

Towards cluster survivability

Chokchai Leangsuksun¹, Anand Tikotekar¹

Stephan L. Scott², Makan Pourzandi³, and Ibrahim Haddad⁴

Louisiana Tech University¹
Oak Ridge National Laboratory²
Open Systems Lab, Ericsson Research Canada³
Open Source Development Labs⁴

box@latech.edu¹
aat007@latech.edu¹
scottsl@ornl.gov²
makan.pourzandi@ericsson.com³
ibrahim@osdl.org⁴

Abstract. This paper propounds an investigation, a feasibility study, and performance benchmarking of vital management elements for critical enterprise and HPC infrastructure. We propose concepts of integrating high availability cluster mechanism with a secure cluster infrastructure. Our proposed architecture incorporates the Distributed Security Infrastructure (DSI) framework, an open source project providing secure infrastructure for carrier grade clusters, and HA-OSCAR, an open source cluster framework that meets the Reliability, Availability, Serviceability (RAS) needs. The result is a cluster infrastructure that is compliant with the Reliability, Availability, Serviceability and Security (RASS) principles. We conducted an initial feasibility study and experiment to gauge issues and the degree of success in the implementation of our proposed RASS framework. We verified the integration of HA-OSCAR release 1.0 and DSI release 0.3. Although there was a minimal performance overhead, having “RASS” in mission critical settings by far outweighs the performance impact. We plan to further our proof-of-concept architecture to suit the required needs on the production environments.¹

1. Introduction

The cluster wide security attacks in spring 2004 [6] [7] [8] only serve to highlight the vulnerability of the systems in the cyber era. Although, there are many existing security solutions, most previous works are either incoherent or unsuitable for the

¹ Research supported by Center for Entrepreneurship and Information Technology, Louisiana Tech University.

² Research supported by the mathematics, Information and Computational Sciences Office, Office of Advanced Scientific Computing Research, Office of science, U.S. Department of Energy, under contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.

2 Box Leangsuksun, Anand Tikotekar, Stephan L. Scott, Makan Pourzandi, and Ibrahim Haddad

downtime-sensitive cluster computing environment. The most common approach is to package together several different security solutions. The obvious difficulty stems from the issues of integration, incoherence and management of these solutions. Moreover, upgrade and maintenance of these diverse security packages are far from easy. Consequently, this leads to interoperability problems between different security mechanisms. Secondly, current security implementations are based on user login/privileges (e.g. SSH) and do not support a fine-grained security approach. Particularly, they do not support authentication and authorization checks across interactions between two processes belonging to users on different machines. Moreover, in scenarios where a few users run the same application for a long time, the above approach will grant the same security privileges to all the processes since it is a user level security scheme.

Clusters are eminently popular building blocks for infrastructures in an environment such as HPC research, enterprise, telecomm, and cyber worlds. The challenges lie in a wide spectrum of concerns such as manageability, performance, reliability, survivability and robustness to name a few. In order to survive from attacks, cope with outages and sustain quality of service at expected levels, one must altogether ensure Reliability, Availability, Serviceability and Security (RASS). Clearly, such systems must exhibit the ability to support a complete RASS capability in order to survive and sustain providing services at the expected level. The combination of the recently released HA-OSCAR software stack and the DSI framework is one such approach to address these issues.

HA-OSCAR [10][14][15] is a highly reliable cluster software stack that currently provides a Reliability, Availability, and Serviceability (RAS) framework for Linux clusters. It supports HA and fault tolerance through “self-awareness approaches” including self-installation, self-configuration and self-healing. On the other hand, DSI provides a distributed security framework for authentication, communication integrity, access control, and auditing. DSI [22][23] extends the concept of Mandatory Access Control (MAC) to the Linux cluster environment. Together HA-OSCAR and DSI provide strong, cohesive, and highly survivable and distributed control framework with centralized RASS management.

