SMR.1
Identification and Authentication (FIA)	FIA_UID.2 User identification before any action	None
Protection of the TOE Security Functions (FPT)	FPT_STM.1 Reliable time stamps	None
Trusted path/channels (FTP)	FTP_ITC.1 Inter-TSF trusted channel	None

Table A.1: Security Functional Requirements

A.5.1.1 Security Audit (FAU)

Audit data generation (FAU_GEN.1)

- (FAU_GEN.1.1): The TSF shall be able to generate an audit record of the following auditable events:

  a) Start-up and shutdown of the audit functions;

  b) All auditable events for the [selection: basic] level of audit; and

  c) [assignment: *other specifically defined auditable events*].

  Note: auditable events are given in Table A.2.

- (FAU_GEN.1.2): The TSF shall record within each audit record at least the following information:

  a) Date and time of the event, type of event, subject identity, and the outcome (success or failure) of the event; and

  b) For each audit event type, based on the auditable event definitions of the functional components included in the PP/ST, [assignment: *other audit relevant information*]

User identity association (FAU_GEN.2)

- (FAU_GEN.2.1) The TSF shall be able to associate each auditable event with the identity of the user that caused the event.

Functional Component	Level	Auditable Event
FDP_IFF.1	Basic	All decisions on requests for information flow.
FIA_UID.2	Basic	All use of the user identification mechanism, including the user identity provided
FPT_STM.1	Minimal	Changes to the time.
FTP_ITC.1	Basic	All attempted uses of the trusted channel functions. Identification of the initiator and target of all trusted channel functions.

Table A.2: Auditable Events

A.5.1.2 User Data Protection (FDP)

File Download (information flow control) SFP: Requests from authorised users for a file download from a remote server shall be permitted subject to the following conditions:

1. If the file type is a permitted file type according to the administrators configuration.

2. If the file type is a permitted file type according to the users configuration.

3. A secure communication channel can be established with the remote server when required by administrator configuration settings.

4. A secure communication channel can be established with the remote server when required by the users configuration settings.

Note: Alterations to configurations settings are controlled by the Operating System which passes this information as well as the identity to the TOE for execution of the downloading of files.

Subset information flow control (FDP_IFC.1)

- (FDP_IFC.1.1) The TSF shall enforce the [assignment: *File Download SFP*] on [assignment: *all user requests for downloading of a file*].

Simple security attributes (FDP_IFF.1)

- (FDP_IFF.1.1): The TSF shall enforce the [assignment: *File Download SFP*] based on the following types of subject and information security attributes: [assignment: *the users identity, the file type, the ability of the TOE and remote server to establish a secure connection for communicating the file from the remote server to the TOE*].

- (FDP_IFF.1.2): The TSF shall permit an information flow between a controlled subject and controlled information via a controlled operation if the following rules hold: [assignment: *subject to the conditions outlined in the File Download SFP*].

- (FDP_IFF.1.3): The TSF shall enforce the [assignment: *additional information flow control SFP rules*].

- (FDP_IFF.1.4): The TSF shall provide the following [assignment: *list of additional SFP capabilities*].

- (FDP_IFF.1.5): The TSF shall explicitly authorise an information flow based on the following rules: [assignment: *rules, based on security attributes, that explicitly authorise information flows*].

- FDP_IFF.1.6: The TSF shall explicitly deny an information flow based on the following rules: [assignment: *rules, based on security attributes, that explicitly deny information flows*].

A.5.1.3 Identification and Authentication (FIA)

User identification before any action (FIA_UID.2)

- (FIA_UID.2.1): The TSF shall require each user to identify itself before allowing any other TSF-mediated actions on behalf of that user.

A.5.1.4 Protection of the TOE Security Functions (FPT)

Reliable time stamps (FPT_STM.1)

- (FPT_STM.1.1) The TSF shall be able to provide reliable time stamps for its own use.

A.5.1.5 Trusted Path/Channels (FTP)

Inter-TSF trusted channel (FTP_ITC.1)

- (FTP_ITC.1.1): The TSF shall provide a communication channel between itself and a remote trusted IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure.
- (FTP_ITC.1.2): The TSF shall permit [selection: *the TSF*] to initiate communication via the trusted channel.
- (FTP_ITC.1.3): The TSF shall initiate communication via the trusted channel for [assignment: *communications between and downloading of files from remote servers subject to the user and administrator configuration files subject to the condition that administrator configurations override those of users*].

A.5.2 TOE Security Assurance Requirements

The Security Assurance Requirements (SARs) selected for this PP are given in Table 3 below. The selected SARs are identical to those of the Evaluation Assurance Level EAL3 from Part 3 of the CC.

Assurance Class	Assurance Components
Class ACM	ACM_CAP.3 Authorisation controls
Configuration Management	ACM_SCP.1 TOE CM coverage
Class ADO	ADO_DEL.1 Delivery procedures
Delivery and Operation	ADO_IGS.1 Installation, generation, and start-up procedures
Class ADV	ADV_FSP.1 Informal functional specification
Development	ADV_HLD.2 Security enforcing high-level design
	ADV_RCR.1 Informal correspondence demonstration
Class AGD	AGD_ADM.1 Administrator guidance
Guidance Documents	AGD_USR.1 User guidance
Class ALC	ALC_DVS.1 Identification of security measures
Life Cycle Support	
Class ATE	ATE_COV.2 Analysis of coverage
Tests	ATE_DPT.1 Testing: high-level design
	ATE_FUN.1 Functional testing
	ATE_IND.2 Independent testing - sample
Class AVA	AVA_MSU.1 Examination of guidance
Vulnerability Assessment	AVA_SOF.1 Strength of TOE security function evaluation
	AVA_VLA.1 Developer vulnerability analysis

Table A.3: Security Assurance Requirements: EAL3

A.5.3 Security Requirements for the IT Environment

The following requirements are selected as SMT_MSA.3 is required by the TOE but is to be performed by the Operating System with which the TOE interacts. It is necessary

that the Operating System manage all of the configuration files and access to those files securely as the TOE depends upon this. These requirements are all dependencies relating to SMR_MSA.3 and operations have been performed to specifically tailor them to the TOE requirements.

A.5.3.1 Identification and authentication (FIA)

Timing of identification (FIA_UID.1)

- (FIA_UID.1.1): The TSF shall allow [assignment: *no actions*] on behalf of the user to be performed before the user is identified. This includes access to the TOE and TOE related data.
- (FIA_UID.1.2): The TSF shall require each user to be successfully identified before allowing any other TSF-mediated actions on behalf of that user.

A.5.3.2 Security Management (FMT)

Management of security attributes (FMT_MSA.1)

- (FMT_MSA.1.1): The TSF shall enforce the [assignment: *(named) access control SFP*] to restrict the ability to [selection: *query, modify* [assignment: *other operations*]] the security attributes [assignment: *a users configuration files*] to [assignment: *the authorised user*].
- (FMT_MSA.1.1): The TSF shall enforce the [assignment: *(named) access control SFP*] to restrict the ability to [selection: change default, query, modify, delete, [assignment: *other operations*]] the security attributes [assignment: *a users configuration files or administrators configuration file*] to [assignment: *the authorised user*].

