Institute of Physics Publishing Journal of Physics: Conference Series **46** (2006) 545–549
doi:10.1088/1742-6596/46/1/075 SciDAC 2006

Web-based secure high performance remote visualization

R J Vickery[1], A Cedilnik[2], J P Martin[1], Y Dandass[3], T Atkison[3], R J Moorhead[1], J Clarke[4] and P Adams[5]

[1]Mississippi State University, HPC Building, Box 9627, Mississippi State, MS 39762, USA
[2]Kitware, Inc., 28 Corporate Drive, Suite 204, Clifton Park, New York 12065, USA
[3]Mississippi State University, Butler Hall, Box 9637, Mississippi State, MS 39762, USA
[4]US Army Research Laboratory, Attn: AMSRL-CI-HC, Aberdeen Proving Ground, MD 21005, USA
[5]ERDC MSRC, Bldg 8000, 3909 Halls Ferry Road, Vicksburg, MS 39180, USA

rvickery@gri.msstate.edu

Abstract. This paper describes recent work on securing a Web-browser-based remote visualization capability for large datasets. The results from a security performance study are presented.

1. Introduction

This work investigates a Web browser-based interactive visualization to allow interactive large data visualization and analysis from the desktop within the Department of Defense (DoD). Remote visualization (RMV) is requested more frequently than any other visualization service, and is often the first thing requested by a user with large dataset analysis requirements. RMV is technically possible, but a unified approach is needed across the DoD to make it successful, efficient, and useful. An interactive browser-based capability is the easiest to deploy to multiple desktop platforms, since no additional software beyond a browser is needed.

There are a number of technologies available for remote visualization [1]. The key security feature that differentiates this application is that it follows specific High Performance Computing Modernization Program (HPCMP) security guidelines for Kerberos authentication plus additional encryption and security on the ticket granting ticket (TGT) [2][3].

This work builds upon the results of an Army Research Lab funded SBIR phase II project with Kitware, Inc. that exploits ParaView, and has been closely coordinated with the User Interface Toolkit (UIT) effort [4]. The resulting browser-based product is part of ParaView Enterprise Edition (PVEE). The interactive performance of PVEE with additional encryption and authentication requirements has been investigated in order to facilitate the use of RMV by the DoD user community [5][6]. The emphasis has been on improving performance in the context of secure communications.

The security implementation overview and initial authentication performance studies are discussed. Although the design and testing were done with the DoD version of Kerberos in mind, the implementation would work equally well with the MIT or other open-source version of Kerberos.

2. Overview

The web visualization component of ParaView Enterprise Edition is a framework to perform visualizations and data processing within a Web browser-based environment. It contains project management tools as well as interactive previews of the running visualizations. The system allows users to run an arbitrary number of visualizations and visualization sessions, as well as generate images and animations. Figure 1 and Figure 2 show common views from within WebVis.

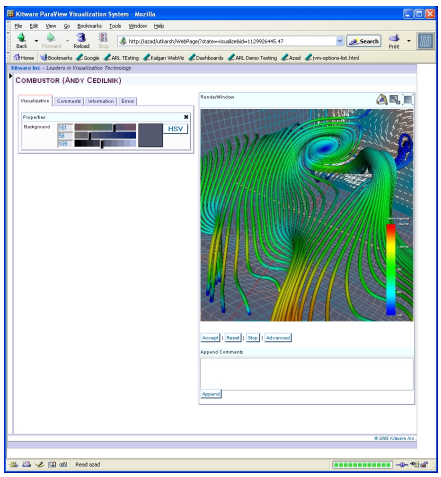

Figure 1. Example Visualization With Editable Fields and Annotations.

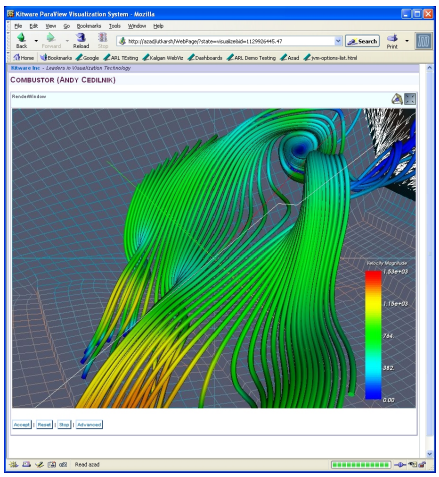

Figure 2. Full Screen Visualization Render Window.