The next section describes related work in the area of RAS and Security. Section 3 describes the HA-OSCAR (RAS) architecture while Section 4 explains DSI architecture. Section 5 contains details of our combined RASS model with HA-OSCAR and DSI towards survivability. It also discusses the proof of concept and integration framework. Section 6 describes the integration challenges that were encountered and a possible set of solutions. Section 7 reports our performance impact related findings of the experiments pertaining to the integration of HA-OSCAR and DSI. Finally, we conclude our study and discuss our findings in Section 9.

2. Related Work

Existing cluster system managements [21] [24] provide straightforward system installation and a set of commands to perform system-wide operations with minimal concerns toward RAS. Patterson and Fox [2] [5] proposed recover-oriented comput-

ing (ROC). They focus on recovery-based approaches, e.g. undo-recovery, to minimize Mean Time To Restart (MTTR), which is one critical factor to maximize availability. HA-OSCAR [10] shares the same goal with additional focus more towards the cluster environment. There are a number of related works on HA clustering: for example, Lifekeeper [27], Kimberlite [13], Linux Failsafe [4], Mission Critical Linux [20] and HP Serviceguard [9]. Most of these solutions, if not all, lack a coherent security framework, which is essential for system survivability. Current security approaches leverage existing solutions that may not be suitable for the cluster environment, creating interoperability challenges. Moreover, security provisioning, integrating with other services requires colossal efforts to manage and operate the system in an un-trusted and vulnerable world.

Most existing security approaches aim to protect individual machines and provide incomplete solutions for the cluster environment, leaving the burden of cluster security issues to the human. William Yurcik et al [29] suggested a technique to manage cluster security based on threat models and monitoring. They suggest a method to address overall security through process monitoring, network port scanning, and traffic analysis. Matti and Ugarte [11] introduced the use of redundancy in software to avoid a single point of breach. This technique employs more than one algorithm to secure some of the system elements. Kai and Muralidharan [12] developed micro firewalls for dynamic network security and intrusion detection. Lee, Kim et al [16] described the DDOS attack and the distributed nature of it. They developed a scheme whereby they can identify its source. National Security Agency SE Linux [25] is the most known MAC implementation.

Nevertheless, most of the above solutions only focus on the individual node, ignoring the distributed nature of a cluster and most importantly RAS attributes that clearly affect system survivability.

3. HA-OSCAR Architecture

HA-OSCAR builds on top of the Open Source Cluster and Application Resources (OSCAR) cluster software stack, a proven open source HPC clustering solution. However, OSCAR and any typical Beowulf cluster architectures suffer from a single point of failure (SPOF). Many known techniques can be used to eliminate SPOFs and to ensure service availability in face of hardware or software failure. Such techniques include the active/active, active/hot standby, active/cold standby.

