XOS-SSH: A Lightweight User-Centric Tool to Support Remote Execution in Virtual Organizations

An Qin Haiyan Yu Chengchun Shu Bing Xu
Institute of Computing Technology, Chinese Academy of Sciences
Beijing,China
{*qinan,shuchengchun,xubing*}*@software.ict.ac.cn*
yuhaiyan@ict.ac.cn

Abstract

Large-scale virtual organizations (VOs) often comprise resource providers from different administrative domains, each probably with a specific security model. Grids try to solve this problem by providing a new security infrastructure featured with single-sign on (SSO). However, the usability of Grids is often impaired by the complicity of configuring and maintaining the new security infrastructure as well as adapting to new interfaces of security enabled services. The co-existing of different Grid platforms and SSO solutions among resource providers makes this situation even worse. In this paper, we present XOS-SSH, a lightweight user-centric tool to support remote execution of jobs among heterogeneous nodes of VOs. XOS-SSH is a modified version of the widely used OpenSSH tool based on several OS-level VO support mechanisms developed in XtreemOS project [21]. XOS-SSH adopts a pluggable framework that is capable of supporting different authentication schemes and making them transparent to shell users. The performance evaluation of XOS-SSH around NAS Parallel Benchmarks (NPB) shows that our current implementation incurs trivial overhead comparing to the unmodified one.

1 Introduction

Virtual Organizations(VOs) have generated increasing attention from both research and industry communities for that they support augmented and cost-effective resource sharing among geographically distributed service providers (SPs). The foremost issue of building VOs is the integration of different security models adopted by different administrative domains in order to provide a unified and seamless access to users. This issue needs to be addressed in a scalable way without compromising on usability and flexibility.

Grids [15] try to solve the cross-domain security issue by providing new security infrastructures featured with single sign on (SSO) [23]. As a de-factor standard, Grid Security Infrastructure (GSI) [11] enables SSO access of nodes by introducing proxy certificates and delegation [28]. However, domain administrators and end users are often frustrated by the complicity of configuring and maintaining a new security infrastructure. From our experiences on deploying Grid software on China National Grid [6], there are several reasons hindering the promotion of Grid software in a production testbed:

- Grid software is generally provided as a heavy software stack which consists of many tightly coupled components, and any mis-configured one of them could stop the working of the whole software. Our survey shows that it generally takes three or four days for a professional engineer to make Grid software work smoothly, whereas deploying packages of current Linux distributions only takes minutes for unpacking and installation.

- Both administrators and end users need to maintain a new set of security data (e.g. credentials, mapping files) together with existing ones managed by traditional security frameworks such as NIS/YP [27], kerberos [26] and LDAP [12]. A typical example is that a new Grid user needs to be enrolled in grid-map files of each node besides the allocation of local accounts in each node. Another example is that proxy certificates need to be renewed periodically for long running jobs. Such additional effort could be partially saved by developing automatic routine services while at the risk of exposing new security holes.

- For HPC developers and end users, security-enabled Grid services nowadays exhibit new access interfaces to them which generally beyond the knowledge and skill of them. For example, programming with Grid services with WS-Security [24] is a nontrivial task for traditional Fortran developers. The user survey of NSF cyberinfrastructre [2] shows that only 18% of TeraGrid users have experienced with Grid tools in 2005, and 10% of users have used grid tools in production runs in 2006, while another interesting result is that `scp` is the most popular data management tool (up to 52% of users). Comparing to daily used Linux utilities to launch processes, the job execution on Grid nodes is generally a very complex workflow due to the layered security infrastructure working behind.

In large-scale VOs, the co-existing of different Grid platforms, such as Globus [15], Glite [20], Unicore [13] and OMII [3], together with several SSO solutions like Shibboleth [14] and Liberty alliance [1], make the VO-level resource sharing more complicate than ever. The problem of Grid interoperability has been proposed for several years whereas there are no simple solution for it due to the fact of lacking common accepted standards.

In this paper, we present XOS-SSH, a lightweight user-centric tool to support remote execution of jobs among heterogeneous nodes of VOs. XOS-SSH is a modified version of the widely used OpenSSH tool. It is based on several OS-level VO support mechanisms developed in XtreemOS project [21]. XOS-SSH adopts a pluggable framework that is capable of supporting different authentication schemes and making them transparent to shell users. The performance evaluation of XOS-SSH around NPB [9] benchmarks shows that our current implementation incurs trivial overhead comparing to the unmodified one.

The rest of the paper is organized as follows: we first analyze related technologies in section 2 and identify several key challenges in our design in section 3. Then, we address the detailed design issues in section 4 and carry out performance evaluations in section 5. Finally, we discuss related work in section 6 and conclude the paper in section 7.

