SECURITY ASPECTS OF A JAVA-SERVLET-BASED WEB-HOSTED E-MAIL SYSTEM

Eleanor Hepworth and Ulrich Ultes-Nitsche*

Department of Electronics and Computer Science

University of Southampton

Southampton, SO17 1BJ, United Kingdom

{emh195,un}@ecs.soton.ac.uk

Abstract The project WEBMail addresses the problem of e-mail access from a workstation different from a user's local machine. Usually this scenario requires setting up an e-mail system with particular server information. WEBMail aims to reduce this effort by hosting the e-mail system on a web-page that can be accessed from all over the world, requiring only the presence of a web-browser on the local machine. World-wide accessibility makes this type of e-mail system particularly vulnerable and, consequently, calls for a very thorough consideration of security aspects related to its Internet-accessibility. We present in this paper the architectural model of the WEBMail e-mail system, concentrating on its security features.

Keywords: Internet Security, Security in Java, Java Servlets, E-Mail, World-Wide Web.

1. INTRODUCTION

Most of us know the situation very well where, while being abroad, reading e-mail at home causes a fair amount of hassle, e.g. requiring the configuration of an e-mail system at a foreign institution to access your e-mail account. Quite often such temporary configurations are not worth the effort compared to a rather short time spent away. It is the aim of

*Former name: Ulrich Nitsche.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: [10.1007/978-0-387-35575-7_19](https://doi.org/10.1007/978-0-387-35575-7_19)

J. H. P. Eloff et al. (eds.), *Information Security Management & Small Systems Security*
© IFIP International Federation for Information Processing 1999

the WEBMail-project to overcome this problem by hosting the e-mail system on a web-page rather than on your local machine. So one will have the same access to e-mail using the same user interface and settings from all over the world. Obviously this requires particular attention to be spent on security issues related with the Internet.

The key idea of WEBMail is to access e-mail from a web-server that is equipped with additional functionality to handle e-mail manipulation by means of standard e-mail-protocols. The system provides a web-interface and all security-related functionality ensuring privacy, authenticity and integrity. The new system must not give rise to alteration of e-mails and mail-spoofing as it must not infringe the privacy of e-mails received and stored. In addition, it must not reveal any information such as passwords to non-authenticated servers, protecting against various threats.

WEBMail is written as a Java Servlet, i.e. a Java program that runs on the server-side, that adds functionality to a Java-enabled web-server. It communicates with the user via an HTML-interface that the servlet writes to and reads from. WEBMail has an object-oriented design and supports different e-mail protocols such as POP3 [9] and IMAP4 [2] to access the user's inbox and SMTP [11] to send messages. To ensure security, communication with the user is protected using the SSL (secure sockets layer) protocol and security features of Java servlets.

We present in this paper a brief architectural discussion of WEBMail, concentrating on security aspects and their implementation. In Section 2. we give a short introduction to Java servlets. Section 3. contains a description of the SSL protocol. Section 4. comprises the architecture of WEBMail. In Section 5. we discuss how the security aspects have been addressed in WEBMail effectively. The paper ends with Section 6. where we give concluding remarks on WEBMail.

2. JAVA SERVLETS

The past year has seen the rise of server-side Java applications, known as Java servlets. Servlets are used to add increased functionality to Java-enabled servers in the form of small, pluggable extensions. When used in extending web servers, servlets provide a powerful and efficient replacement for CGI (Common Gateway Interface) and offer many significant advantages [13]. These advantages include:

2.1 PORTABILITY

Java servlets are protocol and platform independent and as such are highly portable across platforms and between servers. The servlets must

conform to the well defined Java servlet API which is already widely supported by many web servers.

2.2 PERFORMANCE

Java servlets have a more efficient life cycle than the either CGI or FastCGI scripts. Unlike CGI scripts, servlets do not create a new process for each incoming request. Instead, servlets are handled as separate threads within the server. At initialisation, a single object instance of the servlet is created that is generally persistent and resides in the servers memory. This persistence reduces the object creation overhead. There are significant performance improvements over CGI scripts in that there is no need to spawn a new process or invoke an interpreter [7]. The number of users able to use the system is also increased because fewer server resources are used for each user request.

2.3 SECURITY

The Java language and Java servlets have improved security over traditional CGI scripts both at the language level and at the architecture level:

Language Safety. As a language Java is type safe and handles all data types in their native format. With CGI scripts most values are treated and handled as strings which can leave the system vulnerable. For example, by putting certain character sequences in a string and passing it to a Perl script, the interpreter can be tricked into executing arbitrary and malicious commands on the server.