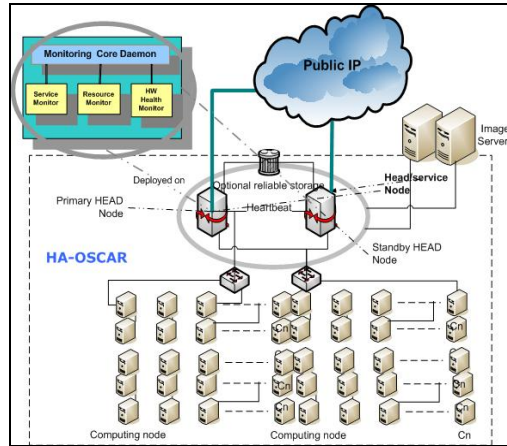


Fig. 1: HA-OSCAR architecture

Fig. 1 describes the HA-OSCAR architecture. The beta 1.0 release supports the active/hot standby approach to alleviate the SPOF thereby increasing system and service availability. An enhancement is currently ongoing to support the active/active scenario. HA-OSCAR employs self-installation and self-configuration approaches using an operating system (OS) image capture and configuration tool called SystemImager. A typical Beowulf cluster requires persistence of certain critical services like DHCP (Dynamic Host Configuration Protocol), NTP (network Time Protocol), TFTP (Trivial File Transfer Protocol), NFS (Network File System), and job queue schedulers, etc. Any failure in these services could potentially bring the cluster to an unusable state, thus affecting the services provided to its end users. Therefore, it is critical to adopt an intelligent approach that can help keep services available while simultaneously minimizing the time to fix the failure. HA-OSCAR addresses this challenge with Adaptive Self-Healing (ASH) technique. ASH MON daemon [14] [15] monitors the service availability and health of the system at tunable intervals. It triggers alarms upon detecting failure, and launches a recovery procedure to fix the problem encountered. HA-OSCAR performs an automatic failover when it detects a failure in the head node. It then drops the network interfaces of the primary server, the standby server then clones the primary server and eventually, in about 4-seconds, takes over the role of the primary server.

4. DSI Architecture

The telecommunication industry traditionally employs cluster systems to exploit high availability and scalability characteristics to satisfy its stringent requirements. Carrier grade security [3] requires a solution that must be coherent and handle the distributed nature of cluster systems. Moreover, carrier class clusters impose tight re-

strictions on performance and response time. A high resource consumption solution is therefore unacceptable in carrier grade systems.

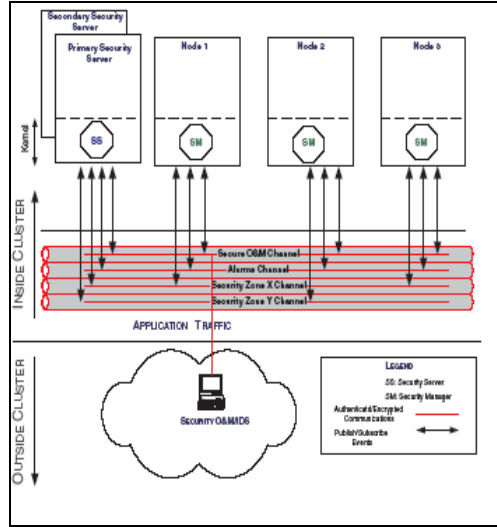


Fig. 2: Distributed architecture of DSI

The Distributed Security Infrastructure (DSI) was developed to alleviate the above problem. DSI is a distributed, coherent framework that addresses cluster security services such as confidentiality, integrity, access control and auditing with a process level granularity. Moreover, DSI aims at simplifying a cluster's security administration. Fig. 2 illustrates the DSI architecture. DSI consists of two primary components, namely a Security Server (SS) and Security Managers (SM). SS resides on the head node (also called master node). Each server node consists of Security Managers (SM) that provides a process level authentication and access control. The security service is duplicated for redundancy purposes. DSI supports an event driven Secure Communication Channel (SCC), which is used for the communication between SS and SMs. The SCC does not present itself as a single point of failure and has a mechanism to filter events; as a result, it consumes less bandwidth and less time before discarding irrelevant information. This process results in a reduction of the system load.

DSI shares many similarities with SELinux [26][17] [25] with regard to Mandatory Access Control (MAC) [18]. The Flask architecture [26] was developed to demonstrate the general architecture of MAC. DSI implements and extends MAC to the entire cluster, with its Distributed Security Model (DSM). DSM is based in the kernel (following the concept of the Linux Security Module (LSM) [28] for policy enforcement logic. It effectively uses the concept of separation of policy decision logic and policy enforcement logic. This shuts down the option of bypassing the security since the security is enforced at a kernel level. The underlying security policy mechanism in DSI is called Distributed Security Policy (DSP). It is a policy written in XML as op-

posed to having many configuration files written in a declarative language, which is the model adopted by SELinux. XML provides extensibility, flexibility and has many security mechanisms built in. The security policy in DSI [1] consists of classes similar to object classes in SELinux and rules based on source and target object Security context IDs similar to type pairs of MAC model in SELinux. The goal of DSP is to define a unique, homogeneous and cluster wide security policy to be enforced over all nodes. To further exemplify the role of DSP in Access control mechanism/service, consider the following remote access control scenario.

