Seamless Computing with WebSubmit

Ryan P. McCormack, John E. Koontz, Judith Devaney
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899

December 23, 1998

Abstract

WebSubmit is a Web-browser based interface to heterogeneous collections of remote high-performance computing resources. It makes these resources easier to use by replacing a constantly changing range of unfamiliar, command-driven queuing systems and application environments with a single, seamless user interface. WebSubmit lets users run in their regular accounts on the remote system. Strong authentication using the Secure Sockets Layer protocol allows registered users connect to the WebSubmit authority. When validated by the authority they gain access to a group of application modules. Each application module is presented as an HTML form; this form is filled out and submitted to the server, which then processes the request and executes the desired tasks on the specified remote system using the Secure Shell protocol. The system is flexible and extensible, and its modularity promotes ease of use, maintainability, and interface development.

Keywords: high-performance computing, CGI applications, user interfaces, computer security, remote login, Tcl/Tk

1 Introduction

Effective use of high-performance computing (HPC) systems can be a daunting task. Users must deal with an array of constantly changing hardware fronted by diverse operating systems. The tools for similar tasks – for example, job queuing – can vary from system to system, even when the operating systems (UNIX variants) are nominally the same. The barriers to using HPC systems can be overcome by providing users with a single, easy-to-use interface that insulates them from direct contact with the operating systems, tools, and applications on the HPC systems. **WebSubmit**,[1, 2, 3] a browser-based gateway to remote applications, is one way to do this. It provides a friendly, system-neutral environment in which trusted users can access application software on HPC systems.

Trusted is the key word here. The Web is used mainly to transmit documents and images; the introduction of Java has made it also a way to distribute client-side executables safely. Apart from this, programs can be run on remote Web server systems using the Common Gateway Interface (CGI). CGI programming tasks have been restricted to those that can be accomplished anonymously. They are executed as the server user (e.g., `nobody`) and have only the limited privileges of this account. In most cases, this is appropriate: allowing random users to execute any command on the server system would be giving away the keys to the store. Still, it would be nice to be able to identify valid, trusted users and give them the same privileges they would get with a regular login.

The contribution of WebSubmit is that it provides a framework for establishing just this sort of trust relationship in a CGI environment. In this way WebSubmit adds a telnet-like functionality to the ftp-like functionality of the existing Web. The client side execution facility of Java is supplemented by a remote execution facility that can run user-owned jobs on existing, unmodified legacy systems, including HPC systems, the application discussed here. The familiar, pleasant user interface used in Web browsing is extended from document retrieval to remote execution.

The primary goal of the WebSubmit project is to provide users with seamless access to a collection of HPC resources. The ideal has been to create an environment in which, from the user's viewpoint, the distributed nature and heterogeneity of the resources disappear. WebSubmit is not intended to be a distributed computing system, although it is extensible in that direction. In this sense, the scope of WebSubmit is not as large as that of projects like Globus [4] or Legion,[5] which create and provide access to a distributed computer. WebSubmit is more similar to UNICORE,[6] as both projects seek to provide simplified access to existing HPC systems. Both UNICORE and WebSubmit utilize the World Wide Web as the interface to their systems. Both address security similarly through SSL. They differ in the method of implementation and scope. WebSubmit is based on CGI and Tcl, whereas UNICORE uses Java. The scope of UNICORE is also larger, in that it is intended for interdependent tasks targeted at multiple geographically distributed sites. WebSubmit is intended to simplify access to software and HPC systems at a single site. WebSubmit allows both batch and interactive use of the machines that it interfaces; it has an interactive module that enables the user to submit commands to any of the included computers as if they were logged on. UNICORE is intended for batch access only.[7]

At present WebSubmit provides access to batch queues and to a range of interactive utilities including file editing and file transfer on several HPC systems at the National Institute for Standards and Technology (NIST). The currently supported systems are an IBM SP2 running LoadLeveler,[8] two SGI Origins 2000s running SGI's NQS,[9] and a Linux-based Pentium array running LSF.[10] We hope it will be obvious that WebSubmit is not limited to these HPC systems, or to HPC applications generally, or, indeed, to any particular kind of application at all.

In this paper we will illustrate the basic structure of WebSubmit, and describe the applications we are supporting with it. We will pay particular attention to the security framework used to provide the strong authentication we rely on. We will also address some of the policy issues faced by users and administrators, and outline some future directions for the software and project.

2 An Overview of WebSubmit

WebSubmit operates over a set of networked systems, linked by common Web and Internet protocols in a simple transaction model. The user interface, which appears in the user's browser, is composed of a group of application modules, each of which is implemented by a pair of CGI scripts. These CGI scripts reference some shared library code. The software is modular, flexible and extensible, with hooks for including existing CGI code, and for developing and adding new applications. The code is portable and can be modified to suit the needs of a given site quickly and easily.