Static attribute initialisation (FMT_MSA.3)

- (FMT_MSA.3.1): The TSF shall enforce the [assignment: *(named) access control SFP*] to provide [selection: *restrictive*] default values for security attributes that are used to enforce the SFP.

- (FMT_MSA.3.2): The TSF shall allow the [assignment: *the authorised administrator*] to specify alternative initial values to override the default values when an object or information is created.

Security roles (FMT_SMR.1)

- (FMT_SMR.1.1): The TSF shall maintain the roles [assignment: *the administrator, users*].
- (FMT_SMR.1.2): The TSF shall be able to associate users with roles.

A.6 Rationale

A.6.1 Security Objectives Rationale

A.6.1.1 Assumptions

- A.ACCESS: *All access to the TOE and its data is controlled by the Operating System (including TOE audit data).*
  The environmental objective OE.OS ensures that the underlying operating system provides required supporting services for the TOE. This includes controlling all access to the TOE and its data.
- A.STORAGE: *The Operating System provides secure storage for all TOE data (including all TOE audit data).*
  The environmental objective OE.OS ensures that the underlying operating system provides required supporting services for the TOE. This includes secure storage for all TOE data such as configuration data, audit data, any appropriate security attributes, etc.
- A.AUTHN: *The Operating System will authenticate a user prior to granting access to the TOE.*
  The environmental objective OE.OS ensures that the underlying operating system provides required supporting services for the TOE. This includes authenticating user identity before granting access to the TOE.
- A.IDENTITY: *The Operating System provides the TOE with all required user identification information.*

The environmental objective OE.OS ensures that the underlying operating system provides required supporting services for the TOE. This includes providing user identity information to the TOE as required.

- A.MALICIOUS: *Downloaded files are checked by the Operating System to determine if they contain malicious data.*
  The environmental objective OE.OS ensures that the underlying operating system provides required supporting services for the TOE. In particular, this includes determining if downloaded files contain malicious data.

- A.ADMIN: *System administrators manage the TOE and the information it contains, and can be trusted not to abuse their privileges.*
  The environmental objective OE.TRAIN ensures that the system administrators are adequately trained to manage and operate the TOE in a manner that maintains TOE security.

- A.SERVER: *The servers, with which the TOE interacts, are trustworthy.*
  The environmental objective OE.SERVER stipulates that the TOE must be limited to interact with servers that are regarded as trustworthy (in terms of data confidentiality, data integrity, and file type specification), directly covering this assumption.

- A.OSCNTRL: *The Operating System securely controls all information exchange with the TOE.*
  This assumption is covered with the environmental objective OE.OCNTRL which ensures that the Operating System with which the TOE interacts is able to secure and control all information exchanges.

A.6.1.2 Threats

- T.ABUSE: *Authorised users may perform actions that are not permitted.*
  This type of threat can occur as a result of either an intentional or accidental user action. One objective to use to counter this threat is O.AUDIT as it ensures that records of user security relevant activities are kept. Such records are necessary to enable later review and hold users accountable for their actions. The objective

O.AUDIT also supports prevention of this threat as users may not knowingly perform actions that are not permitted if they know such actions are being recorded. The objective O.RESTRICT ensures that users are authenticated and limited to downloading permitted files and under appropriate security controls as set by the administrator, but also helps prevent user errors in this regard by submitting users to their own predefined settings. The proper configuration of the TOE, enforced by the environmental objective OE.OPERATE, directly counters this threat as it prevents such actions from being permitted in the first place.

- T.ADMIN: *Administrators may commit errors in installation, configuration, or management of the TOE, compromising security.*
  In this SRP, administrators are assumed to be non-hostile. However, they still can make errors. This kind of threat is difficult to prevent, but detection will at least bring attention to the error which in turn can help alleviate the problem of administrator errors. The objective O.AUDIT addresses this threats by enabling detection and tracing of a security breach that occurs due to administrator error. The objective Prevention of this threat can be supported by training administrators. Therefore the objective OE.TRAIN counters this threat as it requires that the administrators are properly trained, promoting secure operation of the TOE.

- T.AUDIT: *Users may perform undetected security relevant actions.*
  This threat cannot be directly prevented, but it can be detected by analysing the audit trail record. The objective O.AUDIT addresses recording user's security relevant activities.

- T.CAPTURE: *A file or data transferred between a remote server and the TOE may be exposed and captured by an illegitimate user.*
  It is relatively easy to capture data during transmission. As it is impossible to prevent data capture without assuming control of the transmission media, the TOE must be capable of making the transmitted data unreadable to unauthorised entities. If the data is unreadable, then an attacker gains no knowledge of the data even if they obtain it. Specifically, if this is the case, then the data is not considered to be captured or exposed. Therefore this threat is addressed by the security objective O.CONFD which ensures that transmitted data or files are not

illegitimately obtained. The threat posed by traffic analysis of transmissions is not countered in this PP as it is not deemed to be sufficiently relevant to the focus of this PP. However, products similar in service to this TOE may provide additional security measures to address such concerns or such threats may be addressed by other products performing separate functions.

- T.INTG: *Modifications during transmission may lead to alteration of the file type or the transmitted data.*
  Currently, the best way to counter this threat is to check transferred files and data to establish that their integrity has been maintained. The objective O.INTG provides for the required data and file integrity checking.

- T.SPOOF: *A server may be spoofed.*
  Generally, it is difficult for a user or application accessing a server to determine whether the server is being spoofed or not. To counter spoofing threats it is necessary to provide some way of verifying the identity of the server. The objective O.AUTHN directly counters this threat by enabling the servers identity to be authenticated. However, this service is optional but may be enforced by the administrator where considered necessary.

- T.SWFLAW: *TOE software flaws may allow users to download files of illicit file type or result in the TOE forwarding the downloaded file to an inappropriate application.*
  It is difficult to develop flawless software. This becomes even more difficult as the software becomes more complicated. One way to prevent and minimise flaws in software is to ensure that it is developed using sound implementation guidelines and practices. This is addressed by the environmental objective OE.OPERATE which may therefore be applied to counter this threat. More formal software engineering techniques are not deemed appropriate for this TOE as the TOE itself relies on the correct operation of the underlying operating system and the associated software would be too complicated to permit such analysis.

- T.CRASH: *Human error, a failure of software, hardware or environmental support (e.g. power failure) may cause an abrupt interruption to TOE operation.*

This threat can be countered by ensuring the correct operation of the TOE with the objective OE.OPERATE and by ensuring that those responsible for providing services external to the TOE but necessary for its operation do all that they can to maintain them with the objective OE.SUPPORT.

A.6.1.3 Organisational Security Policies

- P.ACCOUNT: *Users shall be held accountable for their actions.*
  This policy simply states the requirement to provide for user actions to be traceable. In particular, the threat T.AUDIT focuses on the problem where user security relevant actions are not traceable. This threat is countered by O.AUDIT which covers the recording security relevant events and so partially meets this policy. However, to completely cover this policy, other user actions need to be traceable but this is beyond the scope of O.AUDIT. Therefore, the environmental objective OE.ACCOUNT is selected to fully address this policy requirement

A.6.2 Security Requirements Rationale

The arguments given in this section demonstrate that the set of security requirements identified in Section 5 are suitable and appropriate to meet the security objectives identified in Section 4.