Java has built-in bounds checking on data types such as arrays and strings. This prevents potential hackers from crashing the program, or even the server, by overfilling buffers, in case of not properly written code. For example, this can occur with CGI scripts written in C where user input is written into a character buffer of a predetermined size. If the number of input characters is larger than the size of the buffer, it causes a buffer overflow and the program will crash. This is commonly known as stack smashing.

Java has also eliminated pointers and has an automatic garbage collection mechanism which reduces the problems associated with memory leaks and floating pointers. The absence of pointers removes the threat of attacks on the system where accesses and modifications are made to areas of server memory not belonging to the service process.

Finally, Java has a sophisticated exception handling mechanism, so unexpected data values will not cause the program to misbehave and

crash the server. Instead an exception is generated which is handled by the program that usually terminates neatly with a run time error [6].

Security Architecture. Java servlets have been designed with internet security issues in mind and mechanisms for controlling the environment in which the servlet will run have been provided.

CGI scripts generally have fairly free access to the server's resources and badly written scripts can be a security risk. CGI scripts can compromise the security of a server by either leaking information about the host system that can be used in an attack, or by executing commands using untrusted or unchecked user arguments. Java significantly reduces these problems by providing a mechanism to restrict and monitor servlet activity. This is known as the servlet sandbox. The servlet sandbox provides a controlled environment in which the servlet can run and uses a security manager to monitor servlet activity and prevent unauthorised operations. There are four modes of operation that include trusted servlets, where the servlet has full access to the server resources, and untrusted servlets which have limited access to the system.

JDK 1.2 contains an extension to its security manager, the access controller. The idea behind the access controller is to allow more fine-grained control over the resources a servlet can access. For example, instead of allowing a servlet to have write permission to all files in the system, write permission can be granted for only the files required by the servlet for execution [7].

However, Java-based servers are still vulnerable to denial of service attacks where the system is bombarded with requests in order to overload the server resources. This approach invokes so many servlet instances that all the server resources are allocated. This can impact all the services supported by the server. However, the effects of this can be reduced by specifying an upper limit on the number of threads that can be run concurrently on the server. If all the threads are allocated, that particular service can no longer be accessed, but because the server still has resources left to allocate, the rest of the services are still available.

3. SECURE SOCKETS LAYER PROTOCOL

The secure sockets layer protocol (SSL) is designed to establish transport layer security with respect to the TCP/IP protocol stack. Version 3 was published as an Internet draft document [5] by the IETF (Internet Engineering Task Force). We introduce SSL briefly along the lines of [12] and motivate its usage in WEBMail.

3.1 THE PROTOCOL STACK

The transport layer part of SSL, the SSL record protocol, sits on top of TCP in the Internet protocol stack. It is accessed by an upper layer consisting of the hypertext transfer protocol (http) and different parts contributing to SSL: SSL handshake protocol, SSL change cipher spec protocol, and the SSL alert protocol, used to set up, negotiate, and change particular security settings used by the SSL record protocol. Schematically, the SSL architecture is presented in Figure 1.1.

Figure 1.1 SSL within the Internet protocol stack [12].

3.2 SECURITY FEATURES OF SSL

SSL allows for different security features being chosen. First of all, different encryption algorithms can be used to produce ciphertexts and authentication messages. For authentication, different hash algorithms can be negotiated. SSL can also use X509.v3 peer certification [6]. Using a session identifier, active states of SSL are identified, where a state consists of a number of keys involved in the session, both on the server and on the client side, and sequence numbers to count the messages exchanged. By using these different parameters, SSL sets up a session configuration that then allows for ensuring integrity, confidentiality, and authentication depending on the set up parameters.

3.3 USE OF SSL IN WEBMAIL

Unlike other concepts that secure connections or even only data-packages, SSL includes the concept of a secure session, determined by the parameters mentioned in the subsection above. It is this session con-

cept that makes it appealing for being used in WEBMail. The secure session last as long as you access the e-mail system, starting when you identify yourself to the system. Since communication with the user is based on html-documents sent to her or him using http that sits on top of the SSL record protocol in the Internet protocol stack, the use of SSL is transparent to the client/server-connection via http.

4. THE ARCHITECTURE OF WEBMAIL

A proposed design for the WEBMail system has been developed using the Unified Modelling Language (UML). Figure 1.2 gives an overview of the basic system architecture.

Figure 1.2 Overview of System Architecture.

The remote domain will be accessed via the Web server, which can be connected to from most external domains. The connection between the users machine and the Web server will use HTTP and the Secure Socket Layer(SSL). Together these are the HTTPS protocol. SSL will be used to maintain the data integrity. A servlet running on the Web server will communicate with the mail servers using the e-mail protocols such as POP3 [9] and IMAP4 [2] to access the user inbox, or SMTP [11] to send messages.