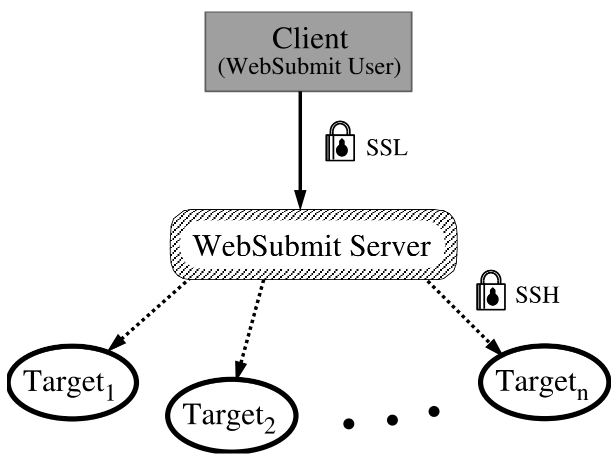

Figure 1: The basic WebSubmit transaction model. A client contacts the WebSubmit server using a Web browser, which then forwards requests to any number of target computing systems.

2.1 The WebSubmit Transaction Model

The WebSubmit user interface is set up to help users accomplish specific tasks on one or more HPC (or other) systems. Each of these tasks is accomplished within a basic transaction model with three parties (see Figure 1):

1. **Clients:** the systems of the users requesting performance of tasks

2. **WebSubmit Server:** the system that does user authentication and formats and routes client task requests

3. **Targets:** the systems on which the tasks are performed

The WebSubmit server is configured to interact with a specific group of one or more target systems (hereafter referred to as the WebSubmit *cluster*) specified by the WebSubmit administrator. For security reasons, a particular WebSubmit server can interact only with systems within its configured cluster.

The user uses a Web browser on the client system to obtain a secure connection with the WebSubmit server's master page, then follows a link on that page to the *application module* (see Section 2.3) page for the task of interest. The module page is the user interface to a task. It is an HTML form that the user fills out and submits to the WebSubmit server. The form is generated by the module's form generator CGI script, which is designed to minimize error checking by precluding entry of incorrect values where this is possible. Modules can be in generic format, requiring the user to specify the target system in the form, or in specific format, restricted to a particular target.

The WebSubmit server processes the submitted form with the module's form processor CGI script, which performs any target-side error checking of the input data, and executes the specified task on the proper target system. Execution may consist of submitting a job to the job queue on the target, or of running a command script. Output from whichever is the case is then returned to the user's browser for viewing. This process is detailed in Figure 2.

If the task is a job queue submission, the output returned is that produced by the act of submitting the job, not the final output of the job itself. As we have formulated our interface, it is up to the user to keep track of the progress of the job and to retrieve the final output and direct it to subsequent jobs.

2.2 Cluster System Requirements

The client system can have any kind of operating system that has a browser with Secure Sockets Layer (SSL), HTML 3.0, and JavaScript 1.2 support. With JavaScript we are skirting difficulties with JavaScript version compatibility and the related Document Object Model (DOM) standards issue, but we have not encountered any problems so far. The WebSubmit server system can be any system that runs an SSL- and HTML 3.0-capable Web (HTTP) server (daemon) and the Secure Shell (SSH) client code. The target systems can be any kind of system that supports SSH servers (daemons). These requirements arise because SSL is used to ensure a trusted connection between the client and server systems, while SSH fills the same role between the server and the targets. HTML 3.0 and JavaScript are required to support the HTML forms we use.

The WebSubmit server must have certain administrative features for use with WebSubmit. It must be possible to direct the Web server to place the client's certificate in the CGI environment when the client makes a request for restricted resources. This is essential to allow WebSubmit authentication, during which

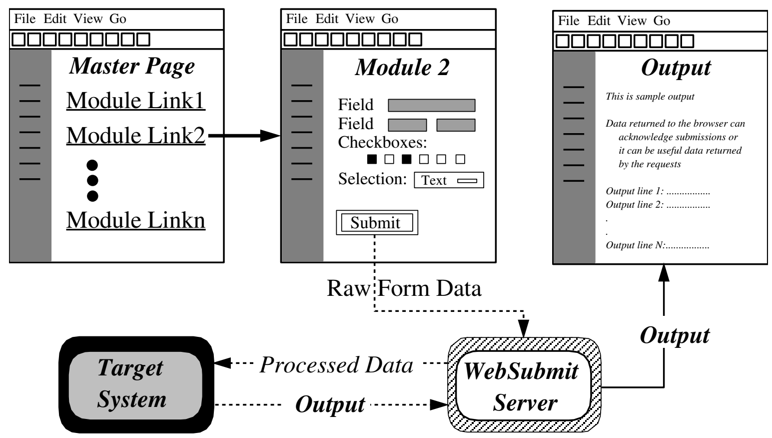

Figure 2: A simple WebSubmit transaction. The client connects to the WebSubmit master page and then selects the link for the desired application module. The module page is an HTML form, which is filled out and submitted to the server. The server processes the raw form data and executes any necessary code on the target system. Once these commands are executed the target returns the output to the server, from which it is then forwarded back to the user's browser.