O.AUDIT

The TOE must provide the means for recording security relevant events to hold individual users accountable for the actions they perform.

To meet this objective, an audit record with the minimum of essential information consisting of the user identity, details of the action performed, and a reliable time stamp, is required. Note that the TOE is not itself responsible for the related services covering the analysis and storage of the audit data as this data is passed back to be managed by the Operating System. Note that the requirement for the audit data to be securely transmitted to the Operating System (for analysis and storage) from the TOE has been included as an assumption (A.OSCNTRL). The following components are used to satisfy the objective:

Policy/Threat/Assumption	Objectives
A.ACCESS	OE.OS
A.STORAGE	OE.OS
A.AUTHN	OE.OS
A.IDENTITY	OE.OS
A.MALICIOUS	OE.OS
A.ADMIN	OE.TRAIN
A.SERVER	OE.SERVER
T.ABUSE	O.AUDIT, O.RESTRICT, OE.OPERATE
T.ADMIN	O.AUDIT, OE.TRAIN
T.AUDIT	O.AUDIT
T.CAPTURE	O.CONFD
T.INTG	O.INTG
T.SPOOF	O.AUTHN
T.SWFLAW	OE.OPERATE
T.CRASH	OE.OPERATE, OE.SUPPORT
P.ACCOUNT	O.AUDIT, OE.ACCOUNT

Table A.4: Mapping the TOE Security Environment to Security Objectives

1. FAU_GEN.1: This component defines the auditable events and generates a audit record providing the required audit information (see Table 2). By ensuring that reliable audit records are kept, administrators are able to use these records to detect the misuse or unauthorised activity of the users.

2. FAU_GEN.2: This component links audit events and the users who cause the events. By having the reliable audit records including associated users, administrators are able to detect the misuse or unauthorised activity of the users.

Objectives	Policy/Threat/Assumptions
O.AUDIT	T.ABUSE, T.ADMIN, T.AUDIT, P.ACCOUNT
O.RESTRICT	T.ABUSE
O.AUTHN	T.SPOOF
O.CONFD	T.CAPTURE
O.INTG	T.INTG
OE.OS	A.ACCESS, A.STORAGE, A.AUTHN, A.IDENTITY, A.MALICIOUS
OE.TRAIN	A.ADMIN, T.ADMIN
OE.SERVER	A.SERVER
OE.OPERATE	T.ABUSE, T.CRASH
OE.SUPPORT	T.CRASH
OE.ACCOUNT	P.ACCOUNT

Table A.5: Tracing of Security Objectives to the TOE Security Environment

3. FPT_STM.1: This component provides an accurate time that is recorded along with each security event. The reliable time is critical for audit records as they assist administrators to detect the sequence or/and pattern of the security related events.

In summary, FAU_GEN.1 defines the conditions for generation of the necessary audit records along with the associated user identity, provided by FAU_GEN.2. The record contains a time of the user's activity which is provided by the FTP_STM.1. Note that the assumption A.OSCNTRL enables the necessary time information to be obtained from the Operating System but that the TOE is responsible for linking the time of an event to other event details.

O.RESTRICT

The TOE must limit access to authorised users and to download of files of permitted

files type and in accordance to configurations requiring secured exchanges with remote server where any limitations imposed by the administrator override those of all users.

This objective requires that users of the TOE are identified before accessing TOE services and that the file downloaded is a permitted files types (determined from the administrator and user configuration file) and that the connection is able to be properly secured. The TOE shall be able to establish a secure channel but it is also necessary that the remote server can support this service if it is required in the configuration file. The following components are included in order for the TOE to cover this objective.

1. FIA_UID.2 User identification before any Action.

   This objective requires that users are identified before any other action is taken by the TOE. The identification information is provided to the TOE by the underlying operating system. Thus the TOE will not respond to any request from a user that has not been identified by the Operating System.

2. FDP_IFC.1 Subset information flow control.

   This component ensures that all user request are subject to the identified rules of the File Download SFP. This will not permit users to mistakenly download files of file type that they do not wish to download or to either mistakenly or deliberately download files of a file type not permitted by the administrator. This is also holds for cases where secure communication has been stipulated by the user or administrator.

3. FDP_IFF.1 Simple security attributes.

   This component supports the previous one and restricts all users to the rules outlined in the File Download SFP in regards to user identity, the file type and whether a secure connection for communicating the file is required. Subjects for this component are of two different type: subjects that cause the information flow and subjects that act as recipient of the information. It also defines rules on information flow to be permitted or denied. By having security attributes and rules for the information flow, it is ensured that the file transmissions are only those allowed by the File Download SFP.

In summary, FIA_UID.2 ensures that the user has been identified before they are able to access TOE services while FDP_IFF.1 and FDP_IFC.1 together ensure that the

information flows are subject to the rules defined in the File Download SFP. Note that the assumption A.OSCNTRL enables the necessary information regarding user identity and configuration file information to be obtained from the Operating System but that the TOE is responsible for limiting file transfer between the remote servers and the TOE subject to the File Download SFP.

O.AUTHN

The TOE must provide an authentication mechanism to verify the identity of servers with which it interacts.

This objective requires that the TOE supports a mechanism to verify servers to gain a level of trust regarding their identity. The TOE shall use a secure channel in which the entire authentication process is accomplished. The following component is included in order for the TOE to implement the objective.

1. FTP_ITC.1 Inter-TSF trusted channel.
   This component provides a communication channel separated from other channels and assures the identification of connecting servers. By having trusted channels, the TOE can securely authenticate a server and a level of trust is consequently built for the data downloaded from the server.

Note that the TSF is allowed to initiate the trusted channel but that this is only required when mandated within either the users or administrators configuration file, subject to the condition that configurations of the administrator override those of the user. This enables service to be provided between servers that do not support establishment of a trusted channel but ensures that the TOE can support this service.

O.CONFD

The TOE must ensure that data or files can not be illegitimately obtained during transmission.

This objective aims to limit disclosure of the downloaded files to authorised users. To achieve this objective, the TOE shall provide a secure channel through which data are downloaded. In order to accomplish the objective, the following components have been selected for the TOE.

1. FTP_ITC.1 Inter-TSF trusted channel

   This component provides a secure communication channel separated from other channels and assures that the information transmitted through this channel can not be obtained by any unintended recipient. In other words, having trusted channels ensures that the data, during transmission, can not be seen by any unauthorised users, therefore confidentiality of the transferred data is attained.

2. FDP IFF.1 Simple security attributes

   This component ensures that all user requests are subject to their identity. This will not permit users to mistakenly access other users' downloaded files or configuration files.

Note that the TSF is allowed to initiate the trusted channel but that this is only required when mandated within either the users or administrators configuration file, subject to the condition that configurations of the administrator override those of the user. This enables service to be provided between servers that do not support establishment of a trusted channel but ensures that the TOE can support this service.