2 Background

In this section, we present related techniques that motivate us to implement a lightweight remote execution tool for end users, which could deal with security challenges in consuming resources from several heterogeneous domains in a VO.

2.1 XtreemOS and VOs

XtreemOS [21] is a European project that aims to design, implement, evaluate and distribute an open source operating system, which supports Grid applications and runs on a range of platforms, from clusters to mobile devices. The goal is to provide an abstract interface to local resources, as a traditional OS does for a single computer. XtreemOS is based on the existing Linux OS. A set of system services, extending those found in Linux, provide users with the capabilities associated with Grid middleware. This native support means that XtreemOS will significantly ease the management and use of VOs without compromising on efficiency, flexibility, and backward compatibility. From the perspective of end users, they do not need to learn new interfaces and tools to use VOs as most tools will expose the standard UNIX commands familiar to users. Also, applications will not need to be re-factored to run on VOs as most XtreemOS APIs are POSIX-compliant. As part of the XtreemOS work, a ssh-based remote execution tool is developed to facilitate end users to launch jobs and move data among nodes of a VO, which aims to overcome many of the barriers to the use of VOs.

2.2 OS-level plug-in frameworks

In Linux/Unix-like distributions, Operating System (OS) is equipped with some pluggable frameworks and interfaces, which can be exploited by system developers to customize the OS behaviors. With these pluggable frameworks, customized modules can manually be plugged in system and interact with system software via standard interfaces, without any modification in OS codes. In XtreemOS, we have leveraged Pluggable Authentication Module (PAM) and Name Service Switch (NSS) to insert VO support functionalities into OS [21].

PAM, originally proposed by sun, is now widely used in Linux/Unix-like distributions. Its pluggable framework enables system administrators to choose authentication scheme for specific applications [25]. In large-scale VOs , SPs may belong to different domains which adopt different authentication schemes and security protocols. Specific PAM modules developed for each authentication model could be used by SPs without affecting applications. In addition, PAM modules could also be used to perform authorization and resource usage enforcement.

NSS is also a pluggable framework for name resolving of Linux system objects such as users, groups and hosts. In NSS, query against traditional Unix file-based information stores (e.g. `/etc/passwd` and `/etc/group`) could be substituted with querying other databases such as NIS+, LDAP and customized NSS modules [5] [29]. NSS APIs are standard `libc` calls and NSS modules could be configured outside the application (by `/etc/nsswitch.conf`). With NSS, we can develop customized modules to process user related information (such as resolving Distinguished Name of a Grid user to a local account) while making this transparent to legacy applications.

2.3 OpenSSH-based modification

In our design, we modify the OpenSSH [4] to support secure communication among nodes with different security models. The modification based on OpenSSH can benefit from several aspects.

Firstly, OpenSSH is a standard component in Linux/Unix-like OS nowadays. It is extensively used by end users for secure login and data transmission (e.g. `scp`, `sftp`) among local and remote nodes. Many projects adopt it as a standard secure channel for communication. For example, parallel applications built upon MPI depend on OpenSSH to launch processes on trusted nodes. Extending OpenSSH to support VOs could greatly improve the usability for end users and provide transparency to traditional parallel applications.

Secondly, OpenSSH is open-source and it has a well-designed crypto library as well as an extensible code skeleton. Extensions could be added into current OpenSSH without affecting its original functionalities. The extended OpenSSH can turn back to the original authentication method if customized extensions fail. Also, system administrators could determine whether the extensions are enabled.

Lastly, the latest OpenSSH release was implemented as a PAM-aware application [25], which means that it could be configured to use customized PAM modules to do authentication against VO users.

3 Challenges

Several challenges still remain when developing a secure remote execution tool among heterogeneous nodes in a VO.

The first challenge is the design of efficient protocol to support multiple authentication models. In a large-scale networked environment, the communication channel between heterogeneous nodes needs to securely carry sufficient information to prove users' identities and attributes. The protocol of data transmission is to be designed as flexible as possible to accommodate multiple authenticate models while keeping the packet size small, as redundant data could reduce the efficiency of authentication processing codes.

The second is the support of multiple authentication models at server-side. Additional work needs to be done to make OpenSSH work seamlessly with PAM and NSS modules. Various client credentials need to be securely passed from client-side to server-side and then to PAM modules. As PAM framework allows multiple modules to work together for a PAM-aware application, there should be a negotiation process to determine which set of PAM modules are put into action for a specific authentication model.