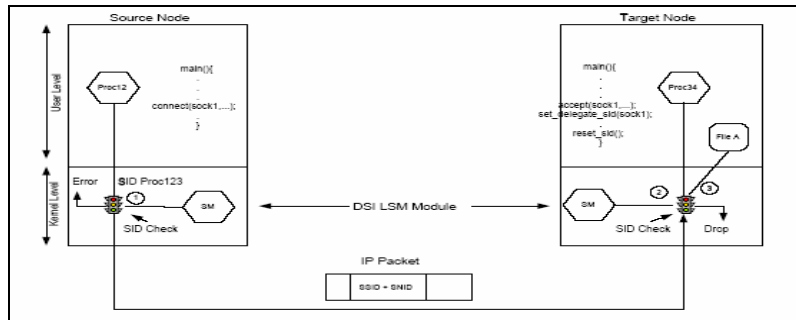


Fig. 3: Checks performed between two nodes to verify access control

As illustrated in Fig. 3, a process on source node (subject) is accessing a resource (object) on the target node. The access rights are the functions of Source node Id (SNID1), Subject security ID (SID1) the Target node Id (SNID2) and Object security ID (SID2). DSI performs various validations by checking whether the subject has rights to access local resources such as network resources. If the subject has access rights, the source node ID and subject security ID are added as an option to the IP packet and sent over the network. Similar validations are performed at the receiver's end after extracting the security information from the IP packet with regard to the network resources. Finally, DSI authenticates whether the subject is allowed to access the resource (object). These security verifications are based on the rules in Distributed Security Policy (DSP).

The DSI security management supports various levels of security policies. For example, when some external IP is accessed due to a back door on some software used, this information is sent to the security server. At this time, the administrator can use this information to modify DSP to forbid access to all resources in the cluster for the malicious or bogus software. This new DSP is then propagated to security managers through SCC and is enforced by the DSM.

5. HA-OSCAR and DSI: Proof-of-concept Integration

Our primary objective is to investigate and develop a cohesive and well integrated Reliability, Availability, Serviceability and Security (RASS) management framework for Linux-based cluster critical infrastructure. Both HA-OSCAR and DSI projects

have exhibited their strengths towards Carrier Grade Linux (CGL) survivability goals. We conducted a feasibility study and experiment on a combined HA-OSCAR and DSI framework. Fig. 4 illustrates the proposed integration architecture. In our framework, DSI provides security and access control services at a process granularity level. On the other hand, HA-OSCAR is responsible for service HA, cluster health monitoring and self-healing capabilities. In addition, it also supports higher serviceability through ease of cluster installation and configuration via the OSCAR framework.