O.INTG

The TOE ensures the integrity of downloaded files and data is maintained during transmission.

This objective is required as it preserves the integrity of the data during transit from the remote server. Like some of other objectives, the objective is implemented using a trusted channel. The following components cover this objective.

1. FTP_ITC.1 Inter-TSF trusted channel

   This component provides a communication channel separated from other channels. A trusted channel will prevent unauthorised users from modifying the transmitted data in any way.

Note that the TSF is allowed to initiate the trusted channel but that this is only required when mandated within either the users or administrators configuration file, subject to the condition that configurations of the administrator override those of the user. This enables service to be provided between servers that do not support establishment of a trusted channel but ensures that the TOE can support this service.

Objectives	Security Functional Requirements
O.AUDIT	FAU_GEN.1, FAU_GEN.2, FPT_STM.1
O.RESTRICT	FIA_UID.2, FDP_IFF.1, FDP_IFC.1
O.AUTHN	FTP_ITC.1
O.CONFD	FTP_ITC.1
O.INTG	FTP_ITC.1

Table A.6: Mapping of Objectives to Security Functional Requirements

Security Functional	Objectives
FAU_GEN.1	O.AUDIT
FAU_GEN.2	O.AUDIT
FDP_IFC.1	O.RESTRICT
FDP_IFF.1	O.RESTRICT
FIA_UID.2	O.RESTRICT
FPT_STM.1	O.AUDIT
FTP_ITC.1	O.AUTHN, O.CONFD, O.INTG

Table A.7: Tracing of the security Functional Requirements to the Security Objectives

The SFRs, selected to meet the TOE objectives, can be seen in Table A.6 and Table A.7. These requirements, in their entirety, have been selected from Part 2 of the Common Criteria (version 2.1). There is a single dependency (FMT_MSA.3) that has not been met by any of the SFRs for the TOE. This SFR and its dependencies outline requirements for managing the configuration files relating to the permitted file types for downloading and requirements for secure channel between the TOE and remote server for each user. They also outline the management functions associated to maintaining user identity associations to these configuration files. These services are beyond the simple function of the TOE and are controlled by the underlying Operating System.

For this reason they have been included as requirements for the IT environment.

The set of functions selected to cover the TOE security objectives include: audit generation functions (FAU class), information flow control function (FDP class), restrictions for identification before any actions (FIA class), timing functions (FPT class) and trusted channel functions (FTP class). The TOE is responsible for audit generation but this is then passed back to the Operating System for processing. For this reason the TOE is not required to support any audit analysis tools or audit management functions. Information flows from remote servers to the TOE must be subject to the rules outlined in the File Download SFP and this ensures the TOE objectives for permitted file transfers are met. Even though the operating system ensures that all users are identified before access is granted to the TOE it should not be possible for anyone to circumvent this mechanism motivating the selection of the requirement covering identification before any TOE action. The timing function is simply related to enable the ability to closely link actions with the time that they occurred, an important aspect to support useful and relevant audit recording. Although not all connections to remote servers will require a trusted channel the provision of this service is central to the TOE. Although it would limit TOE usability to restrict all file transfers to be secured, it is essential to the TOE service to provide for some transfers that may contain sensitive material and therefore require sufficient security measures to be in place.

A.6.3 Assurance Security Requirements Rationale

This profile has been developed for circumstances where a moderate level of security is required. The selected security level is identical to EAL3 which includes high level design and configuration management control, testing for all parts of the TOE, and documentation of the TOE. There are two issues to be considered in regards to the selection of the assurance level EAL3. Firstly, the TOE is dependent on the underlying Operating System in terms of user identification, authentication, and access control and for data storage. Secondly, the TOE assists users downloading files from remote servers, some of which may be designated as trusted with communications between the TOE and the remote serves being secured. These components are therefore necessarily related components affecting the TOEs secure operation. In order to support and maintain the TOE's assurance level, the user identification and authentication

mechanism and the access control mechanism in underlying operating system, and the servers shall also have to have an appropriate EAL level of EAL3 or above. To match with current and expected best practice EAL3 is deemed appropriate as it provides a degree of assurance to support the secure interchange of files between trusted servers.

A.6.4 Strength of Function

While identification and authentication are handled by the underlying Operating System, a browser needs a trusted channel between servers and itself. Since the TOE security functions are realised by a probabilistic or permutation mechanism, the SOF medium level (adequate protection against straightforward or intentional breach of TOE security by attackers possessing a moderate attack potential) has been chosen.

Acronyms

CC - Common Criteria
EAL - Evaluation Assurance Level
IT - Information Technology
PP - Protection Profile
SF - Security Function
SFP - Security Function Policy
SOF - Strength of Function
SRP - Security Requirements Profile
ST - Security Target
TOE - Target of Evaluation
TSC - TSF Scope of Control
TSF - TOE Security Functions
TSP - TOE Security Policy

References

Common Criteria for Information Technology Security Evaluation; Part 2: Security Functional Requirements, CCIB-99-032, August 1999

Common Criteria for Information Technology Security Evaluation; Part 3: Security Assurance Requirements, CCIB-99-033, August 1999

Controlled Access Protection Profile, Version 1.d, October 1999.

Labeled Security Protection Profile, Version 1.b, October 1999.

http://niap.nist.gov/cc-scheme/PPRegistry.html

Appendix B

File Transfer Protocols

B.1 File Transfer Protocol (FTP)

The file transfer protocol provides user level file transfer between two hosts. It allows convenient use of storage and sharing of files for two different types of file handling capability. FTP uses two specific sockets, one is for an initial connection protocol, and the other for standard data transferring and related operations [48].

B.1.1 General Functionality

FTP can be used indirectly (without login) or directly logging into a remote host. FTP has a mechanism for verifying user identity and passwords for exchange of access information. Username and password identifiers contain information of the respective identification and are sent by a user to a server when the connection is established. The server may send the user an error message indicating an access control violation if it receives incorrect identification information. FTP remote login processes are portrayed in the figure B.1.

FTP considers a file as an ordered set of arbitrary length composed with computer data including system instructions [48]. The nature of information in the file is not restricted by FTP, but it does indicate the type of data for parsing, interpretation, reconfiguration, and storage purposes. FTP allows the programmer to extend its commands and transferable data types. A number of data types are defined by FTP but a host must accept the data transferred even if it does not recognise the data type. If this

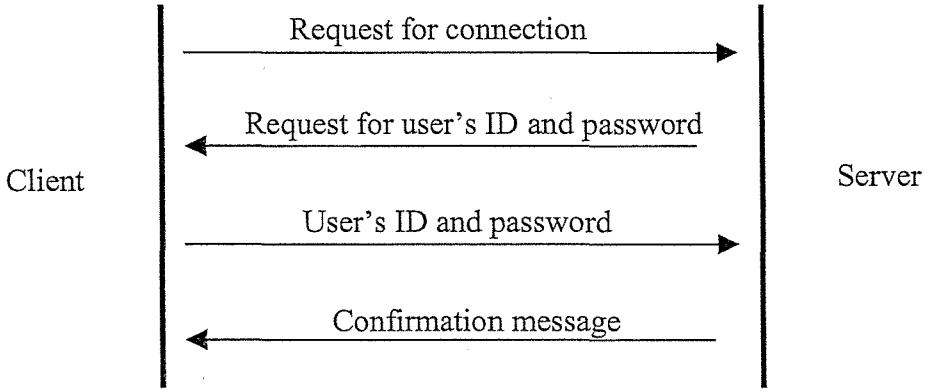

Figure B.1: Login process of FTP

is the case, the data is treated as a bit stream (binary data).