The third is the scalable support for large amount of VO users. This scalability issue lies in two aspects: a) concurrent accessing of the same SP node by numerous VO users needs to be differentiated and isolated; b) the management of user accounts in a node needs to be performed in a scalable manner without compromising on security in terms of access control and accounting. Traditionally, system administrator may allocate a local account for each VO user in each SP node to guarantee the isolation among them. However, this could be a nightmare for administrators when there are large amount of users in VOs where memberships of users and access rules of nodes are dynamically changing. To achieve a scalable node-level VO support , it is natural to allow VO users to access nodes without pre-allocation of local accounts.

4 Design and Implementation

In this section, we will present our design in details. We introduce the overall architecture first, then we explain in details how we address challenges mentioned above.

4.1 Overview

The overall architecture upon which current XOS-SSH works is shown in Figure 1. A VO user obtains an X.509 certificate from a VO manager service (e.g. from the Credential Distribution Authority (CDA)[30] or from Globus SimpleCA [11]) and presents it to a PAM-aware application running in a VO node. This PAM-aware application checks XOS-Cert for validity and whether the requested account already exists. For a VO user who has been granted to access but with no corresponding local account, an Account Mapping Service (AMS) maps the user's identity to a dynamically created *virtual* account in local nodes (discussed later). After the VO user is authenticated, the mapping information of VO-level users and groups could be obtained by the NSS extensions via standard `libc` calls. The AMS guarantees that only authorized processes can obtain this information.

As a PAM-aware application, OpenSSH is extended to use the new developed PAM and NSS modules. It is worth mention that the current architecture is not bundled with a specific security model (i.e. not limited in fitting with default VO model of XtreemOS).

4.2 Protocol

Currently, standard OpenSSH do not support authentication methods based upon X509-based certificate. XOS-SSH extends the OpenSSH by introducing a customized

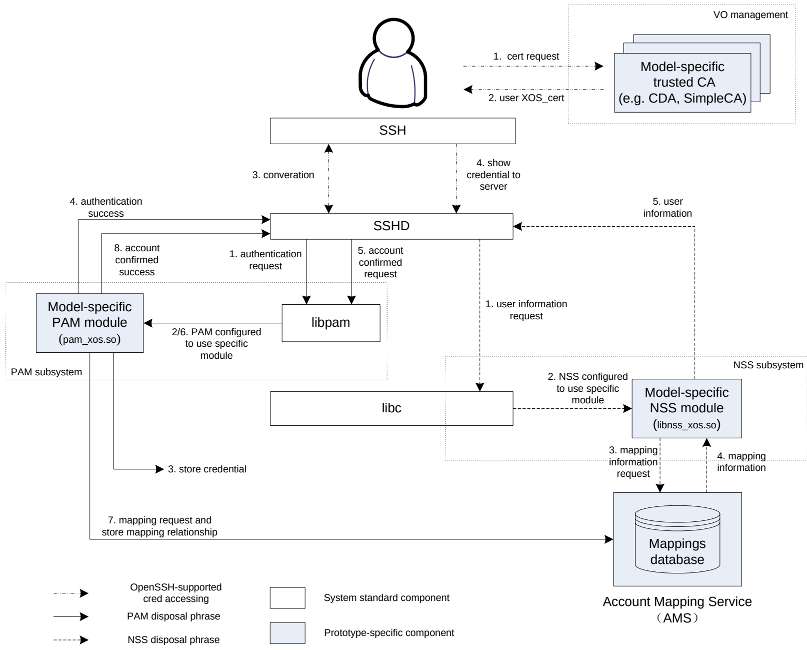


Figure 1: Overall architecture

packet format and communication protocol. The packet format is illustrated as Figure 2. A packet is composed of SSH header (HDR) followed by a series of segments, each of which representing a user credential within a specific authentication model. Data fields of each segment are explained as follows.

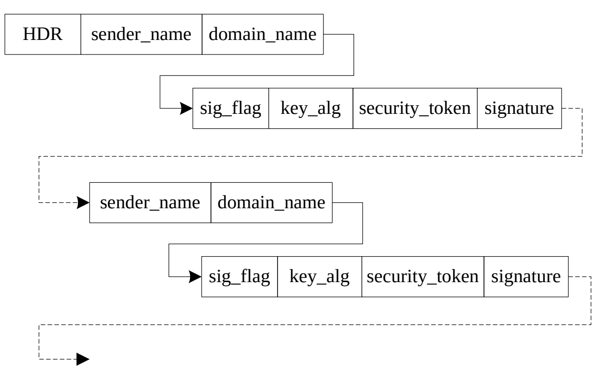


Figure 2: Packet format in XOS-SSH

- `sender_name`: The name of the sender (e.g. DN or usernmae), which is to be checked with security token by sever-side.