The system has been designed to reduce dependencies between the servlet and the associated protocols. The servlet accesses the protocols via methods defined in a specified interface that each protocol must conform to. This allows the servlet to use the protocols with no knowledge of the underlying protocol and it's implementation. The system has been designed this way to allow the modular addition of new protocols to the system.

As shown in Figure 1.3 the system is split into two packages: the WEBMail package and the Protocol package. The dependency between these two packages has been kept to a minimum so only changes to the interfaces defined in the protocol package will require modifications in the WEBMail package.

Figure 1.3 The Package Diagram

As shown in Figure 1.4, the WEBMail package contains the WEBMail Servlet and the Registry class.

Figure 1.4 The WEBMail Package Class Diagram

4.1 WEBMAIL SERVLET

The WEBMail Servlet is responsible for interacting with the user and interpreting user data and commands. It also controls user authentication and session tracking, and checks session validity before executing user commands. The servlet also interacts with the e-mail protocol servers and issues commands to execute user instructions and interprets returned data.

4.2 REGISTRY

The Registry is responsible for tracking the protocols currently in use by the system. At initialisation, the Registry reads a configuration file that specifies the protocols to use and their associated servers. The

Registry object is then responsible for creating the appropriate protocol object when requested to do so by the servlet. The servlet utilises this object using the known methods of the interface.

The Registry configuration file contains an ordered list of all the servers that could be used to send or read messages. The Registry usually returns a reference for the first server in that list, but if that server is unavailable there is a list of backup servers that could be used instead. The configuration file is also dynamically configurable allowing on-the-fly addition of new servers, or alterations to the list order.

Figure 1.5 The Protocol Package Class Diagram

Figure 1.5 shows the Protocol package class diagram. The Protocol package contains the following classes:

4.3 MESSAGE

This class encapsulates message information in a format that the servlet understands. The Message class has various attributes which summarise certain fields of the message header for easy access by the servlet.

4.4 SERVER

This class encapsulates information about the protocol servers in the system.

4.5 GENERIC PROTOCOL

This is an abstract class which contains attributes and methods common to all protocols that may be used by WEBMail. These methods include `connect`, which sets up a connection to the relevant protocol server, and `quit`, which closes the connection. The `status` attribute is used to track the current state of the protocol server.

4.6 PROTOCOL INTERFACES

There are three interfaces defined in this diagram, the Receiver Interface, the HierarchicalReceiver Interface and the Sender Interface. The Receiver Interface is for protocols that read data from a server, but do not have support for directory trees, such as POP3. The HierarchicalReceiver Interface is for protocols that read data from a server and support directory structures, such as IMAP4. The sender interface is used for protocols that send data to a server for posting to a destination, such as SMTP. These example protocols are shown on the class diagram to provide an idea of how the Interfaces are used. They are also the initial set of protocols implemented by WEBMail.

To add a new service to the system, the protocol must be implemented so that it conforms to one or more of the interfaces. For example, if support for the NNTP protocol were to be built into the system, it would implement both the Sender and the HierarchicalReceiver interfaces.

5. THE SECURITY OF WEBMAIL

We discuss in this section the security features of WEBMail. We do so by focussing on the use of SSL in WEBMail. In order to explain why we chose SSL for authentication purposes, we firstly introduce two other authentication approaches existing for the Internet: HTTP authentication and digest authentication.

5.1 HTTP AUTHENTICATION

HTTP authentication is a simple challenge/response protocol for the authentication of users. It is used to prevent unauthorised access to resources on an HTTP server. This basic authentication mechanism was originally defined in HTTP 1.0[1] and is based on the assumption that the connection between the client and the server can be trusted. As a result of this assumption, the username and password are passed over the network as clear text, which means that HTTP Authentication is unsuitable for applications that require anything other than weak, non-secure authentication.

5.2 DIGEST AUTHENTICATION

Digest Authentication[4] is a variation on the on the basic authentication scheme intended as a replacement for HTTP Authentication. The password is transmitted across the network as a digest, which is generated by taking the MD5 hash of the username, password, HTTP request method, the requested URI (Universal Resource Identifier) and a one time pad, a *nonce*, generated by the server. The transmitted digest is compared with the digest generated by the server, and if they match, access is granted. This mechanism is more secure than HTTP authentication, but it still only provides weak user authentication, and is not suitable for applications that require secure data transfer as well.

5.3 THE BENEFITS OF USING SSL

As mentioned above, HTTP authentication and digest authentication only provide a basic authentication mechanism, lacking mechanisms for confidentiality and integrity. In contrast, SSL offers the full range of security mechanisms needed to establish a secure session. It is particularly the secure session concept that makes SSL appealing to the considered application area. As soon as an SSL session is established, it is completely transparent to http, preserving http as a stateless protocol. The WEBMail servlet also supports session tracking which makes it even more attractive to choose a session-oriented security mechanism rather than anything else.