A file transferred by FTP may not come with access control attributes. It is a responsibility of a resident file system to provide the transferred files with access control attributes and protection.

The Data Transfer Protocol (DTP) specified in RFC 171 [7] is used for transferring data in FTP. FTP transfers more than a single control transaction over the same connection. There are a number of transaction types defined in RFC 171 but only a few of them are utilised for the FTP mechanism. In fact, there are only two types implied in FTP as listed below.

- Type B9 - Transparent block-control
- Type BA - Descriptor and counts-control

With data transferring, FTP does not define the structure of files therefore it uses a file separator to indicate the end of the file. In indefinite bit streams where there is no file separator, the connection is closed at the end of the transaction. Control transactions include requests, identifiers, and terminations. Control transactions can be distinguished by their first byte which is called "op code".

A FTP request from a client host contains a unique pathname. The syntax of the pathname should comply with the server host convention. Common requests in FTP are store, retrieve, append, rename, and delete. It is necessary for a server to send an

acknowledgement if the transfer occurs between the server and the client. Any host can send an abort message to terminate the connection for any reason. For a more drastic type of abort, a host can close the connection. If this happens then the closing host has a responsibility to reopen the connection.

B.1.2 Overview Analysis on Protocol Functions

As a summary, FTP can identify users and provide the timing of the operation. The FTP protocol simply obtains these security related information with the help of the underlying operating system. In fact, these functions, supported in FTP, are fundamental requirements for the underlying operating system. Hence, security in the FTP protocol is what the underlying operating system provides in terms of security measures.

B.2 SCP and SSH

The Secure Copy program is designed to facilitate file copying in a secure manner on networks. Secure Shell (SSH) is used for data transferring in SCP [38]. SCP and SSH work like FTP and TELNET, respectively. The difference is that data is transferred in plain text in FTP while SCP encrypts data for transmission.

B.2.1 General Functionalities

SSH provides a secure connection over insecure networks, that allows users to access to a remote host in a security means [38]. There are three security features which can be obtained from the SSH mechanism.

Authentication of entity

The users and the remote host can mutually authenticate in a number of different ways: password authentication, RSA authentication, Kerberos authentication, Rhost authentication, or no authentication if desired.

Integrity of data

Data integrity is obtained when SSH computes a MAC (message authentication code)

and places it in every data packet [38].

Port forwarding security feature for other applications

Some application's data stream can travel across networks under SSH protection. Their protocols use SSH port for transferring purpose. At the end of the channel, the appropriate port handles them for further processing. For example POP receives messages through SSH port (default number 22) in order to take advantage of SSH facilities - a secure channel.

In order to provide the above security features, SSH consists of three protocols as follows:

1. SSH transport layer protocol: Responsible for encryption, decryption, compression, and decompression.
2. SSH connection protocol: Responsible for sending and receiving data.
3. SSH authentication protocol: Responsible for authentication, handshaking, negotiating cryptographic keys and algorithm. The detail communications involved in the authentication mechanism of SSH are depicted in Figure B.2.

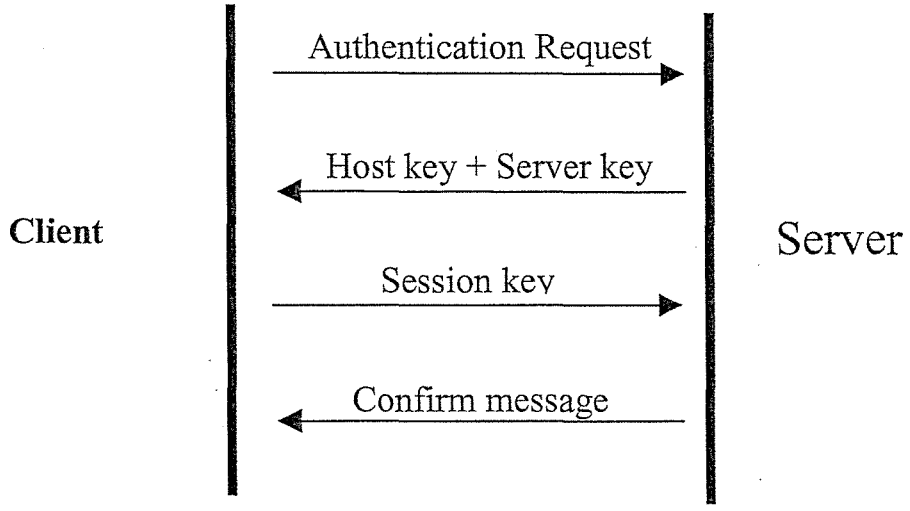

Figure B.2: Login process of SSH

B.2.2 SCP and SSH collaboration

When SCP commands are invoked, SSH's authentication is activated and prompts a user for password or passphrase [38]. After successful authentication, the user command is placed in the payload field of a SSH data packet and that is sent under the control of SSH. The result of the command comes back in the same way as well. This process enables the user to proceed with normal copying file processes with additional security. The security attributes for the file copied can be preserved from the server system.

B.2.3 Overview Analysis on Protocol Functions

The SCP protocol fully provides three security functions: identification and authentication, timing functions, and trusted channel functions. It has also partially supported the information flow control functions by enabling preservation of default security attributes of the transferred files. However, the user can lose some features (due to the problem in attributes translation) if the file system mechanisms of hosts are different. If a supplementary security features (such as a fully functional access control mechanism and generating audit data) are available, then this protocol could be a better protocol for data transferring purposes.

B.3 Case study on FTAM

FTAM (File Transfer, Access, and Management protocol) is a specification for file transfer protocols that could be used between two hosts with incompatible file systems (e.g., Windows file system and Unix file system) [36].

B.3.1 General Functionality

FTAM provides file service mechanisms in relation to file transfer, access, and management of the file in an "Open System Interconnection" environment. However, these features are optional and an implementation design may choose only some of which according to their requirements. Such implementations could become incompatible with other implementations due to the selection of different services. This problem is overcome with the International Standardised Profile (ISO/IEC ISP 10607) by defining

some FTAM file services as mandatory. The main entities in FTAM services [14] are outlined in the following sections.

B.3.2 FTAM Virtual Filestore

The core feature of the FTAM file service is a virtual filestore (VFS). The VFS describes files and their attributes. The VFS makes a file understandable by the different real file systems of hosts. For this service, the local file system is required to map the VFS to the local filestore. The VFS stores a file with file attributes, data units (representing the contents of the file), and file structuring information.