the certificate is mapped to the appropriate username for the specified target system. The server should also allow specifying document MIME types on a per directory basis.[11] This facility is used to ensure that downloaded files produce a save-file-as-some-name dialog in the browser instead of viewing the file. The same effect can be achieved more awkwardly if the MIME types can be specified on a per extension basis.

The WebSubmit server and target systems must support SSH, which is used to ensure a trusted connection between the server and target systems. We have so far avoided requiring SSH on the client systems, because SSH for MS Windows systems is a commercial product, F-Secure SSH,[12] and we would like to make it possible to use MS Windows client systems without this investment. This restricts the means used to transfer files between the client and server to those provided under HTML 3.0, which are rather weak and function asymmetrically (see Section 2.4).

At this point we have no experience with WebSubmit on non-UNIX server or target systems. However, because of the difference in approach to remote execution with MS Windows systems it seems likely that it would take a fair amount of work to adapt WebSubmit to handle a Windows NT system as a target system. It is not normally possible to remote login to a Windows NT system, and that, in essence, is how the WebSubmit server reaches a target system. Given this, the applications, for example, would have to be set up as Windows client/server applications, or the version of Windows NT would have to be Windows NT Terminal Edition.[13] We have not investigated what it would take to secure these connections. The commercial version of SSH could perhaps be used. In any case, it seems likely that the CGI programs that communicate with the target systems would need considerable reworking. So, for practical purposes WebSubmit is restricted to UNIX targets. This is not true of the client and WebSubmit server systems, as far as we can see.

2.3 Application Module Design Elements

2.3.1 Application Interfaces

An application interface must exist for each of the desired services on target systems. HTML forms serve as the interfaces in WebSubmit, with each interface and its supporting CGI code referred to as an *application module*. Application modules used in the WebSubmit framework can come from either of two sources. If a site should chance to have already a set of HTML forms and CGI processing scripts (written in any language), then these can be plugged into the WebSubmit hierarchy directly with little or no modification. Alternatively, new scripts and forms can be written in a developer's favorite implementation language and then included. We are still considering ways of simplifying creation of new modules by automating as much as possible the writing of the CGI code (see Section 2.3.4).

A form-based approach to applications was elected in WebSubmit for a variety of reasons. First, most users are already familiar enough with forms, including HTML forms, to understand the basic elements of such interfaces. Second, a standard implementation of forms within HTML exists and is supported off the shelf by all browsers and servers, so that there is no need to develop or distribute and install the code to support the necessary widgetry. Third, the HTML forms interface is reasonably uniform in appearance, regardless of the browser and platform. Fourth, HTML forms are a stable, well-known technology that are easily used. Alternatives like Java would allow for a great deal more flexibility in terms of interface design and functionality, but introduce some performance and development issues.

In the future, it might be desirable to migrate to some other interface technology, when these have had time to develop and settle. Possibilities include Java, or rather, user interface libraries based on it; other client-side user interface development tools like the Tcl/Tk plug-in; or perhaps XML-based forms or other user interface specifications.

2.3.2 Modes and User Skill Levels

In an attempt to address the needs of both experienced and infrequent users of the system, most modules support at least two modes (advanced and basic). Modes present users with different subsets of parameters, in this case addressing different levels of control over the application, i.e., a varying number of elements are presented on the form and default values are chosen appropriately where elements are omitted in the basic mode. The modal approach can be extended to other forms of modality, at the discretion of application module developers.

2.3.3 Sessions

WebSubmit supports a notion of state that makes the software easier to use and more flexible from the user's standpoint. A session in WebSubmit consists of the set of values selected for the various form input elements for a single module. For each module, a default session exists to assign values to these elements in the absence of other information. In addition, a user can fill out any form and save the result as a named session to be loaded at a later time, thus reducing the need for repeatedly entering the same information.

2.3.4 Module Creation

One desideratum for WebSubmit has been to provide a method for producing new application modules easily. Producing modules is a matter of providing two CGI scripts, one to define the HTML form that solicits the input, and another to capture the entered values, check them for errors, submit code to the target system to accomplish the associated task, and produce a report on the execution of the task.