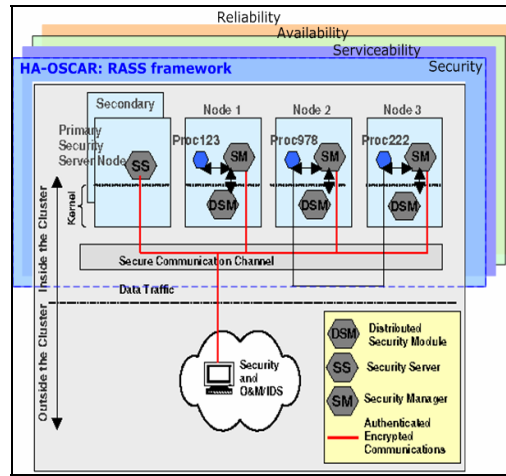


Fig. 4: RASS Framework

There were three staging process in our experiment. First, the Linux kernel was patched with Linux Security Modules (LSM) [28] security hooks on all nodes. After DSI installation, we proceed with building a cluster with OSCAR and then following this with the HA-OSCAR tool sets. Finally, HA-OSCAR monitoring services were applied to DSI Security Server (SS) and Security Managers (SMs). In the dual head-node architecture, the SS resides on both head-nodes from which one instance is active at a time on the primary server. Any outage due to SS or other important services may cause a failover to the standby node where HA-OSCAR deals with cluster availability and fault issues at the master node level. This is one of the instances where RASS principles are realized. The whole objective of this endeavor is to implement RASS principles in a downtime sensitive cluster environment. Even though the attacker/attack succeeds in bringing essential services down, the serviceability layer provided by HA-OSCAR quickly detects which services are down, and subsequently an action is initiated. Similarly, there are monitoring services for each the SMs residing on the compute nodes. Although at present there is no failover support at the compute node level, efforts have been directed towards incorporating recovery mechanisms in the compute nodes similar to the head node failover technique. We assume that a native cluster scheduler handles job migration. In section 4, we have outlined how DSI support various levels of security policies that guard against a possible back door or intrusion. This way DSI handles the security part. HA-OSCAR would

8 Box Leangsuksun, Anand Tikotekar, Stephan L. Scott, Makan Pourzandi, and Ibrahim Haddad

then take care of the cluster management issues of job migration, load balancing once the node is localized – after the compute node failover is supported. We have identified this as a possible future work. In addition, we envision that HA-OSCAR Adaptive Self-Healing (ASH) and security monitoring will be extended to support different levels of security and recovery escalation policy. We plan to further this investigation in a near term future work.

6. Integration Challenges

During our proof-of-concept implementation, there were several challenges. Since DSI 0.3 supports Linux kernel 2.4.17 and GCC 2.9x. It requires the LSM patch for kernel 2.4.17. Therefore, it imposed a requirement to recompile the kernel with the LSM option enabled.

Another challenge was to obtain the right combination of distribution and compiler. Red Hat 8.0 and 9.0 support GCC 3.x. However, DSI 0.3 supports GCC 2.96 (Red hat 7.x). Since HA-OSCAR requires OSCAR 3.0, our approach was to compile DSI individually and hand patch it to the experiment.

Currently, the work is in progress to port DSI to the latest stable kernel. Moreover, we will remove any dependency on GCC. Other integration work in progress includes finding ways to make HA-OSCAR compliant with the current OSCAR 4.0 release and future releases.

7. Experimental Results & Benchmarking

We conducted experiments using OSCAR 3.0, HA-OSCAR 1.0, and DSI 0.3. The experimental cluster consisted of two dual Intel Xeon server head nodes running at 2.4 GHz, each with 1GB RAM, 40GB HD with at least 2GB of free disk space and two network interface cards. There were 2 client nodes based on dual Intel Xeon servers with 512MB RAM and a 40GB hard drive. Each client node was equipped with at least one network interface card. We have evaluated the performance impact and the results are presented in Table 1.

Table 1. : Comparison of performances between a LSM patched kernel with and without DSM. Time units are microseconds. They represent the average of ten measurements.

<i>Test Type</i>	<i>Without DSM</i>	<i>With DSM</i>	<i>% Overhead</i>
Stat	1.92	1.94	1.0 %
Open/Close	2.68	2.68	0%
Exec	322.56	328.33	1.78 %
Sh proc	2140.75	2150	0.43 %

We performed additional testing on Pentium III 650 MHz Dell laptops with 256 MB RAM. All numbers are in microseconds.

The results suggest that there is minimum performance impact due to overhead while maintaining cluster reliability, availability, security and serviceability at the same level as original HA-OSCAR and DSI capacity.

We have also recorded a 3-5 seconds for a failover time based on the 2 second polling interval, and measured HA-OSCAR resource usages. We conducted 8 sets of HA-OSCAR resource usage experiments based on different polling intervals. We observed an average 0.9% CPU usage required by HA-OSCAR monitoring daemons. TCPTrace was used to measure HA-OSCAR network usage from each experiment. Fig. 5 shows network traffic load from each of the 30 minutes tests by various HA-OSCAR polling interval sizes.