The two parties involved in the FTAM transaction are called the initiator and the responder. The initiator is the controlling party that initiates an activity such as the file data transfer or file maintenance operations. The other party, the responder, performs an action in response to the request of the initiator. In other words, the initiator acts as a client and a responder as a server. The FTAM file service model is illustrated in Figure B.3.

B.3.3 File Attributes

FTAM classifies a set of attributes that describe the characteristics and contents of a file. Most of the file attributes are set for the file at the time of creation and only some of them can be modified later. All attributes may not relevant to the real filestore mechanism in the responder system. Attributes are therefore grouped based on the their characteristics in terms of an essential or optional requirement. For the initiator, the FTAM service provider must support all attribute groups in order to accept and supply values for all FTAM file attributes to the file received from a remote server.

B.3.4 Kernel Group

Attributes in this group are essential and must be supported. A FTAM responder stores these attributes and returns values to an initiator. The essential attributes composed in the kernel group are:

1. *File name*: There are no semantics specified by ISP for this attribute. It is used for mapping to the real file system, hence, it is a local issue.
2. *Permitted actions*: This attribute has two parts. One part identifies a set of actions,

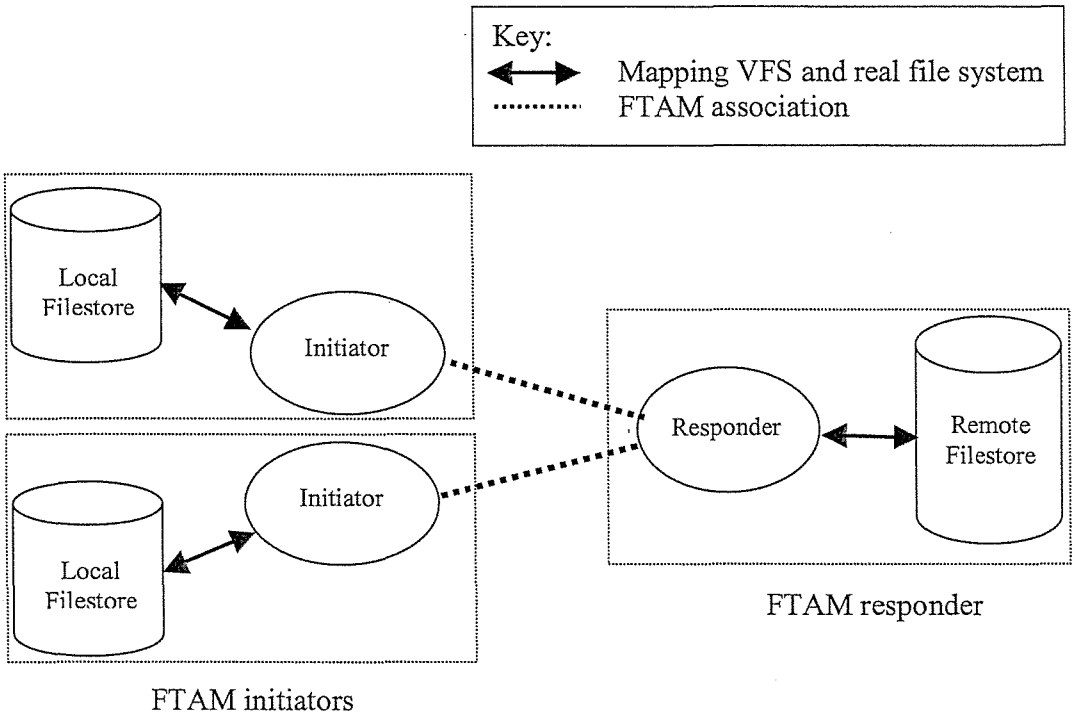

Figure B.3: FTAM File Service Model

e.g., read, and the other part specifies the set of FADU (File Access Data Unit) identity style, e.g., traversal. This unalterable attribute is set at the time of file creation.

3. *Contents type*: This attribute describes the file structure and the content data. This unchangeable attribute is set when the file is created.

B.3.5 Storage Group

This attribute group is optional for a responder. These options are:

1. *Storage account*: Identifies user accountable for charges regarding the storing file.

2. *History attributes*: Indicates the date and time of access to a file that includes creation, modification, and read access.

3. *File availability*: An indication made by initiators to access a file stored on a non-demountable device or a demountable device, the value of which is immediately available in the case of a non-demountable device, or, deferred available respectively.

4. *Filesize*: A file size given in bytes and set by the FTAM responder after an modifi-

cation or extension is made.

5.*Future filesize*: A maximum size to which a file can grow.

B.3.6 Security Group

This attributes group is also optional for a responder.

1. *Access control*: This attribute consists of a set of elements specifying one access condition (e.g., location of initiator, password requirement, etc.) in each element. Access is denied unless one match is found during checking against the list of access elements. This changeable attribute is set when the file is created.

2. *Legal qualification*: This attribute describes the legal status of the file in terms of data protection legislation. The FTAM specification does not provide detailed use of this attribute.

B.3.7 Private Group

This optional group contains a single attribute - private use. The FTAM specification makes no definition of this attribute.

B.3.8 Document Types

VFS defines a file using a flexible file model that provides predefined simple and common file types. In this model the attributes, structure, and contents of the file are described independently from the real filestore. A document type specifies rules for the file in terms of structure (named constraint set) and content (named abstract syntax). Four common document types defined in FTAM standard are:

- FTAM-1, an unstructured text file document type
- FTAM-2, a sequential text file document type
- FTAM-3, an unstructured binary file document type
- FTAM-4, a sequential binary file document type

Additional document types can be added if national registration authorities corroborate.

B.3.9 File Actions

File actions can be categorised into two classes as listed below.

1. Actions affecting the entire file: create file, select file, deselect file, read attributes, change attributes, open file, close file, delete file.
2. Actions affecting the individual FADUs: locate, read, insert, replace, extend, and erase.

B.3.10 Services and Protocol Functions

Activities and Regimes

The initiator forms an application association when a positive response, sent by a responder, is received. That indicates that an activity is started. During the protocol activity, states - known as regimes - are used to track the action permitted. There are four regimes defined as follows:

- FTAM regime - subsists while the application association is being presented.
- File selection regime - subsists while a named file is selected.
- File open regime - subsists while individual FADUs are in effect.
- Data transfer regime - subsists while a bulk of data transferring occurs.

There are a number of attributes that are associated with FTAM activities, but all are not valid for every regime. These attributes are logically grouped as follows:

- Kernel group - Active content type, current access request, current initiator identity, current location, current processing mode, current calling application entity title, current responding application entity title.
- Storage group - Current account, current concurrency control, current locking style.
- Security group - Current access passwords, active legal qualifications.

F-initialise functionality

This file service initiates an FTAM application association with a responder. Three

parameters, initiator identity, account, and filestore password, are required.

F-create functionality

This file service creates a new file and assigns file attributes. The associated parameters of this service are account, create password, and override. The latter one is required only if the file to be created is already existed.

B.3.11 Overview Analysis on Protocol Functions

FTAM can authenticate users and record the time of operations just like FTP. In addition, the flow of information between two parties is also controlled by FTAM using file access control attributes. The protocol itself can not generate audit data and it does not provide any mechanism to set up the secure channel between hosts.