At present new scripts are hand coded, generally by modifying an existing pair of scripts. In practice this works rather well as a development strategy, though it has deficiencies for maintenance, and it tends to restrict production of modules to specialists. For these reasons we have been investigating alternative approaches that reduce the extent to which specialized knowledge of CGI, the WebSubmit programming environment, and the scripting language (Tcl/Tk) are needed. One approach would be to treat the script to be run on the target system as a procedure body and bind it to a list of parameters. The form generator script would be a generic tool that processed the parameter list to produce the form. The form processor script would be another generic tool that performed error checking, associated the actual parameters with the formal parameters, executed the task on the target system, and reported the resulting output.

2.4 Current Application Modules

Applications are divided into generic modules, which are deemed useful for any target system and therefore require the user to specify the target system, and host-specific modules, which are geared towards a specific service on a target system and specify the target implicitly.

2.4.1 Generic Modules

Three generic modules exist at this point: a command execution interface, a simple file editor, and a file transfer utility. The command interface allows users to execute arbitrary commands on remote (UNIX) systems. The file editor allows users to make quick changes to text files on remote systems and save them. The editing facilities are those of the HTML textarea element. The file transfer utility provides a way to transfer single files (text or binary) between systems. It, too, depends on the facilities of HTML.

The resulting transfer mechanism is somewhat asymmetric, due to the nature of file transfer in HTML. Without going into detail, this file transfer behavior is implicit in HTML browsers and our three part transaction model of client, server, and target. To do better we would need some auxiliary tool on the client system, such as SSH scp or SSH-protected ftp.

2.4.2 Host-Specific Modules

Host specific modules have been developed for three different HPC systems and the job queuing software currently (third quarter 1998) used with these systems at NIST:

1. the IBM SP2 with LoadLeveler
2. the SGI Origins 2000 with NQS/NQE
3. a Linux Pentium Cluster with LSF

The selection of systems is idiosyncratic to the NIST site. Other HPC systems and other queuing software could be supported.

For each of these systems we provide a general module for submitting batch jobs to the queuing system (e.g., LoadLeveler) and another module to monitor the jobs on the system. More specific interfaces have also been built for Gaussian 94 [14] (a quantum chemistry package), and for Message Passing Interface (MPI) jobs. In principle, there are few constraints on the types of modules that can be constructed. The limits are mainly the imagination and needs of the user community and the time of the developers.

2.5 Implementation

The entire WebSubmit package is written in Tcl, the Tool Command Language.[15, 16] Tcl was chosen as a development language because it is fast and easy to use, and supports rapid development and prototyping. Tcl has proved to be superbly adapted to the application. WebSubmit is broken up into several separate bodies of Tcl code: CGI scripts, service procedures, and configuration data. The code assumes Tcl 8.0, since namespaces and some other features of version 8.0 are used throughout. The CGI scripts are designed around Don Libes' excellent `cgi.tcl` library.[17] The remainder of the code was written by the WebSubmit development team.

2.6 Configuration

The WebSubmit configuration is stored in flat tables we call databases. The major databases used with WebSubmit are:

1. **Master Page Database:** defines the structure of the user-browsable list of modules

2. **Authorization Database:** defines the list of users and which systems they can access

These databases and parameters in the code are maintained by the local WebSubmit administrator with the aid of a maintenance tool, a graphical user interface written in Tcl/Tk.

In an effort to support some level of data hiding and to make loading sets of routines more transparent, the service routines are broken up into several functionally-distinct Tcl packages. All of the procedures and data for each package are then encapsulated inside a namespace whose name matches that of the encapsulating package. In addition, each database is stored within a specified, common data structure in its own namespace. In some cases, there is a correspondence between packages and databases (e.g., `auth.db` is manipulated by the authentication package `wsAuth.tcl`). Additional details about the Tcl implementation of WebSubmit can be found in ref. [18].

2.7 Performance Issues

WebSubmit is being used as a frontend to computations on HPC target systems. These computations consist of two sorts: invocation of system commands to generate reports on job status or the target system environment generally, and job submissions. The former typically involve short computations that users expect to lead to immediate output. The latter involve lengthy computations for which users expect immediate notification only of the success of the submission. Clearly the overhead added by WebSubmit is only relevant to the expectations of quick responses, for status and submission reports.

We have not attempted to measure the response times in question formally, because they are in large measure dependent on the fluctuating behavior of the local network and the load on target system, which would be factors whether or not WebSubmit were used. Apart from the latency of the network and target, there is also the issue of the time consumed by the CGI code on the WebSubmit server, and by the target system in responding to SSH requests. We have not observed any performance degradation due to CGI processing by our dedicated WebSubmit server, with the following exceptions.

We do notice the effect of overhead SSH requests to the target system. These are requests hidden in the WebSubmit code, as opposed to the main one made by the user. For example, WebSubmit might make additional SSH requests to retrieve job class parameters or to do error checking such as determining the existence of files referenced in the job. We have eliminated the first sort of overhead request by performing it out of line at specified intervals. The second sort of request we have elected to live with. Error checking always takes more time than crashing.