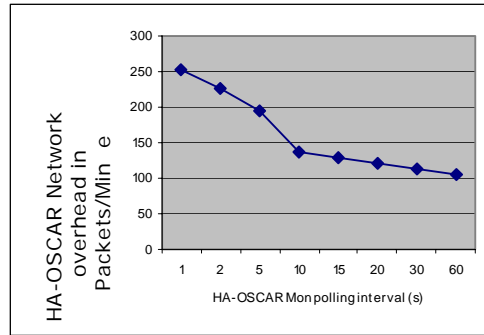


Fig. 5: Overhead in package according to polling interval

We also found that the impact of packet size was minimal. The details of HA-OSCAR 1.0 beta software release can be found at [10].

Table 2. : Performance analysis with IP packet modification.

	<i>Linux 2.4.17</i>	<i>Linux 2.4.17 with DSM</i>	<i>% Over-head</i>
Fork	167	169	+1.20%
UDP Local Access (Send message)	16.388	19.7	+20%
UDP Remote Access (Loop back)	133.44	173.88	+30%

We conducted a separate set of experiments to measure network overheads due to the DSM component. The results in Table 2 describe performance impact with IP header modification. The results illustrated in Table 3 are for UDP remote access based on a Pentium II 300 MHz desktop with 128 MB RAM. The overhead becomes significant with IP header modification in an IP packet. The results shown in Table 2 and Table 3 are obtained using Lmbench 2.0 [19]. LMBench ensures for the variabil-

ity in the data and produces accurate results by automatically rectifying for various overheads such as loop and timing measurement overheads. Furthermore, the results above are an upper bound on the performance overhead since no single security operation has been optimized. The percentage overhead is expected to significantly decline once the optimizations measures are in place. The distributed security policy (DSP) in DSI contains rules that govern a process's access to a resource. We consider supporting the IP header modification option to be disabled in DSP according to the process type to maximize the performance. However, doing this, may institute some security consequences, which demand a critical evaluation. This situation offers a perfect ground for a tradeoff between the performance and security needed.

Table 3. : Performance analysis without IP packet modification.

	<i>Linux 2.4.17</i>	<i>Linux 2.4.17 with DSM</i>	<i>% Overhead</i>
UDP Local Access (Send message)	16.388	17.084	+4.2%
UDP Remote Access (Loop back)	133.44	140.64	+5.4%

The feasibility study and results demonstrate that there was no significant degradation in the performance with our RASS experimental framework. DSI components such as SS and SCC are HA-enhanced, and thus result in alleviating single point of failure. Although the impact of DSI and HA-OSCAR overhead can not be eradicated, it is observed to be minimal.

8. Future Work

We plan to investigate the security policy enhancement and further improve the DSP management and ease of expression issues including security provisioning. Similar to SELinux security policy, the DSP remains an active area of research. In DSI, although the policy is in XML, the maintenance of the policy is far from easy. The main challenge remains in finding the right granularity for security rules. Fine-grained security rules are powerful and desirable from a security point of view. However, challenges remain for implementing these rules in a real world system when one needs to write, provision and maintain a data base of thousands of rules. This is considered an unacceptable burden to many administrators. Moreover, the ease of expression remains an issue. A study has been completed to formulate methods that aim to reuse information already contained in package management systems (RPM for Linux) to generate part of the security policy. This investigation aims to alleviate the above problem with security policy maintenance. Finally, we see opportunities in DSP in the way of extending classes [22] that are already present (such as file system support). The DSI currently implements MAC architecture through the use of the DSM. It uses the concept of LSM. LSM defines certain security hooks which are leveraged by the