- `domain_name`: The field is holding the information to tell server-side which category of authentication models the user certificate is belonged to, so that PAM can choose corresponding module for certificate verification.

- `security_token`: security related data to prove the user's identity and attributes (e.g. password, proxy certificate, attribute certificate).

- `sig_flag`, `key_alg` and `signature`: Each segment is attached with a signature which is signed with sender's private key. The `key_alg` denotes the signature algorithm and the `sig_flag` is marked with whether the segment is signed.

The packet format is flexible, because several segments containing same fields can be glued together and each segment may belong to different authentication models. The length of each segment is variable because not all the fields are necessary in some authentication models. (e.g. MyProxy [18]).

Although communication data are encrypted by OpenSSH, it can only guarantee that data are from the right peer rather than the right user. For example, malicious users may capture other users' proxies and then send them to SPs via legal clients. The server can not find this potential attack if nothing is done to defend the attacker. Hence, customized protocol need to be defined to consolidate OpenSSH, by signing message packet with user private key. To the GSI model for example, the customized protocol requires client to pack sender's DN and proxy into same segment, and then signs the segment with private key before transmission. In server-side, public key would be fetched from proxy to verify signature. Next, sender's DN would be checked whether it matches the DN of proxy. Further, PAM module plugged in server would verify user proxy. The procedure collaborating with PAM authentication is to guarantee the packet is from right user.

4.3 Pluggable modules for specific authentication models

As mentioned above, pluggable modules are utilized to cope with heterogeneity of authentication models. The key issue is to specify interaction agreement between PAM and SSH server. In our implementation, model-specific PAM conversation handlers are developed to pass private identity information from SSH server to the PAM modules. The interaction agreement defined in handler is specific to given authentication model. Each module has its own handler, to get information from application. Due to feature of PAM-aware application, SSH server can be configured to use a chain of several PAM modules. Each PAM module in the chain first requests domain information from packet (defined in `domain_name`), and determine whether it is suitable to process user credential. If a PAM module takes charge of credential process, the rest information will be passed by its conversation handler, according to model-specific interaction agreement. Generally, each PAM module only processes those credentials matching given authentication model. If there are several modules matching the same authentication model, priority will follow the chain.

With the help of PAM framework, customized modules could be easily developed to fit with new authentication models. Currently, we have developed two specific PAM modules, `pam_xos.so` and `pam_gsi.so`, for XtreemOS model and GSI-like model; the third one for MyProxy is ongoing. The `pam_xos.so` is developed to authenticate users with certificate issued by CDA in XtreemOS project [30]; the other one, `pam_gsi.so`, is used to authenticate those with certificates observed RFC3820 [17] used in Globus and VOMS [8].

4.4 User mapping

As discussed in section 3, when there are large number of VO users with all kinds of authentication credentials concurrently access resources or execute remote commands in the same SP node, it is critical to guarantee the isolation of users in terms of resource usage, security and fail-recover in a scalable manner. We first analyze current OS mechanisms to address the issue and then propose a "*virtual account*" mapping mechanism to address the scalable accessing and isolation.

In current OS, there exist two items named `uid` and `gid`. Users are identified and isolated by their `uid`s which constructs a container including all resources (process, file, memory, etc) labeled with given `uid`. And, they can share their resources via `gid`. Currently, permission checking and file access control in OS are performed based on `uid` and `gid`. Without the modification of kernel, we could make use of existing mechanisms to realize the isolation of VO users if each VO user can be mapped onto local `uid` and `gid`s.

Unlike the Globus grid-mapfile approach [11], we are not going to prepare several accounts in each SP server for user accessing. Because all user or group information are requested via `libc` interfaces and further returned by NSS subsystem, a specific NSS module and an AMS are developed to provide *virtual account* for each VO user. The *virtual account* means they are accounts owning `uid` and `gid`, but they are not stored in system databases (`/etc/passwd` and `/etc/group`), and they are agnositic to local applications but recognized by kernel. Figure 3 depicts the mapping mechanism of virtual accounts.

When an application requests user information, the request will first be filtered in a specific NSS module (`libnss_xos.so` in figure 3, for example) to determine the type of request. If the request is not of VO user, `libnss_xos.so` will free the request. If so, `libnss_xos.so` will ask AMS for the information. The information will be encapsuled as `struct passwd`, which can be used as normal data structure. Hence, it is not necessary to have the pre-allocated accounts in SP server.

The total number of pre-allocated accounts is hard to cater for concurrent accessing of large-scale users, so it is not easy to guarantee isolation in this approach. Also, the approach aggravates the administrative burden because pre-allocated accounts would be added in system databases (via `useradd`).Compared with physical accounts, scalability support of virtual account lies in that a) user information is not stored in system databases but in separate databases built with BerkelyDB [22], and b) the allocation of `uid` and `gid` can be expanded to wider boundary, from 0 to $2^{32}-1$ (the length of integer word).