3 Authentication and Security

One of the primary concerns in a system like WebSubmit is security. Indeed, this is a primary concern in seamless and metacomputing systems in general.[4, 7, 6, 19] The definition of security varies with the context, but it usually encompasses authentication, authorization, and encryption. In the context of WebSubmit, the primary concern from a security standpoint is authentication. Since the user is executing commands on remote systems using the WebSubmit server as a proxy, it must be possible to establish the user's identity with certainty. Without strong authentication, unauthorized individuals – hackers – impersonating valid users could abuse WebSubmit for nefarious purposes, corrupting files to which the user has access, and possibly compromising in that fashion the integrity of systems in the cluster. They could also consume system resources at the user and the system's expense. This would probably not be considered a feature in most quarters, so systems have been incorporated into WebSubmit to prevent such breaches of security.

The previous version of WebSubmit [1, 2] used a combination of basic HTTP authentication or HT Access (via `.htaccess` files) with an SUID wrapper utility (`cgiwrap` [20]) to allow execution of CGI scripts as the required user. This architecture was not deemed adequate, since basic authentication is subject to simple dictionary attacks [25] and network snooping. In addition this approach was used in a two-party transaction model, in which the server and target were identical. It would be difficult to extend it to the three party model used at present.

The new, more robust, WebSubmit security architecture works in three stages:

1. Client-Server SSL Authentication
2. UserId Determination (WebSubmit Authentication)
3. Remote Execution via the Secure Shell (SSH)

These stages are described in more detail below. The security system is documented in ref. [21].

3.1 Client-Server SSL Authentication

An HTTP server that is capable of using the Secure Sockets Layer (SSL) protocol [22] handles requests for execution of CGI scripts or retrieval of static pages on the WebSubmit server system. Such a server is also referred to as an HTTPS server, from the URL scheme, https, used to invoke the SSL protocol. The WebSubmit Web server is configured to require bidirectional authentication (server-to-client and client-to-server). SSL is the current *de facto* standard for security, although this may change in the future to Transport Layer Security (TLS). A transition to TLS will not affect operation of the server since the two protocols are very similar and support the same functionality.

The server is configured only to accept certificates signed by Certificate Authorities (CAs) that are considered trusted by the WebSubmit installation. We have experimented with our own home grown CA, but we prefer and currently rely upon a NIST site CA maintained independently of WebSubmit. The choice of a CA is up to the WebSubmit administrator.

3.2 WebSubmit Authentication

Once the client's certificate is accepted by the Web server, and the client is requesting data in the WebSubmit directory hierarchy, the second stage of the process occurs: identity establishment and authentication translation. These processes occur on the WebSubmit server, and allow the WebSubmit user's username on the target system to be determined, so that the user's task can be executed in the associated account.

To support the mapping of the certificate to the username, the Web server must write accepted certificates to the CGI environment. We have experimented with extracting the identity of the user, in the form of a user identifier, from the certificate, using Leerssen's utility [23] to parse the certificate. The disadvantage of this approach is that it requires certificates to adhere to a structure known to WebSubmit that incorporates this information. We now use a digest of the certificate as the user identifier. This approach allows us to use any certificate at all that the Web server will accept. However, we need the digest of a new user's certificate before we can add the user to the WebSubmit authentication database. To permit this, WebSubmit is set up to capture the digests of all unauthenticated accessors' certificates, and pass these to us together with the text of the certificate. New users simply access WebSubmit once as an unauthorized user. We identify and interview them before adding them to the authentication database.

To produce a digest of the certificate it is hashed with the secure hash function MD5.[24] MD5 is a hash function that is difficult to invert and collision resistant, making it suitable as a way to compute fingerprints, or digests, of larger objects like a certificate. This digest serves as the WebSubmit user identifier. The authentication database maps pairs of user identifiers and target system names to the corresponding usernames on that target system, for all users and targets served by a given WebSubmit server. A user is allowed access to the WebSubmit system if the following conditions are met:

1. The WebSubmit server system Web server was accessed via the secure protocol (HTTPS).

2. The MD5 hash of the accepted certificate is contained in the authentication database.

3. Access privileges have not been coded as revoked for the certificate holder.

Failure to meet any of these criteria will result in a denial of access to WebSubmit services and an error message. The complete process of SSL authentication and WebSubmit verification is illustrated in Figure 3.

The inclusion of the digest in the database establishes the identity of the user, and retrieving the username indexed by the digest and a target system name provides the translation of the server's authentication to the target's authentication, completing the mapping of the certificate to the user's proper account on the relevant target. It is, of course, *imperative* that this username be determined correctly.