The PVEE framework is based on several components (see Figure 3). These components include the Web Client, Web Server, Database, User Connections Manager (UCM), Visualization Manager (VM), Server Runner (SR), and ParaView Backend Server (PBS).

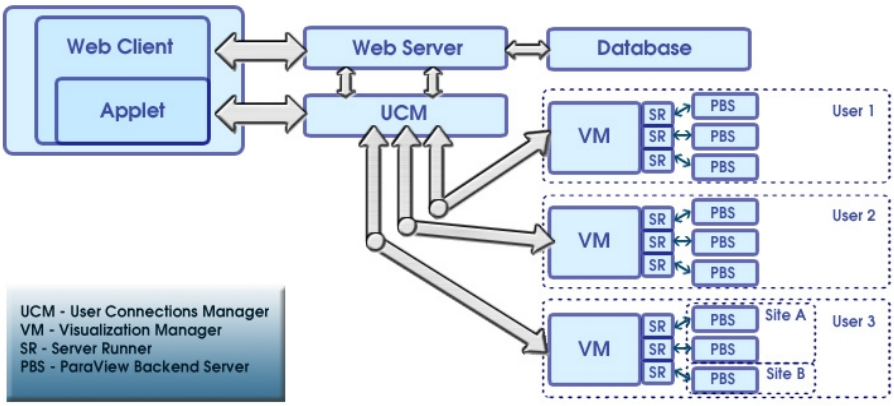

Figure 3. WebVis Architecture

The Web Server performs all the necessary communication between the client and various server components, as well as the necessary authentication of the client. Once the client is authenticated, random keys are generated that are then used for the life of the user's session. The UCM secures incoming connections to the system and makes sure only connections that are authenticated are accepted. The UCM also routes the visualization traffic between the client and an appropriate VM. The VM is responsible for managing the running visualizations for a specific user (one per user). It is started when the user logs into the system and persists until the last visualization terminates. The UCM routes all of the user's visualization sessions to the VM, which sends the request to the actual

visualization server through the use of an SR. The SR is a simple program that abstracts the running of visualizations for the site. The PBS performs all of the actual reading, processing, and visualizing of data. It is an extension of the regular Desktop ParaView server that supports encryptions and image transfers appropriate for PVEE .

3. Secure Communications

The design of the UIT Kerberos authentication process was adapted for PVEE [4]. The primary difference between the UIT and PVEE Kerberos authentication implementations is that PVEE does not use a "pipe cache" for caching requests, but instead, uses a memory cache. This is because, in order to streamline execution, the PVEE authentication modules do not execute the standard Kerberos applications (e.g., kinit and krsh) that require use of a shared credential cache across process invocations. Instead the PVEE modules replicate the required subset of knit and krsh functionality using the Kerberos API directly, thereby obviating the need for a shared cache.

3.1. Kerberos Authentication Dataflow

This section describes the process of mutually authenticating PVEE users and PVEE servers (see Figure 4):