Since each VO user has a local mapped account (either physical account or virtual account), OS can isolate and control their behaviors like conversational local users. Resource control, security control and fault isolation can be maintained via original mechanisms. Furthermore, to enhance the fault isolation and alleviate the burden of garbage collection, files created by VO users are stored in global filesystem such as NFS, but local temporary files will be cleared out when they log out.

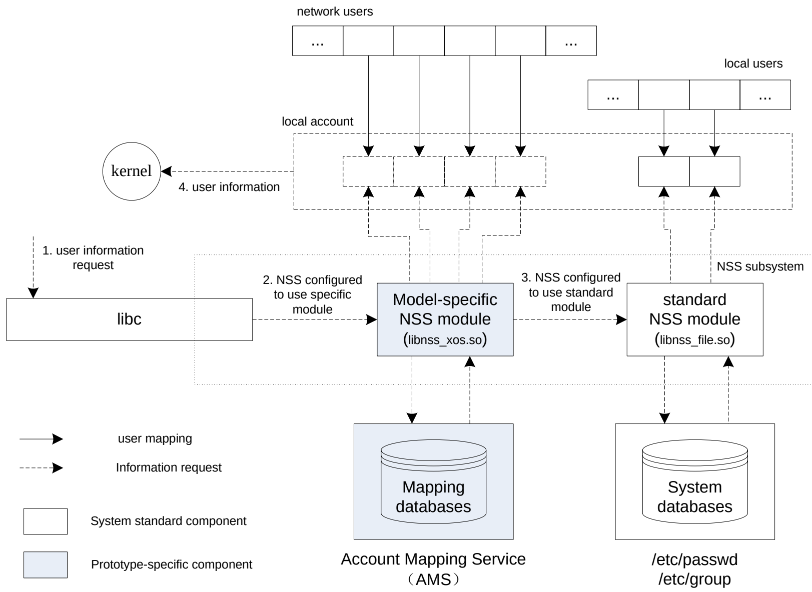


Figure 3: using mapping

5 Performance Evaluation

In this section we present our performance evaluations, which we design to address various important metrics of remote execution in distributed environment. We first evaluate the basic performance with respect to response time of authentication, average transfer rate and network connections, comparing with OpenSSH. Finally, we examine the impact of our prototype on hosted parallel application using NAS Parallel Benchmarks (NPB) [9].

5.1 Experimental Environment Setup

We implement our prototype based on OpenSSH-4.5pl, and name it with XOS-SSH because it is developed for XtreemOS project [21]. We compare our prototype with OpenSSH (version 4.5) on a simulated distributed environment, which contains four virtual machine nodes in a physical server. The simulated environment is built on DELL PowerEdge 1950 with Intel Xeon L5310 1.60GHz CPU, 3.25G memory, and 80G disk. We construct four virtual machine nodes with VMware server (version 1.0.4) and each virtual machine node is equipped with single 1.6 GHz CPU, 388M memory and 8.0G disk. CentOS4.3 is running in each virtual machine node and is connected with Ethernet. We deploy our prototype on this simulated distributed environment and then carry out our experiments.

5.2 Basic Performance Evaluation

The basic performance is evaluated with simple and easygoing methodology, which is widely used in OpenSSH developer community. Although the methodology is not imprecise, it can illustrate the difference between XOS-SSH and OpenSSH in authentication and transfer traffic. To compare the authentication time, we use the command ”`time ssh IPaddress /bin/true`” in console.

As shown in Figure 4, the time in user mode is same, but the time in kernel mode is longer for XOS-SSH than that for OpenSSH. As mentioned above, XOS-SSH has to deal with the complex authentication models, so it needs more time in kernel mode to deal with certificate verification, not just password.

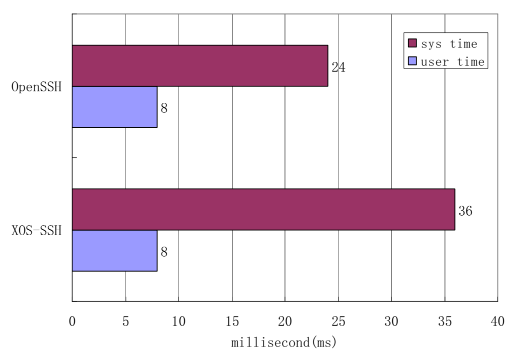


Figure 4: authentication time

For evaluation of network connection, the next experiment is set to transfer a 100M file from client to server by `scp`. With `wireshark` [7], a traffic analyzer, we show the statistics information in Table 1.