3.3 SSH Execution

Commands are executed on remote systems by the WebSubmit server using the Secure Shell (SSH) protocol.[26] SSH has the virtues of encrypting all transmissions made with it, and of being able to use a public-key cryptography to identify the remote system (the WebSubmit server) to the local system (the target system). This latter feature prevents unauthorized individuals from executing commands on the target system using IP or DNS spoofing. To avoid requiring users to enter their target system passwords to execute commands on a target, the home directory for the username *must* contain a `.shosts` file that allows the *WebSubmit server user* to execute commands in the user's account. For example, if the WebSubmit Web server is running as `nobody`, then the `.shosts` file must contain the following line:

```
serverName.serverDomain nobody
```

This raises some serious policy concerns that will be addressed in Section 5. The would-be spoofer must convince SSH that they are `nobody` on the WebSubmit server system, and this is where SSH's ability to insist on strong authentication of the host is important.

4 Administration

The WebSubmit administrator is responsible for several different tasks: installation, configuration of the Web server on the WebSubmit system, configuration of application modules for the site, construction and maintenance of the authentication database, installation and possible development of application modules, and, of course, general troubleshooting. It may also be necessary for the WebSubmit administrator to collaborate with (or perform as) the Certificate Authority for the system. The WebSubmit administrator is essentially the system administrator for the WebSubmit virtual machine.

4.1 Site Configuration

Clearly, different sites will have different needs in terms of the applications available to users, and these needs have been addressed. The main entry point to WebSubmit is a master page that has links to all relevant application modules, help, and configuration information. Application modules are organized hierarchically on the master page, each module under a relevant application class or heading. WebSubmit allows administrators to turn modules or application classes on and off as needed, or to add new applications, by changing the master page. The master page is generated automatically from the master page database, and adding a new module means adding a few lines to this database. The configuration tool mentioned in Section 2.6 can be used to do this. Users don't have access to the site-wide databases, but they can use the configuration features of WebSubmit to make undesired applications or classes invisible.

A very critical factor in WebSubmit security is the authentication database, hence configuration and maintenance of this database is of the utmost importance. On initial installation, the database needs to be configured to specify the systems in the cluster. The administrator must update the database as new users are added, obtaining the usernames for these users on the systems in the target cluster. If for some reason a user's privileges on the system are revoked, the database needs to be updated to reflect this fact. If this database is misconfigured or contains inaccurate information, the security of WebSubmit can be thoroughly compromised.

5 Policy Issues

Any meta- or distributed computing system must eventually grapple with issues of policy. Existing policies may determine whether these packages can be used as intended. With WebSubmit, we can differentiate internal policy issues, relating to basic functionality and the WebSubmit server, from external issues, dealing with interaction between the WebSubmit server system and the target systems in the cluster.

5.1 Internal Policy

5.1.1 WebSubmit Server System

The WebSubmit server should ideally be a dedicated system. WebSubmit is a collection of CGI and Tcl scripts; there is no single piece of compiled, executable code that is impervious to examination by users. It is easy to prevent users on client systems from examining or modifying the code, because they only see the output of the code executed on the WebSubmit server. It is more difficult to protect the code from users on the WebSubmit server. To prevent tampering with the WebSubmit source code, we recommend a server system with very few (or no) user accounts other than `root` and the WebSubmit HTTP user account. In addition, the latter should be an account without a password, without a useful home directory, and with minimal access privileges. Security measures are often good at preventing unauthorized, outside users from accessing or tampering with a system, but overlook the very real possibility that valid users may also attempt to corrupt the system, or, at least, to take advantage of it. If the server has no user accounts, this will not be a problem.

5.1.2 Web Server

Some installations, especially those with firewalls, take special care with access to their systems. Various sorts of traffic are considered security risks, and firewalls often block such things as HTTP traffic to and from Web servers. They may block logins from outside. However, if this is the case for a site trying to implement WebSubmit, and if usage originating from outside the firewall is deemed necessary, then at least one of these restrictions must be relaxed. One possibility is for the WebSubmit server and targets all to be placed inside the firewall, in which case the HTTP(S) traffic between outside clients and the server must be allowed to pass the firewall. The other possibility is for the WebSubmit server to sit outside the firewall, which must then freely pass the SSH login packets that flow between the server and the targets. Some satisfactory route must also be provided for the HTTP packets passing between inside clients and the WebSubmit server.

Even in the absence of a firewall, some versions of Web servers are considered insecure, and administrators have requirements concerning the type of server that their system uses. However, whatever Web server is used most support the minimal requirements of WebSubmit (see Section 2.2).

5.1.3 Certificate Authority

For SSL authentication of clients to be meaningful, the public-key certificate possessed by a client must be signed by a trusted third party (i.e., CA). Instead of trusting users to present valid information to the system, the issue then becomes trusting this issuer of certificates to vouch for the identity of the client. The difficulty is then to decide who (or what) should act as the CA. The design of WebSubmit can handle multiple CAs and any format of certificate. Extended discussions of certificate authorities and the difficulties involved in public key distribution can be found in ref. [25].