DSM to implement the functionality. The DSM has an internal representation of the DSP and uses hash values to efficiently store it. The DSP is consulted for the security decision within the DSM and the policy is subsequently enforced. The DSP as of now supports 6 object classes which are socket, network, DCI, process, transition and socket_init. The object classes group together all the objects of its kind. There is a need to support more object classes within the DSP and DSM. SELinux [26] [25] is a security enhanced Linux which is based on the Flask security architecture. SELinux can enforce an administratively-defined security policy over all processes and objects in the system, basing decisions on labels containing a wide array of security relevant information. SELinux works by defining a separate logic for Policy decision and Policy enforcement. The policy decision logic is contained in a component called security server, which may be customized according to needs. SELinux uses Type enforcement (TE) method which makes use of an access matrix to perform decisions. The DSM could be replaced by the equivalent use of the SELinux. SELinux supports wide variety of security policies. This is accomplished since it is based on Flask architecture where policy flexibility was a major goal achieved. SELinux can work with the DSP, on which to base its decision rather than the TE policy. The TE policy already specifies large number of object classes and some rules are already enacted to save the efforts from customizing it from the start. Although the policy decision logic needs to be shifted from the DSM to some other component, this change is perceived to be relatively easy. The downside of using SELinux could be the heavy implementation of the policy enforcement logic. Moreover, by using SELinux any control on the type of functionality provided such as using DSM is lost and unnecessary functions could take their place in the performance impact. The feasibility study of using SELinux is already underway. The warnings and alarms generated by the DSM in the cases of access violations are transmitted to the corresponding security managers and are logged for auditing purposes. The error/audit logs can be classified according to their severity by using Naïve Bayes Text classifier. This can help generate decisions faster using already classified logs. Moreover, based on the severity an effective and efficient response can be mustered. This is also applicable in the case of ASH where different situations call for different solutions. The current strategy of initiating a failover is not efficient or optimal. A failover can take place for a “not so severe” situation and thus cause unnecessary failover. A recovery policy can be based on the “classified error/system/audit logs”. A strategy can be derived which bases its decision upon the severity of the situation and thus does not initiate failover for every situation. The Naïve Bayes classification works on the Naïve Bayes assumption that the words appearing in the text are independent of the class label with which they are attached. This assumption, even if obviously not true, works well in real life situations. Naïve Bayes classifier uses prior probability of the words to calculate the posterior probability of a class. In addition, we plan to explore more application opportunities for our RASS framework. For instance, our early study demonstrated that DSI is an adequate solution to guard against buffer overflow attacks. We plan to carry out some performance testing and verify how systems running DSI can resist attacks such as denial of service attacks. Better intrusion detection and recovery as well as damage localization will also be studied.

9. Summary

While there are solutions attempting to solve the many security issues involved with cluster environments. These solutions are not tailored to meet the requirements of cluster wide distributed security needs. In addition, existing security solutions are not compliant with the RASS principles; on the contrary they rely more on each machine “individually repealing attacks”.

This paper has demonstrated a need for RASS based cluster security and how the absence of one parameter eventually renders the cluster system vulnerable to further damage. Our approach of integrating HA-OSCAR (RAS) and DSI (S) is an integral attempt to view the perspective of RASS and the cohesiveness of cluster security, which we believe leads to a layered technique and thus becomes highly survivable.

The preliminary results produced integration between HA-OSCAR and DSI with minimal challenges. Thus, having “RASS” in mission critical cluster settings by far outweighs the performance impact. However, they also showed need for further investigation to reduce communications overhead due to DSI, especially for very ultra-scale HPC clusters where communications are particularly important and the size is extremely large. Based on our initial experiences, we proposed a solution to enhance DSI with a selective mechanism for remote authentication based on process type to limit the overhead to some specific processes. We plan to investigate the solution further in future work.

