ezHPC Security Architecture

George Moncrief, Trent Townsend, and Scotty Swillie
USACE Engineer Research and Development Center, Major Shared Resource Center (ERDC MSRC), Vicksburg, MS
{George.C.Moncrief, Trent.W.Townsend, Scotty.Swillie}@erdc.usace.army.mil

Abstract

High performance computing (HPC) within the Department of Defense (DoD) is vital to the execution of the research and development mission. However, users must access these resources using interfaces that are complicated, specialized, primitive, and opaque. ezHPC was developed to address this complexity. The ezHPC project is interesting in that it abstracts for the user many of the technical details and tools related to HPC within the DoD High Performance Computing Modernization Program (HPCMP) while providing a Web-based application that allows user access to HPC resources.

Leveraging open-source, widely available technology, the ezHPC security model addresses the DoD security requirements for authentication, confidentiality, availability, and integrity. This paper discusses an architecture that employs end-to-end encryption, open-source technologies, message authentication codes, and authentication of users via the HPCMP-modified implementation of the Massachusetts Institute of Technology Kerberos. This architecture provides ezHPC the flexibility to reduce the technology barrier to those unfamiliar with interactive use of HPC systems while providing significant capability to more advanced users. The subsequent abstraction of HPC resources constitutes a security model for any Web-enabled application programming interface.

1. A Brief Historical Perspective

The High Performance Computing Modernization Program (HPCMP) deploys commercial, high performance computing, storage, and networking to the Department of Defense (DoD) Science and Technology and Test and Evaluation communities. The platforms providing these high performance technologies have historically occupied a niche in computing when compared with the numbers and cultural changes brought about through single-user, so-called personal computers.

The user interfaces provided with these systems by their vendors reflected their status as niche providers and were intended for and used by the very few, highly skilled niche consumers of high performance systems. As the price-performance ratio has fallen and market penetration increased with HPC systems, the number of users with access to these systems has increased. Consequently, the failure of HPC user interfaces to expose the available computational power to the less-initiated user has become increasingly obvious. They are often character-mode, command-line interfaces. Sometimes they are more graphical in nature, employing color and form that represent function, but without the refinement found in more popular, more widely deployed operating systems. HPC user interfaces are complicated. They mirror the architecture of their systems by being very specialized, requiring significant investment of the user's time to learn, and being primitive in the granularity of available operations.

The HPCMP has developed and provided to users a collection of command-line clients as a means of interaction with HPC systems. These clients, though very capable, have all the inherent user interface failures mentioned already. The clients function best for a subset of users who have strained to develop their skills and memory to the point that the interface becomes transparent to them. The commands and functions of the systems have become a mental expectation. Conversely, those users who have not developed this ability are duly hampered in their efforts to accomplish efficient interaction with the HPC systems.

By contrast, the Web browser interface to the World Wide Web has taken the world by storm in a relatively few years. The Web browser itself has become a popular icon of computing. Many variations on a theme exist, but the essential and well-understood functions of Web browsers are easily identified by novice users. The Web browser has mapped interaction to common sense.

HPCMP Users Group Conference (HPCMP-UGC'06)
0-7695-2797-3/06 $20.00 © 2006 IEEE

Trend Micro, Inc.
EX1009-1

Authorized licensed use limited to: University of Texas at Austin. Downloaded on March 31,2023 at 18:01:40 UTC from IEEE Xplore. Restrictions apply.

2. Introduction

The ezHPC project at the US Army Engineer Research and Development Center Major Shared Resource Center abstracts the function and use of HPC batch computing systems and makes those functions available through a Web server interface in a way that enables developers to write applications that take better advantage of the associations already present in almost everyone's mind.

The ezHPC project has studied both the batch processing and most commonly used input/output functions of HPC systems and developed an application programming interface (API) that organizes and codifies those functions. The project deployed this API as a Web-enabled technology to provide the greatest capability to the greatest number of users. The ezHPC project is authoring both a client using Microsoft .NET development tools and a Web browser client to demonstrate the value, elegance, and portability of this API.

But this is a double-edged sword. The pervasive presence of the Web technology, making the API available, also offers its greatest potential for exploitation. Using a Web server to provide an API means that anyone who understands the transport protocols involved in communicating with the Web server may potentially abuse the service in one of three locations: the client that communicates with the Web server via the API, the Web server advertising the API, or the communications between the client and Web server.