5.2 External Policy

The target systems in the WebSubmit cluster must meet certain requirements, and must exchange some information with the WebSubmit administrator. The primary software requirement for systems in the cluster is that an SSH daemon must be installed and properly configured. This should not be an impediment, since SSH has been ported to a vast array of systems,[26] and is increasingly considered a critical system utility. At present, the SSH protocol has not been formally adopted by the Internet Engineering Task Force (IETF), although the protocol has been submitted in draft form.[27] As part of the WebSubmit installation activities SSH public keys must be exchanged between the WebSubmit server and the target systems and added to the table of known hosts on the respective systems.

One potential sticking point relative to SSH is the need for an `.shosts` file in the home directories of WebSubmit users, configured to accept password free access from the Web server username on the WebSubmit server. This requirement may worry some administrators (and/or users), and could be a stumbling block, even though it should be impossible (i.e., impractically difficult) to spoof SSH into believing an interloper system to be the WebSubmit server system.

The other pivotal issue involving cluster systems is obtaining usernames from these systems for each valid WebSubmit user. This information is required in order to propagate properly the chain of trust established during authentication and authorization at the WebSubmit server, or, in short, to execute the user's task in the user's own account. If a policy is in place that prevents distribution of this information, then less reliable methods must be used (e.g., getting the username from the user). Hopefully, the WebSubmit and target system administrators will know each other, or overlap, in which case the distribution of login information will not be a major problem.

6 Conclusions

WebSubmit is a flexible, modular framework for accessing and using remote computing resources across the World Wide Web. Though it has been developed at NIST for use as an interface to high-performance computing systems, it is certainly not limited to this field of endeavor. WebSubmit should be useful in any circumstance where a user community need authenticated individual access to applications on remote systems and a certification authority is available. It is designed to be portable and can be installed at most sites with a minimum of effort. It can support an existing body of CGI code, as well as providing a framework for developing new applications. The security implemented in WebSubmit is robust and provides both strong user authentication and data encryption, although it produces some policy issues that may need to be addressed before it can be adopted. In summary, WebSubmit extends the basic conception of the Web as a data archive and retrieval system to one of a general computing environment.

7 Acknowledgments

The authors would like to acknowledge Robert Lipman and Katherine Pagoaga for their previous work on the WebSubmit project. We would also like to thank Don Libes for useful discussions of `cgi.tcl`. James Dray of the NIST security division provided useful insights into the WebSubmit security architecture.

8 Author Biographies

8.1 Ryan P. McCormack

Ryan McCormack is a Physical Scientist in the Information Technology Laboratory at the National Institute of Standards and Technology. He has a PhD in Materials Science Engineering, and prior to performing work on WebSubmit, studied order-disorder phenomena and phase equilibria in transition metal alloys and ceramics.

8.2 John E. Koontz

John E. Koontz is a Mathematician in the Information Technology Laboratory at the National Institute of Standards and Technology. He has an MS in Computer Science and an MA in Linguistics.

8.3 Judith E. Devaney

Dr. Judith Devaney is a Project Leader in Information Technology Laboratory at the National Institute of Standards and Technology. Besides leading a parallel applications effort, she also does research in Machine Learning.

References

[1] Robert R. Lipman, Judith E. Devaney, "WebSubmit - Running Supercomputer Applications via the Web", *Proceedings of SuperComputing 96*, November 1996, Pittsburgh, PA.

[2] John. E. Koontz, Ryan P. McCormack, and Judith E. Devaney, "WebSubmit - A Paradigm for Platform Independent Computing", presented at the *Workshop on Seamless Computing*, Reading, England, Sept. 1997.

[3] Current details of the WebSubmit project can be found at `http://www.itl.nist.gov/div895/sasg/websubmit/websubmit.html`.

[4] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit", *Int'l Journal of Supercomp. Appl. and High Perf. Computing*, 11(2), 115-128 (1997).

[5] For information on the Legion project see `http://www.cs.virginia.edu/~legion`.

[6] UNICORE project home page (`http://www.kfa-juelich.de/zam/RD/coop/unicore/`).

[7] Jim Almond, "UNICORE: Secure and Uniform Access to Distributed Resources via the World Wide Web", white paper at `http://www.kfa-juelich.de/zam/RD/coop/unicore/`.

[8] For information on LoadLeveler see `http://ppdbooks.pok.ibm.com:80/cgi-bin/bookmgr/bookmgr.cmd/BOOKS/LLL34A41/CCOVER`.

[9] For information on NQS (NQE) see `http://techpubs.sgi.com/library/infosearch`.

[10] For information on LSF see `http://www.platform.com`.