HTTP authentication or digest authentication would establish a suitable mechanism for security in WEBMail only if TCP/IP were secured. This could be achieved by running IPsec or IPv6 (IP version 6 is the latest IP version and includes IPsec) [6] rather than the basic Internet protocol (IPv4). IPsec is a low layer protocol in the Internet protocol stack for end-to-end confidentiality of the transmission of IP-packages. However, SSL is a security protocol widely deployed such that its choice appears to be more practical than choosing any combination of IPsec/IPv6 with HTTP authentication/digest authentication. That is, finally, why we have chosen SSL (version 3) for WEBMail.

5.4 EXAMPLE USE CASE: THE LOGIN-PROCEDURE OF WEBMAIL

The WEBMail system provides a simple but secure challenge/response mechanism for authenticating users. The user enters their username and password into a login form and the data is sent to the server via a secure SSL connection. The servlet uses this information to authenticate the

user by invoking the `authenticate` method. The WEBMail system can be configured to authenticate users from a local file, or by connecting to a server specified in the Registry's configuration file. If the `username/password` pair is valid, the server starts a session to track requests from the same client and sends a positive response back to the user. This sequence of events can be seen in Figure 1.6.

Figure 1.6 User Login and Authentication

6. CONCLUSIONS AND FURTHER WORK

We have presented in this paper an e-mail system that is hosted on a Web-page, focussing on the security problems arising in such a system. Besides the practicality of the system itself, the security issues addressed are of a much wider interest than only to the presented application area. Internet security is discussed in the framework of Java applications running on the server: *Java servlets* [7]. By this, the problems and solutions discussed in this paper are relevant to security aspect of all major Internet-based application, in particular to applications in electronic commerce.

As pointed out earlier, we consider Java servlets as more applicable than CGI-scripts and other scripting languages when it comes to security. Java servlets have a built in support for digital certificates, including a built in support for the secure sockets layer protocol. Comprising SSL (secure sockets layer protocol), we believe WEBMail to be as secure a system as other e-mail systems available on the market that are hosted

on local machines. WEBMail is implemented in such a way that any browser can be used to access it on the client side.

As mentioned above what we have learned from implementing WEBMail has a much wider focus than only e-mail applications. For instance in the project MobiMed [3] that addresses secure IT-support for clinical trials [10] an extension to distributed clinical trials is addressed currently. Java servlets, including the discussed security features, as applied to WEBMail will thoroughly be taken into consideration for the extension of MobiMed.

References

- [1] T. Berners-Lee, R. Fielding, and H. Frystyk *Hypertext Transfer Protocol - HTTP/1.0*, Internet Standards Request For Comments number 1945, May 1996
- [2] M. Crispin *Internet Message Access Protocol - Version 4rev1*, Internet Standards Request For Comments number 2060, December 1996
- [3] H.-R. Fischer, S. Teufel, Ch. Muggli, and M. Bichsel. *Privacy and Efficiency of Mobile Medical Systems (MobiMed)*. Case of Support, Swiss National Science Foundation SPP-IuK, Module: Demonstrator, No. 5003-045359, 1995. See <http://www.ifi.unizh.ch/ikm/MOBIMED/>.
- [4] J. Franks, P. Hallam-Baker, J. Hostetler, P. Leach, A. Luotonen, E. Sink, and L. Stewart *An Extension to HTTP : Digest Access Authentication*, Internet Standards Request For Comments number 2069, January 1997
- [5] A. O. Freier, P. Karlton, and P. C. Kocher. *The SSL Protocol Version 3.0*. Internet Draft. Netscape, Transport Layer Security Working Group, <http://home.netscape.com/eng/ssl3/draft302.txt>, November 1996.
- [6] S. Garfinkel and G. Spafford *Web Security and Commerce*, O'Reilly and Associates 1997.
- [7] J. Hunter *Java Servlet Programming*, O'Reilly and Associates, 1998.

- [8] B. Kantor and P. Lapsley *Network News Transfer Protocol*, Internet Standards Request For Comments number 977, February 1986
- [9] J. Myers and M. Rose *Post Office Protocol - Version 3*, Internet Standards Request For Comments number 1939, May 1996
- [10] U. Nitsche, R. Holbein, O. Morger, and S. Teufel. Realization of a Context-Dependent Access Control Mechanisms on a Commercial Platform. In *Proceedings of the IFIP TC11 14th International Conference on Information Security (IFIP/SEC)*. Vienna, 1998.
- [11] J. Postel *Simple Mail Transfer Protocol*, Internet Standards Request For Comments number 821, August 1982
- [12] W. Stallings. *Cryptography and Network Security*. Prentice Hall, New York, second edition, 1998.
- [13] Sun Microsystems. *Java Servlet API Whitepaper*, 1998.