1. Exploitation of the client may permit an attacker to assume the identity of a legitimate user, issuing commands that take advantage of the user's access to attack the remote system or violate the integrity of the user's data.
2. Exploitation of the API at the Web server may permit an attacker to control the Web server and, thus, the relationships in which the Web server is trusted. The Web server is trusted by and communicates with other systems on the local network to which it is connected, the HPC systems with which it corresponds on behalf of the user, and at least one other system used to manage a database of user metadata.
3. Exploitation of the communications between the client and the Web server may permit an attacker to masquerade as the Web server to the client or the client to the Web server. If the Web server is trusted by the client, the client will provide valid, useable login credentials, on behalf of the user, to the attacker. If the client is trusted by the Web server, the Web server may provide valuable user data to the attacker.

Protecting these identities and relationships is paramount to preserving the integrity of the data that is processed by the systems that trust them.

This paper discusses the measures taken in the design of the ezHPC project to preserve the attributes of confidentiality, integrity, authentication, and nonrepudiation as they relate to the identity of the client and of the Web server, and the communication between these entities, as well as protecting the API from abuse. In doing this, it illustrates how open-source software has been used to inexpensively accomplish development of security features and enable much-needed user interface development.

3. Design Features

Some prominent assumptions basic to the security of this project are briefly described below:

1. The system on which the client application executes is either a single-user system or is a multiuser system that assigns unique accounts to individual users and distinguishes unprivileged user accounts from privileged user accounts. Individual users possess unique credentials that uniquely identify them to the Web server. *Note that use of these credentials by someone other than the person to whom they are assigned defeats all further attempts at security or privacy.* Multi-user systems that feature elevated privileges for certain groups or users must have policies that restrict the user(s) or group(s) with elevated privileges from inappropriate use of credentials belonging to other users.
2. Physical access to the server computer system has been restricted such that only authorized staff may gain contact with the server.
3. The operating system on the computer executing the Web server must have been configured for secure operation in accord with relevant security policy and best practices. Best practices include steps to disable unnecessary services, and assign file system permissions and ownership consistent with existing users, keep operating system patches up to date, and exercise access control where appropriate.
4. The Web server has been configured with default settings updated to control access, and default or unnecessary scripts, files, etc., have been removed from the Web server content directory tree.
5. A code review has been conducted of all scripts and source code for executable binaries used by the Web server to provide content to the client. This code review should be performed by

Trend Micro, Inc.
EX1009-2

Authorized licensed use limited to: University of Texas at Austin. Downloaded on March 31,2023 at 18:01:40 UTC from IEEE Xplore. Restrictions apply.

someone familiar with secure software coding practices.

6. If a database provides content to the Web server for distribution or transfer to the client, all input passed to the Web server from the client should be sanitized, prior to use, by someone familiar with the data access procedures.

The steps indicated above cohere as a process, or layer of protection. The absence or thoughtless application of any of them may result in the ultimate failure of all of them to protect the systems from attack. A chain is only as strong as its weakest link.

Initial authentication of users is performed by a system called "Kerberos." Kerberos is a network authentication protocol developed and maintained by the Massachusetts Institute of Technology (MIT) and employs secret key cryptography to obtain a high level of confidence regarding the identity of a client user. MIT provides an open-source implementation of Kerberos that has been reviewed and used for several years by many governmental, private, and public sector organizations all over the world. The Kerberos implementation used by the ezHPC project has been used by the HPCMP for several years to authenticate its user community to DoD systems deployed across the United States.

The primary authenticating principle used by the Kerberos protocol is that of a shared secret. The Kerberos server, called a "KDC," stores a secret given to it by and known only to the client user. Often, this secret is simply a password. A client system employing Kerberos for authentication can obtain a password, or other identifying object, from a client user and transform that object in a way that makes it virtually impossible to identify the original object. The transformed object can then be passed to a KDC, and the KDC will perform the same transformation operations on its copy of the shared secret. If the object transformed on the KDC is identical with the object received from the client user, the KDC will agree that the client is who they say they are and provide the client user with a credential endorsing their identity.