[11] See for example, S. Spainhour & V. Quercia. "Webmaster in a Nutshell", O'Reilly & Associates (Bonn, etc., 1996), Chap. 25.

[12] For information on F-Secure SSH see `http://www.datafellows.com`.

[13] For information on Microsoft Windows NT Terminal Edition see `http://www.microsoft.com/ntserver/terminalserver/default.asp`.

[14] Gaussian94 Home Page (`http://www.gaussian.com`).

[15] See for example, Brent Welch, "Practical Programming in Tcl/Tk", 2nd Edition, Prentice-Hall PTR (New Jersey, 1997).

[16] Tcl/Tk (Scriptics) Home Page (`http://www.scriptics.com`).

[17] Don Libes, "Writing CGI Scripts in Tcl", *Proceedings of the Fourth Annual Tcl/Tk Workshop '96*, Monterey, CA, July 10-13, 1996, USENIX Association (Berkeley, CA).

[18] R. McCormack, J. Koontz, J. Devaney, "WebSubmit: Web-Based Applications with Tcl", NISTIR 6165, June 1998, National Institute of Standards and Technology.

[19] W. A. Wulf, C. Wang, and D. Kienzle, "A New Model of Security for Distributed Systems", University of Virginia CS Technical Report (`http://www.cs.virginia.edu/~legion/papers/CS-95-34.ps`).

[20] Nathan Neulinger, `cgiwrap` home page (`http://www.umr.edu/~cgiwrap`).

[21] R. McCormack, J. Koontz, J. Devaney, "An Authentication Framework for WebAccess to Remote Hosts", NISTIR, in preparation 1998, National Institute of Standards and Technology.

[22] Netscape SSL Overview (`http://sitesearch.netscape.com/eng/security/SSL_2.html`).

[23] Scott Leerssen, `certutil.c` (`http://www.mindspring.com/~leerssen/`).

[24] Bruce Schneier, *Applied Cryptography*, 2nd Edition, John Wiley & Sons (New York, 1996), pp. 429-430, 436-441.

[25] Bruce Schneier, *Applied Cryptography*, 2nd Edition, John Wiley & Sons (New York, 1996), pp. 185-187.

[26] Tatu Ylönn, SSH Web site (`http://www.ssh.fi`).

[27] Internet Draft for SSH (`http://www.ietf.org/ids.by.wg/secsh.html`).

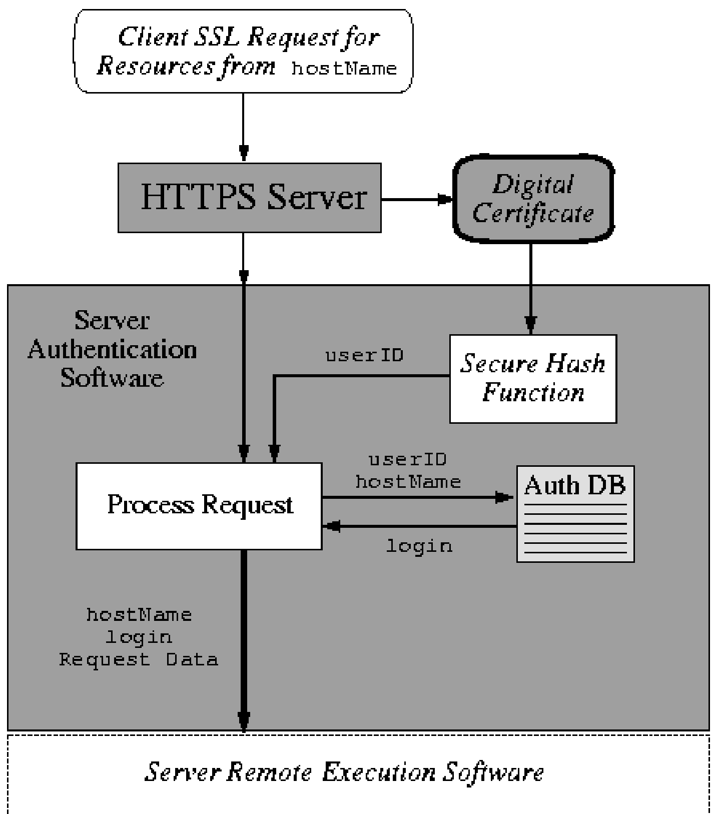

Figure 3: Identity establishment and authentication translation. The WebSubmit HTTPS server is contacted by the user's client using an SSL connection with client authentication; the user wishes to access target *hostName. The client's certificate is made available to the server software, which constructs a user identifier userID using a secure hash function applied to the certificate. The authentication database, indexed by userID and hostName, yields the username login for the client on hostName (if possible). The user's request is then processed, and all information is forwarded to the software to to invoke execution for login on hostName.*