B.4 Protocol Comparison and Analysis

The protocols comparison has conducted to see if a protocol has the security components identified in the security requirements profile for file downloading, which is explained in detail in Chapter 4. The protection profile addresses security issues under five functions as follows:

- Audit generation functions: mechanisms to generate audit log.
- Information flow control functions: mechanisms to impose limitations imposed by organisational policies.
- Identification restrictions functions: mechanisms to verify user identification.
- Timing functions: mechanisms to provide the time of users' activities to the audit generation functions.
- Trusted channel functions: mechanisms to establish a secure connection between hosts.

The HTTP/TLS collaborative protocol effectively offers 3 security features - user identification and authentication, and secure channel. It is predictable as the conventional functionalities of HTTP are to support any kind of data transferring and TLS to

Components	HTTP/TLS	FTP	SCP/SSH	FTAM
Audit generation functions	No	No	No	No
Information flow control functions	No	No	No	Yes
Identification restrictions functions	Yes	Yes	Yes	Yes
Timing functions	No	No	No	No
Trusted channel functions	Yes	No	No	No

Table B.1: Protocols and Security Requirements Relationship

provide a secure channel. The web-based protocol, HTTP over TLS (HTTPS), obviously allows users to perform data transferring under limited security.

FTP provides users with only one function, user identification restriction functions. It is a bottom-line security feature and is really fulfilled by the operating system. Thus, it appears that the FTP provides services that are very vulnerable security wise.

The SCP protocol, using SSH facilities, possesses two security measures. Like the other protocols discussed above, user identification and authentication (the basic security mechanism) are available. More options (e.g., password, RSA authentication, etc.,) are implemented in order to support users with many levels of security. The security channel would be attained as SSH uses its administratively assigned port number (default 22) with data encryption/decryption.

The analysis of FTAM protocol yields two security benefits for users. They are user identification restrictions function and information flow control function. There is no service in FTAM to facilitate a secure trusted channel within its hosts, but FTAM provides the security at other Open System Interconnection (OSI) layers (3, 4, 0r 7, as appropriate).

The table B.1 illustrates the protocols and their supported security requirements of the PP.

According to the table above, some of the security functional requirements that we consider critical are not provided by the protocols utilised in current file downloading applications. Note that none of them provides for the required audit data generation function.

Bibliography

[1] Harald T. Alvestrand. MIME media types. http://www.alvestrand.no/ietf/media-types.html, July, 2002.

[2] L. Ambuel and M. Donaldson. The Process of Building Protection Profiles, NISSC Workshop. http://www.securitytechnet.com/resource/crypto/evaluation/cc/pp_wrkshop98.pdf, October 1998.

[3] Inc. Apple Computer. QuickTime Documentation. http://developer.apple.com/documentation/quicktime/quicktime.html, July 2003.

[4] D. Bell and L. LaPadula. Secure computer systems: Mathematical foundations. Technical Report MTR-2547, Vol 1, MITRE Corp., Bedford, MA , Nov 1973.

[5] T. Berners-Lee, R. Fielding, and H. Frystyk. Hypertext Transfer Protocol (HTTP/1.0). http://www.ietf.org/rfc/rfc2068.txt, May, 1996.

[6] T. Berners-Lee, L. Masinter, and M. McCahill. RFC 1738 Uniform Resource locators (url). http://www.ietf.org/rfc/rfc1738.txt, December, 1994.

[7] Abhay Bhushan, Bob Braden, Will Crowther, and Alex McKenzie. The data transfer protocol. http://www.ietf.org/rfc/rfc0171.txt, June, 1971.

[8] Inc. C2Net Software. Stronghold Web Server 2.4 Administration Guide. http://stronghold.redhat.com/support/docs-2.4/admin/HTML/crypto.html, September, 2002.

[9] W. Caelli, D. Longley, and M. Shain. *Information Security Handbook.* Macmillan, 1991.

[10] Cert. Vulnerability Note VU:443699 Microsoft Internet Explorer Does Not Respect Content-Disposition and Content-Type MIME Headers. `http://www.kb.cert.org/vuls/id/443699`, December, 2001.

[11] Herb Chong. Internet Security, ActiveX and Java. `http://www.windowatch.com/security.html`, May 2001.

[12] Douglas E. Comer. *Internetworking with TCP/IP volume 1.* Prentice Hall, 2000.

[13] Common Criteria. Arrangement on the Recognition of Common Criteria Certificates in the field of Information Technology Security. `http://www.commoncriteria.org/registry/ccra-final.html`, May, 2000.

[14] IBM Corp. FTAM User's Guide. `http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/EHXB5000/CCONTENTS`, December, 1994.

[15] Netscape Communications Corporation. Plug-in guide. `http://developer.netscape.com/docs/manuals/communicator/plugin/index.htm`, January, 1998.

[16] D. Dean, E. Felten, and D. Wallach. Java Security: From Hotjava to Netscape and Beyond. Proceedings of 1996 IEEE Symposium on Security and Privacy (Oakland, California), May, 1996.

[17] T. Dierks and C. Allen. The TLS Protocol Version 1.0. `http://www.ietf.org/rfc/rfc2246.txt`, January, 1999.

[18] M. G. Donaldson. Guide for the production of protection profiles and security targets. `http://csrc.ncsl.nist.gov/cc/t4/wg3/`, April, 2000.

[19] ISO Easy. International Standards for Quality Management Systems. `http://www.isoeasy.org/`, April 2003.

[20] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. Hypertext Transfer Protocol (HTTP/1.1). http://www.ietf.org/rfc/rfc2068.txt?number=2068, January, 1997.

[21] N. Freed and N. Borenstein. Multipurpose Internet Mail Extensions. http://www.ietf.org/rfc/rfc2045.txt, November, 1996.

[22] Alan O. Freier, Philip Karlton, and Paul C. Kocher. The SSL Protocol Version 3.0 Internet Draft. http://home.netscape.com/eng/ssl3/ssl-toc.html, March, 1996.

[23] A. Frisch. *Essential System Administration*. O'Reilly & Associates, Inc., Sebastopol, CA, August, 2002.

[24] George Ryan, Ella Miller, J.J. Rocha, Booz Allen, Neal Ziring, and Charles Lavine. Web Browser Protection Profile. http://niap.nist.gov/cc-scheme/PP_WBPP_V0.5.html, April, 2001.

[25] I. Goldberg, D. Wagner, R. Thomas, and E. A. Brewer. A Secure Environment for Untrusted Helper Applications. In Proceedings of the sixth USENIX Security Symposium, July 1996.

[26] U.S. Government and industry. Web Server Protection Profile. http://niap.nist.gov/cc-scheme/PP_WSPP_V0.6.html, July 2001.

[27] Gregory R. Gromov. History of Internet and WWW: The Roads and Crossroads of Internet History. http://www.netvalley.com/intval.html, January, 2002.

[28] Information Security Group. International Recognition for Australia and New Zealand IT Security Evaluations. http://www.dsd.gov.au/infosec/services/press.html, June, 2002.