Although packet format and protocol have been used to transmit more content in XOS-SSH, there is no much influence on transmitting big blocks of data with `scp`. As illustrated by average package size (Avg. package size), XOS-SSH packs more content in each packet. This also influences the total number of transmitted packages and the average speed of package transmission, as shown in Table 1. Both of them are decreased in XOS-SSH, but transmission speed reduces too. The tradeoff shows trivial overhead comparing with OpenSSH.

5.3 Evaluation of Parrallel Applications on XOS-SSH

The OpenSSH is widely used to provide secure communication for HPC parallel applications, so we also evaluate the actual impact of XOS-SSH on hosted parallel HPC applications. The experiment is constructed to use NAS Parallel Benchmarks (NPB) [9], derived from the computing kernels common on Computational Fluid Dynamics (CFD) applications.

The experiment uses MPICH2 to construct the HPC experiment, which is a popular MPI tools to build MPI applications. Below the MPICH2, both OpenSSH and XOS-SSH provide a basic infrastructure for secure communication, which connects two heterogeneous trusted domains together. Each domain contains two nodes. One of trusted domain is configured as XtreemOS authentication model, and the other is used GSI authentication model. The difference between two authentication models lies in delegation fashion: XtreemOS model can not provide proxy delegation (at least, current implementation can not support), whereas GSI model can. Therefore, we design a *pseudo proxy* to smooth the difference. The *pseudo proxy* mechanism helps XtreemOS users delegate a proxy for remote authentication, which is short-lifetime proxy only containing the address of client and a temporary password, which can be used only once for server to seek client and do authentication similarly as MyProxy fashion [18].

Figure 5 and 6 show the results of NPB running on XOS-SSH and OpenSSH. NPB benchmarks of class A and class B was selected in experiment because class A is proposed for workstation and class B is for small parallel systems constructed by high-end workstations [10]. All the items in NPB benchmarks are tested twice running on four nodes and the avenge experimental value serves as our experimental data.

As shown in Figure 5, XOS-SSH has lower Millions of Operations per second (Mop/s) than OpenSSH in small size of Class A benchmarks. However, when computational size is expended to Class B, XOS-SSH has more improvement, especially in some benchmarks such as LU, SP, and BT, which is more related to applications. In Class A benchmarks, the average Mop/s of XOS-SSH is 724.255, comparing to 830.405 of OpenSSH. But in Class B, XOS-SSH is higher than OpenSSH in average Mop/s, with the ratio of 655.75:630.61.

The compassion of consumed time also illustrates the trivial overhead between XOS-SSH and OpenSSH, as shown in Figure 6. In Class A benchmarks, the average consumed time of XOS-SSH is 41.6525 while OpenSSH is 37.5488. However, in Class B benchmarks, the ratio of average consumed time has reduced, with 4499.858 of XOS-SSH and 4230.563 of OpenSSH. In Class B of Figure 7, we have not provided the time of FT, because total consumed time of FT benchmark is higher than any other benchmark (34452.87:32234.22, in seconds), so that more comparison details can be shown in Figure.

6 Related works

In this paper, we discuss the design of lightweight remote execution tool based on OpenSSH and OS-level extensions. Other similiar SSO support tools include those implemented in the Grid Security Infrastructure (GSI), GSI-SSH [16][28] and MyProxy [18]. The GSI-SSH is also a patched version of OpenSSH to authenticate users within the GSI framework. MyProxy [18] provides an online repository for storing proxy certificates for users, which are accessed by providing normal username/password pair. They are not designed as a flexible framework to support other kinds of authentication models.

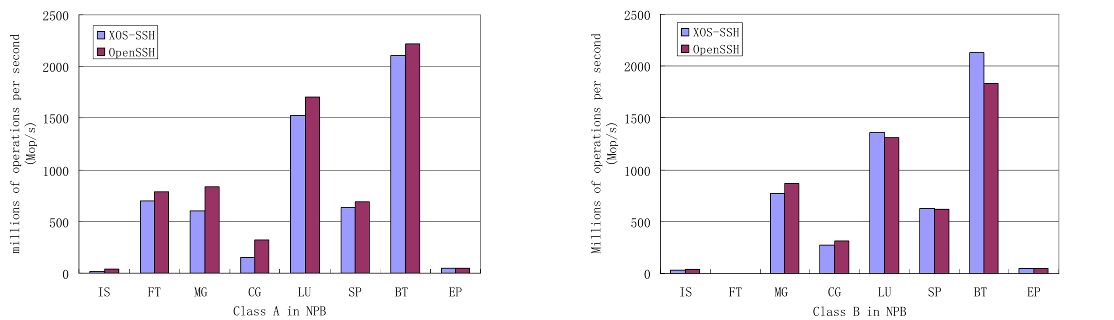
For security issues, maintaining a new set of security data has caused new work burden to both administrators and end users. Traditional security frameworks such as NIS/YP [27], Kerberos [26] and LDAP [12] provide centralized identity management, but at the risk of single point of failure. The mechanisms of Globus mapfile [11] has limitation of scalability when deploying in large-scale Grid applications. Some works such as [19] provide plug-ins for GSI to improve the limitation of grid-mapfile, however the Globus-dependent plug-ins can not be applied to other authentication models. In our design, the virtual account mapping mechanism built upon NSS extensions addressed the scalability issue while providing support for legacy applications.