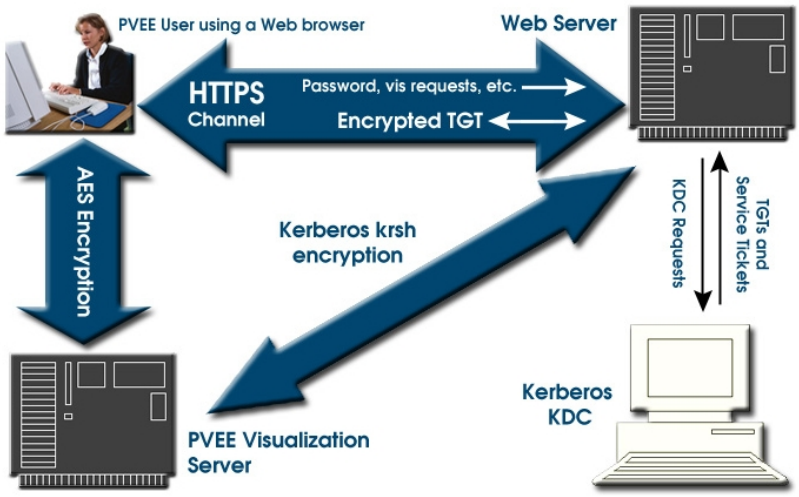

Figure 4. Kerberos Authentication Process

1. The client uses a browser to connect to the PVEE Web server and gets the PVEE login page over an SSL connection (i.e., the connection uses the HTTPS protocol). This login page presents an HTML form that asks for the user to enter his/her Kerberos principal (user ID), password, and SecureID. This user identification data is sent to the Web server which starts the pvee-login CGI program.
2. The pvee-login CGI program connects to the Kerberos KDC and obtains the user's TGT using the user's identification information.The pvee-login program serializes and encrypts the TGT using a randomly generated "aes256-cts" key. An MD5 hash of the encrypted TGT is also generated. The (hash, aes256-cts-key) pair are stored in a database on the Web server machine to be used in the future for decrypting the encrypted TGT (see step 3 below). The encrypted TGT is returned to the user's browser in the form of an HTML cookie over the SSL connection.
3. The user's browser now has the output of the pvee-login (i.e., a web page showing various PVEE function options and the encrypted TGT cookie). The user clicks on a button requesting a PVEE function. The browser sends the request specification and the encrypted TGT to the Web server over an SSL connection. The Web server starts the pvee-krb-auth CGI

program on the Web server machine. The pvee-krb-auth program and computes the MD5 hash of the cookie data (i.e., the encrypted TGT). The hash is used to lookup the decryption key from the database on the Web server machine. Next, the encrypted TGT is decrypted and validated. The validated TGT is used for obtaining the Kerberos ticket for the krsh service for the machine that has the pvee-ucm program for the visualization requested by the user.

4. The pvee-krb-auth program now has a ticket for the krsh service on the visualization server. The pvee-krb-auth program connects to the krshd program on the visualization cluster and, using the krsh ticket, performs mutual authentication between the user and krshd in order to execute the pvee-ucm program on behalf of the user. After invoking the pvee-ucm program on the visualization cluster, the pvee-ucm program sends the PVEE visualization applet to the browser and returns control to the Web server while keeping the connection with the krshd open in order to continue execution of the pvee-ucm process on the visualization cluster. The PVEE applet and the pvee-ucm communicate with each other using an AES encrypted channel. During its lifetime, pvee-krb-auth uses a "memory" cache for Kerberos credentials. This means that other processes running on the Web server machine cannot access the cache data.

3.2. Kerberos Authentication Study

Figure 5 charts the results from experiments designed to measure the latency of the Kerberos authentication software components, including network delays. In this study, authentication latency is defined as the interval of time starting when the user submits his/her identification and passwords to the authentication module using the Web browser and ending when the PVEE UCM establishes a connection with the user's browser.