[29] Webopedia INT Media Group. Extranet. http://www.webopedia.com/TERM/e/extranet.html, June, 2002.

[30] Webopedia INT Media Group. Internet. http://www.webopedia.com/TERM/i/intranet.html, June, 2002.

[31] Ken Hornstein. Kerberos FAQ. http://www.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html, August, 2000.

[32] Craig Horrocks and Averill Parkinson. Cookies violate assumption web surfing is anonymous. http://www.clendons.co.nz/library/articles/cookies.htm, October 1997.

[33] Adobe Systems Incorporated. Acrobat Family. http://www.adobe.com/products/acrobat/main.html, July 2003.

[34] Australian Communications-Electronic Security Instruction 33 (ACSI 33) Information Security Group. Handbook 10 Web Security Version 1.0. http://www.dsd.gov.au/infosec/acsi33/acsi_index.html, December 2000.

[35] Australian Communications-Electronic Security Instruction 33 (ACSI 33) Information Security Group. Handbook 3 Risk Management Version 1.0. http://www.dsd.gov.au/infosec/acsi33/HB3.html, December 2000.

[36] ISO. File Transfer, Access and Management Part 1: General introduction. http://www.iso.ch/iso/en/StandardsQueryFormHandler.StandardsQueryFormHandler, September 1995.

[37] James Q. Jacobs. Multimedia Plug In. http://www.jqjacobs.net/web/plug-ins.html, June, 2002.

[38] Daniel J.Barrett and Richard E. Silverman. *SSH The Secure Shell The Definitive Guide*. O'Reilly & Associates, Inc., Sebastopol, CA, February, 2001.

[39] Brian Kantor and Phil Lapsley. Network News Transfer Protocol. http://www.ietf.org/rfc/rfc977.txt, February, 1986.

[40] Ed Kubaitis. WWW Browser Security and Privacy Flaws. http://www.cen.uiuc.edu/~ejk/browser-security.html, July, 2002.

[41] Thorsten Kukuk. The Linux NIS(YP)/NYS/NIS+ HOWTO. http://en.tldp.org/HOWTO/NIS-HOWTO/, August, 2002.

[42] Inc. Macromedia. Macromedia. http://www.macromedia.com, July 2003.

[43] S. Mcleod and Dr M. Cohen. SSL Vulnerabilities. Auscert Asia Pacific Information Technology Security Conference. Gold Coast, Australia, May, 2002.

[44] Last Stage of Research Group. Java and Java Virtual Machine Security Vulnerabilities and their Exploitation Techniques. http://lsd-pl.net/java_security.html, October 2002.

[45] National Institute of Standards and Technology. Bulletin, Common Criteria: Launching the International Standard. http://csrc.nist.gov/publications/nistbul/index.html, 1998.

[46] National Institute of Standards and Technology. Common Criteria Version 2.1/ISO 15408. http://csrc.ncsl.nist.gov/cc/ccv20/ccv2list.htm, August 1999.

[47] Rolf Oppliger. *Internet and Intranet Security*. Artech House Inc., 1998.

[48] J. Postel and J. Reynolds. File Transfer Protocol (FTP). http://www.ietf.org/rfc/rfc959.txt, October, 1985.

[49] Microsoft Press. ITSEC Rating Confirms Security of Windows NT 4.0. http://www.microsoft.com/PressPass/press/1999/Apr99/UKSecurPR.asp, 1999.

[50] Inc. RealNetworks. 1999 Press Releases. http://www.realnetworks.com/company/press/releases/1999/ibm_rn.html, July 2003.

[51] E. Rescorla. HTTP Over TLS. http://www.ietf.org/rfc/rfc2818.txt, May, 2000.

[52] Computer Security and Industrial Cryptography. A Secure European System for Applications in a Multi-vendor Environment. https://www.cosic.esat.kuleuven.ac.be/sesame/html/sesame_what.html, June 2000.

[53] Kurt Seifried. Log files and other forms of monitoring. http://www.linuxpowered.com/archive/guides/securityguide/lasg-www/logging/.

[54] Karanjit S. Siyan. *Inside TCP/IP*. New Riders, third edition edition, 1997.

[55] Sotiris Ioannidis and Steven M. Bellovin. Building a Secure Web Browser. In Proceedings of the USENIX 2001 Annual Technical Conference, Freenix Track, http://www.cis.upenn.edu/~sotiris/thevillage/6private/pub.html, June, 2001.

[56] Sotiris Ioannidis, Steven M. Bellovin, and Jonathan M. Smith. Sub-Operating System: A New Approach to Application Security. In Proceedings of the ACM SIGOPS European Workshop, France. http://www.cis.upenn.edu/~dsl/STRONGMAN/, September 2002.

[57] L. Stein and J. Stewart. The World Wide Web Security FAQ. http://www.w3.org/Security/Faq/wwwsf2.html, July, 2001.

[58] G. Stoneburner. CSPP - Guidance for COTS Security Protection Profiles. http://csrc.ncsl.nist.gov/cc/pp/pplist.htm, December 1999.

[59] Sun Microsystems. Applets. http://java.sun.com/applets/index.html, May, 01.

[60] Sun Microsystems. What is JavaTM Technology? http://java.sun.com/java2/whatis/, October, 2002.

[61] National Technical Systems. Software Functionality Testing. http://www.ntscorp.com/scripts/test/test20.html, June 2003.

[62] Australian Computer Emergency Response Team. CERT Advisory CA-97.20 - Javascript Vulnerability. http://www.stanford.edu/group/itss-ccs/security/Advisories/97-1312.html, February, 2000.

[63] Australian Computer Emergency Response Team. Distributed Denial of Service Attacks. http://www.auscert.org.au/Information/Auscert-info/Papers/ddos.html, February, 2000.

[64] Microsoft TechNet. Microsoft Security Bulletin MS03-008, Flaw in Windows Script Engine Could Allow Code Execution (814078). http://www-no.ucsd.edu/security/ms-os.html, July 2003.

[65] Microsoft TechNet. What is JScript? http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/js56jsconabout.asp, July 2003.

[66] Microsoft TechNet. Microsoft Security Bulletin MS01-027, Flaws in Web Server Certificate Validation Could Enable Spoofing. http://www.microsoft.com/technet/treeview/default.asp?url=/technet/security/bulletin/MS01-027.asp, June 2003.

[67] The SANS Institute. Introduction to IP Spoofing. http://www.sans.org/rr/threats/intro_spoofing.php, November, 2000.

[68] Rick Thomas. Denial of Service - Hackers Attack. http://www.thepbj.com/021800/a16.htm, September, 2000.

[69] Vinod Anupam and Alain Mayer. Security of Web Browser Scripting Languages: Vulnerabilities, Attacks, and Remedies. 9th USENIX Security Symposium. http://citeseer.nj.nec.com/anupam98security.html, 2000.

[70] E. Z. Ye, Y. Yuan, and S. w. Smith. Web Spoofing Revisited: SSL and Beyond. Computer Science Technical Report TR2001-417. http://www.cs.dartmouth.edu/~pkilab/papers/tr417.pdf, February, 2002.