7 Conclusion and Future work

In this paper, we present several issues hindering the utilization of current Grid software and identify challenges to realize remote execution cross heterogeneous security

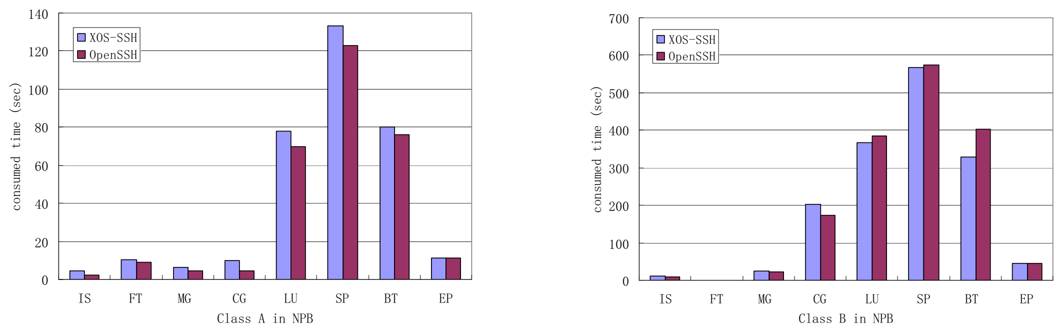
Traffic statistics	**OpenSSH**	**XOS-SSH**
Between first and last package	27.328 sec	23.843 sec
Packages	112967	79365
Avg. packages/sec	4133.815	3328.692
Avg. package size	996.000 bytes	997.000 bytes
Bytes	112534474	79205972
Avg. bytes/sec	4117988.014	3322022.483
Avg. MBit/sec	32.944	26.576

Table 1: Traffic statistics of OpenSSH and XOS-SSH



(a) NPB benchmarks of Class A to evaluate execution speed (b) NPB benchmarks of Class B to evaluate execution speed

Figure 5: Comparison of millions of operations per second (Mop/s)



(a) Consumed time of running Class A benchmarks (b) Consumed time of running Class B benchmarks

Figure 6: Comparison of consumed time (seconds)

domains in VOs. We developed XOS-SSH, a lightweight user-centric tool by patching OpenSSH to use OS extensions developed in XtreemOS project. With the help of pluggable framework, specific authentication models could be processed by XOS-SSH in a unified way. Our design addresses several challenges by fully exploiting existing mechanisms in OS, including flexible protocol design to transfer authentication information, model-specific PAM modules and scalable user mapping Our design significantly smoothes the gap between administrative domains built with specific authentication models, in term of seamless remote execution. We evaluate our solution through a set of experiments to measure its impact on parallel applications. The experimental data of performance show that our prototype incurs trivial overhead comparing to standard OpenSSH. Although our prototype can easily integrate heterogeneous domains together, the experiments in this paper have not covered the measurement of efficiency when there are huge amount of domains. We will continue to exploit existing OS extensions to support resource sharing on VOs in a secure, flexible and scalable manner.

8 Acknowledgments

This research is supported by XtreemOS project under European Commision FP6 Contract (No. 033576). We would like to thank Luis Pablo Prieto, Erich Focht, Yvon Jegou and Zhiwei Xu for their support and insightful discussion.

References

[1] Liberty Alliance. http://www.projectliberty.org/.

[2] NCSA/SDSC Cyberinfrastructure User Survey 2005&2006. http://www.ci-partnership.org/survey/.

[3] Open middleware infrastructure institute (omii). http://www.omii.ac.uk/.

[4] Openssh. http://www.openssh.org/.

[5] System databases and name service switch. http://www.gnu.org/software/libc/manual/htmL node/Name-Service-Switch.html.

[6] The China National Grid (CNGrid) Project. http://i.cs.hku.hk/ clwang/grid/CNGrid.html.

[7] Wireshark. http://www.wireshark.org/.