By design, ezHPC is more than a simple client/server application that is experienced by the user. A client application connects to the ezHPC Web server, but the Web server performs functions that more closely resemble those of a combination proxy/bastion host. In order to abstract the complexity of the HPC system interface from the user, the API receives commands from the ezHPC client, translates those commands into the appropriate form, and passes them to the HPC system for execution. Upon receiving the results of the command from the HPC system, the ezHPC application server provides these data to the ezHPC client. Ultimately, the result is an interface between the client and the target HPC system. However, all communication between the client and the HPC system(s) passes through the Web-enabled ezHPC application server and is processed by the API.

In addition to the three components mentioned thus far, the client browser, the application server, and the HPC system; ezHPC also employs a database server to provide application support by storing per-user information, also called user profile data, about user jobs and HPC system configuration. This database is located on a separate system from the ezHPC application server, restricting the database's exposure to only the ezHPC application server. Neither the client application nor the HPC system ever directly connects to or receives connections from the database server.

Below is a diagram showing the components and interrelationships of the ezHPC project.

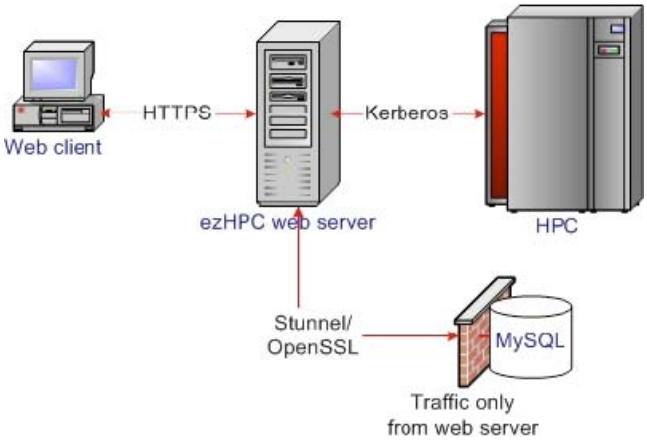

Figure 3.1

Because the target HPC systems are comparatively unlimited in number and physical location, the security model utilized by the ezHPC project will not discuss security practices exercised by administrators of the systems to which ezHPC will connect beyond that already discussed above. However, the design of ezHPC introduces a well-defined interface between the client and the HPC system that adds an additional layer of protection to both. The design of the API lowers the number of potential attack vectors between the client and the HPC system. A client does not have a terminal session established to an HPC resource, so the compromise of a client system does not directly expose an existing, already authenticated connection to the attacker. Authentication credentials may be stolen by an attacker to be used for an HPC system compromise attempt, but the use of short-lived credentials limits even further this window of opportunity.

After discussing the critical prerequisites of host security applied to all components, the next area of exploitation are the communications between each component. To enforce the principles of confidentiality,

Authorized licensed use limited to: University of Texas at Austin. Downloaded on March 31,2023 at 18:01:40 UTC from IEEE Xplore. Restrictions apply.

authentication, integrity, and nonrepudiation related to these interactions, a number of open-source tools were utilized. The first target of communications exploitation is the connection between the ezHPC client and the ezHPC application Web server. The initial authentication is executed by the application server on behalf of the user. If successfully authenticated, the user is given an Advanced Encryption Standard (AES)-encrypted Kerberos credential, or ticket, and a unique user identification number (UID) that are both stored only in memory. The Kerberos ticket is wrapped in an additional layer of strong encryption using a secret key. Along with the user's UID, this key is stored in the user's profile in the database. When the client issues a service call to the API, the encrypted ticket and UID are included in the call. The API compares the UID with that stored in the database and tries to use the associated encryption key to decrypt the Kerberos ticket that is used to issue commands to the HPC systems. If the decryption is successful, the request is issued to the HPC system. In this way, every service request includes a layer of authentication at the API level. To ensure the confidentiality, integrity, and nonrepudiation of this communication between the client and Web server, the Hyper Text Transport Protocol over Secure Sockets Layer (https) was utilized. Https is syntactically identical to the http protocol, but the client/server interaction is performed over an encrypted Secure Sockets Layer (SSL) protocol connection. Along with encrypting the data, the SSL protocol creates a type of digital signature called a Message Authentication Code (MAC) using a cryptographic hash algorithm. The MAC value protects both the integrity of a message and its authenticity by allowing those who possess the secret key to detect any changes to the message content. To ensure the nonrepudiation of the sender, https also utilizes a session key. The client that possesses this key is presumed to be the legitimate originator of the message.