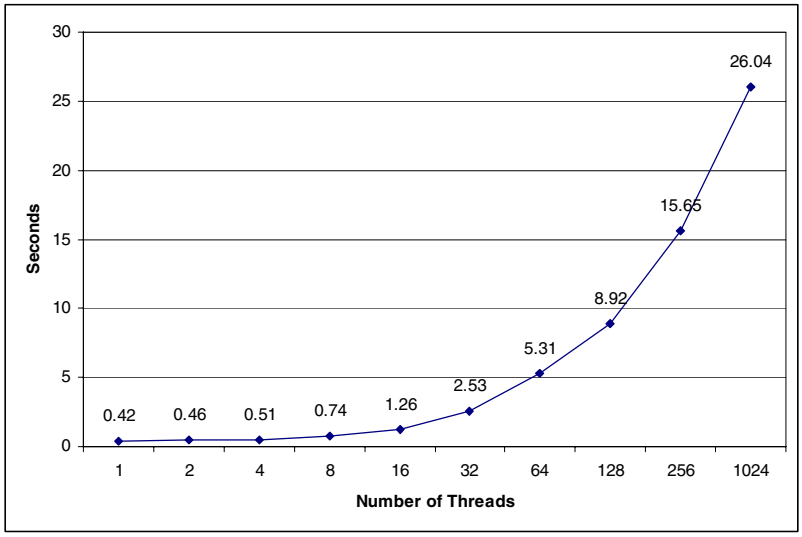

Figure 5. Kerberos Authentication Latency Study

For this experiment, a C program that simulates the users' actions was created. This program connects to the Web server and initiates Web transactions and captures responses (e.g., the encrypted TGT). This program also records the latency interval start time. Another program captures the connection requests from the UCM module that indicates that the authentication is successful and records the latency end time. The MD5 hash of the encrypted TGT is used to correlate the corresponding start and end time of an authentication request. Separate 2.4 GHz dual Xeon workstations were used to execute the simulated client, the Web server, the KDC, and the PVEE server. These four workstations were interconnected using a single 100 Mbps switch. In addition to these workstations, an external (i.e., outside of the Mississippi State University campus) workstation was also used for providing Secure ID pre-authentication capability.

The experiments are designed to measure the average latency incurred for an exponentially increasing number of simulated near-simultaneous requests. During a single experimental run, the simulated client program starts 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024 threads that simulate the user's interactions with the Web server. Figure 5 shows that the average latency is under one second when there are eight or fewer near simultaneous requests. Although the latency appears to increase exponentially, note that the number of requests is also growing exponentially. Because the authentication happens only once at the beginning of a visualization session, it is anticipated that the impact of this latency will be negligible, especially in the face of the time required by browsers to transfer and render complex web pages with embedded applets. Clearly, this figure represents the authentication latency specific to the experimental platform and the results will vary between installations.

4. Conclusions

We described a methodology for securing a web-browser-based system for interactive visualization from the desktop that incorporates a web server. In our study on the latency of the Kerberos authentication components we showed that the impact of this latency was acceptable for our particular configuration. Additional experiments where a number of different workstations simulating user input would be interesting future work. These experiments could also include a study of latency when different machine configurations are used (e.g., the Web server and KDC are placed on the same machine). Some of the most important information from these test scenarios may not necessarily be in the numbers, but whether the application can be used with acceptable perceived latency by a DoD user. We strive to make this a priority.

Having conducted detailed discussions with key DoD security personnel throughout the development, we fully expect a secure version of PVEE to be approved for deployment at some point. We look forward to helping users streamline the visualization process and access the results through a simplified browser based interface.

The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United Stated Air Force, Department of Defense, United States Government, or Mississippi State University. This work was made possible through support provided by DoD HPCMP PET activities through Mississippi State University under contract No. GS04T01BFC0060.

References

[1] Vickery R 2005 Remote visualization: what it could be in the DoD HPCMP Aeronautical Systems Center Major Shared Resource Center Wright Cycles Fall pp 22-23 http://www.asc.hpc.mil/aboutus/journals/fall05.pdf

[2] High Performance Computing Modernization Program 2004 Guidelines for implementing secure access to HPCMP systems http://www.hpcmo.hpc.mil/Htdocs/SECURITY/dren_secure_access_guidelines.pdf

[3] Garman J 2003 Kerberos: The Definitive Guide (Sebastopol, CA: O'Reilly & Associates, Inc.)

[4] Duett P, Monceaux W and Rappold K 2004 EZHPC: easy access to high performance computing Proceedings of the HPCMP Users Group Conference pp 289-292

[5] Vickery R, Cedilnik A, Moorhead R, Dandass Y, Atkison T and Martin J 2005 WebVis secure communications model and preliminary performance study PET RMV-KY5-001 Report 1 December

[6] Moorhead R, Vickery R, Cedilnik A, Dandass Y, Atkison T, Martin J, Vaughn R and Martin K 2006 High-performance secure remote visualization using WebVis PET RMV-KY5-001 Report 31 May