[8] ALFIERIA, R., CECCHINIB, R., CIASCHINIC, V., DELLAGNELLOD, L., FROHNERE, A., LOIXRENTEYF, K., AND SPATAROG, F. From gridmap-file to VOMS: managing authorization in a Grid environment. *Future Generation Computer Systems 2005*, 21 (2005), 549–558.

[9] BAILEY, D., BARSZCZ, E., BARTON, J., BROWNING, D., R.L., CARTER, DAGUM, L., FATOOHI, R., FREDERICKSON, P., LASINSKI, T., SCHREIBER, R., SIMON, H., VENKATAKRISHNAN, V., AND WEERATUNGA, S. The Nas Parallel Benchmarks. *International Journal of High Performance Computing Applications 5*, 3 (1991), 63–73.

[10] BAILEY, D., HARRIS, T., SAPHIR, W., DER WIJNGAART, R. V., WOO, A., AND YARROW, M. The NAS Parallel Benchmarks 2.0. Tech. rep., NAS Technical Report NAS- 95-020, 1995.

[11] BUTLER, R., ENGERT, D., FOSTER, I., KESSELMAN, C., TUECKE, S., VOLMER, J., AND WELCH., V. A National-Scale Authentication Infrastructure. *IEEE Computer 33*, 12 (2000), 60–66.

[12] CARTER, G. *LDAP system administration*. O'Reilly & Associates, Inc, 2003.

[13] E., D. W., AND S., D. F. UNICORE: A Grid Computing Environment. In *Lecture Notes in Computer Science* (Springer Berlin / Heidelberg, 2001), Springer, pp. 81–92.

[14] ERDOS, M., AND CANTOR, S. Shibboleth-Architecture DRAFT v0.5. http://shibboleth.internet2.edu/docs/draft-internet2-shibboleth-architecutre-05.pdf.

[15] FOSTER, I., KESSELMAN, C., AND TUECKE, S. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications 15*, 3 (2001), 200–222.

[16] GSI-ENABLED OPENSSH. http://grid.ncsa.uiuc.edu/ssh/.

[17] HOUSLEY R., POLK W., F. W., AND SOLO, D. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile [RFC 3280], 2002.

[18] J, N., AND V, T. S. W. An online credential repository for the grid: MyProxy. In *10th IEEE International Symposium on High Performance Distributed Computing (HPDC-10 '01)* (2001), pp. 104–112.

[19] JANKOWSKI, M., WOLNIEWICZ, P., AND MEYER, N. Virtual User System for Globus Based Grids. In *Proceedings of Cracow Grid Workshop (Cracow'04)* (2005), pp. 316–322.

[20] LAURE, E., AND HEMMER, F. Middleware for the Next Generation Grid Infrastructure. In *Proceedings of Computing in High Energy and Nuclear Physics (CHEP)* (2004).

[21] MORIN, C. Xtreemos: a grid operating system making your computer ready for participating in virtual organizations. In *Proceedings of ISORC'07* (July 2007), vol. 5, pp. 347–368.

[22] OLSON, M. A., BOSTIC, K., AND SELTZER, M. Berkeley db. In *Proceedings of 1999 USENIX Annual Technical Conference* (1999), pp. 183–192.

[23] PASHALIDIS, A., AND MITCHELL, C. A taxonomy of single sign-on systems. In *Proc. ACISP'03* (2003), pp. 249–257.

[24] ROSENBERG, J., AND REMY, D. *Securing Web Services with WS-Security: Demystifying WS-Security, WS-Policy, SAML, XML Signature, and XML Encryption*. O'Reilly & Associates, Inc, 2004.

[25] SAMAR, V. Unified login with pluggable authentication modules (PAM). *Proceedings of the 3rd ACM conference on Computer and communications security* (1996), 1–10.

[26] STEINER, J., NEUMAN, C., AND SCHILLER, J. I. Kerberos: An authentication service for open network systems. In *Proc. USENIX Winter Conf* (Feb 1988), pp. 192–202.

[27] STERN, H., EISLER, M., AND LABIAGA, R. *Managing NFS and NIS*. O'Reilly & Associates, Inc, 2001.

[28] WELCH1, V., FOSTER, I., KESSELMAN, C., MULMO, O., PEARLMAN, L., TUECKE, S., GAWOR, J., MEDER, S., AND SIEBENLIST, F. X.509 Proxy Certificates for Dynamic Delegation. In *3rd Annual PKI R&D Workshop* (2004).

[29] XTREEMOS CONSORTIUM. D2.1.2 node-level VO support specification. XtreemOS deliverable, November 2007.

[30] XTREEMOS CONSORTIUM. D3.5.4: Second Specification of Security Services. XtreemOS deliverable, November 2007.