During the progress of a user's session, many queries to the supporting database may be made unbeknownst to the user. Communication between the Web server and the database server is tightly controlled to provide an acceptable level of trust. To authenticate to the database for each session, the application server must provide a single username/password combination that is changed manually every 60 days. This credential is stored on the Web server in an encrypted file readable only by the Web server. To certify the confidentiality and integrity of this interaction, an open-source tool called Stunnel was used in conjunction with OpenSSL. Stunnel is a program that provides end-to-end encryption of arbitrary Transmission Control Protocol (TCP) connections inside a pre-existing SSL tunnel. For ezHPC, Stunnel secures a non-SSL aware MySQL database server. A feature of Stunnel is that it is TCP Wrappers-aware. TCP Wrappers adds a layer of access control to applications using its library, providing source address filtering for connection requests sent to local server applications. Stunnel uses TCP Wrappers in the ezHPC project to limit connections to only those from the specific IP address of the Web server. However, IP addresses may easily be spoofed. To address this, a second feature of Stunnel was used; that is, Stunnel may be configured to request a specific Public Key Infrastructure certificate at connection time and reject the connection if the client cannot provide this certificate. Spoofing the IP address and foiling TCP Wrappers are possible, but possession of the correct certificate is exceedingly difficult to fake. In this way, nonrepudiation is ensured assuming the certificate for the Web server is kept secret.

The actual database running on the database server is a MySQL-AB product. While the default configuration of this database accepts connections from any host, it was reconfigured to only allow connections from the local host. Of course, the Web server requires connections to the database that execute on a different physical system. As mentioned, Stunnel was the method used to secure the traffic from the Web server to the database server. The end of the encryption tunnel running on the database server receives packets from the Web server and forwards them to the correct port listening on the local host. This design makes the connection between the database listener and the Web server limited to the local host, rather than exposing the ear of the database to the world. This affords the favorable condition of denying all true external connections to the database, thus not advertising the listening database.

Finally, the communication between the Web server and the HPC system must be addressed. The Web server issues service requests to the HPC systems on behalf of the user, via a version of Secure Shell (OpenSSH) that has been modified to accept only HPCMP Kerberos authentication. As stated above, Kerberos ensures the authentication of the user, while both Kerberos and OpenSSH guarantee the confidentiality of the transmission by using strong encryption of the commands issued to and responses received from the HPC systems. Moreover, Kerberos and OpenSSH utilize a session key to provide nonrepudiation of the sender and a MAC to ensure message integrity.

4. Conclusion

The proliferation of open-source software has significantly reduced the burden of securing projects such as ezHPC. The combination of open-source software with World Wide Web technologies encourages the design and creation of software that is both easy to use and trustworthy. Readily available, up-to-date tools such

Trend Micro, Inc.
EX1009-4

Authorized licensed use limited to: University of Texas at Austin. Downloaded on March 31,2023 at 18:01:40 UTC from IEEE Xplore. Restrictions apply.

as OpenSSL give the software engineer a much greater capability to virtually plug in confidentiality, integrity, and nonrepudiation. Kerberos makes industrial-strength authentication a low-hanging fruit. Open-source software has provided many of the tools that make low security-risk application development more a question of design and far less an issue of code development, even though excellent code implementation cannot alleviate problems caused by poor application design. These authors believe that ezHPC has provided an application design that not only offers a low risk profile but also gives users a significantly improved interface to HPCMP systems. ezHPC reduces the interface complexity for novice users and gives conveniences previously unavailable to experienced users.

References

1. Kerberos, http://web.mit.edu/kerberos/.
2. OpenSSL, http://www.openssl.org/.
3. Apache Web Server, http://httpd.apache.org/.
4. MySQL, Database Server, http://dev.mysql.com/.
5. Stunnel, http://www.stunnel.org.

Trend Micro, Inc.
EX1009-5
IEEE COMPUTER SOCIETY

Authorized licensed use limited to: University of Texas at Austin. Downloaded on March 31,2023 at 18:01:40 UTC from IEEE Xplore. Restrictions apply.