10. References

- [1] A. Apvrille and M. Pourzandi “XML distributed security policy for clusters” *The Computers and Security Journals, Elsevier*.
- [2] A. Brown and D. Patterson, “Undo for Operators: Building an Undoable E-Mail Store.” In *Proceedings of the 2003 USENIX Annual Technical Conference*, San Antonio, TX, June 2003
- [3] Carrier grade security http://www.osdl.org/docs/cgl_requirements_definition_11.pdf
- [4] FailSafe: <http://oss.sgi.com/projects/failsafe>
- [5] A Fox and D. Patterson, Self-Repairing Computers, *Scientific American*, June 2003
- [6] Hacker Attack Prompts Academic Supercomputers to block Outside Access Temporarily. *The Chronicle of Higher Education*, April 15, 2004.
- [7] Hackers Breach Clusters of US Supercomputers. *Arizona Daily Star*, April 15, 2004.
- [8] Hackers Breach Research Systems, But Data Kept Safe. *ComputerWorld*, April 19, 2004.
- [9] HA at HP: <http://www.hp.com/go/ha>
- [10] HA-OSCAR: <http://xcr.cenit.latech.edu/ha-oscar>
- [11] S. Hiltunen, C. Ugarte, “Enhancing survivability of security services Using redundancy”, *The International Conference on Dependable Systems and Networks (DSN01)*, July 01-04, 2001, Sweden.
- [12] K. Hwang, M. Gangadharan, “Micro-Firewalls for Dynamic Network Security with Distributed Intrusion Detection”, *IEEE International Symposium on Network Computing and Applications (NCA'01)*, October 2001
- [13] Kimberlite: <http://oss.missioncriticallinux.com/projects/kimberlite>

- [14] C. Leangsuksun, et al, "Availability Prediction and Modeling of High Availability OSCAR Cluster", *IEEE International Conference on Cluster Computing*, December 1-4 2003, Hong Kong.
- [15] C. Leangsuksun, et al, "High-Availability and Performance Clusters: Staging strategy, self-healing, experiment and dependability analysis" 5th *LACSI Symposium*, Santa Fe, NM October 12-14, 2004
- [16] M. Lee, E. Jung Kim, C. Won Lee, "A Source Identification Scheme against DDoS Attacks in Cluster Interconnects", *2004 International Conference on Parallel Processing Workshops (ICPPW'04)*, August 2004
- [17] P. Loscocco, S. Smalley, "Integrating Flexible Support for Security Policies in the Linux Operating System", in the Proceedings of the FREENIX track of the 2001 USENIX Annual Technical Conference, 2001, <http://www.nsa.gov/selinux>
- [18] P. A. Loscocco, S. D. Smalley, P. A. Muckelbauer, R. C. Taylor, S. J. Turner, and J. F. Farrell. "The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments." In *Proceedings of the 21st National Information Systems Security Conference*, pages 303-314, Oct. 1998.
- [19] L. McVoy, C. Staelin, "Imbench: Portable tools for performance analysis", *USENIX Technical Conference*, San Diego, CA. January 1996. pp. 279-284.
- [20] Mission Critical Linux: <http://missioncriticallinux.com>
- [21] OSCAR: <http://oscar.sourceforge.net>
- [22] M. Pourzandi, "A New Distributed Security Model for Linux Clusters", in the *Proceedings of the USENIX 2004 Annual Technical Conference*, Extreme Linux Special Interest Group, pp. 231-236, June 27-July 2, 2004.
- [23] M. Pourzandi, I. Haddad, C. Levert, M. Zakrezewski, M. Dagenais, "A Distributed Security Infrastructure for Carrier Class Linux Clusters", in the *Proceedings of Ottawa Linux Symposium 2002*.
- [24] ROCKS: <http://www.rocksclusters.org>
- [25] SE Linux: <http://www.nsa.gov/selinux/>
- [26] R. Spencer, S. Smalley, "The Flask Security Architecture: System Support for Diverse Security policies", in the proceedings of the 1999 USENIX Security Symposium
- [27] Steeleye: <http://www.steeleye.com>
- [28] C. Wright, C. Cowan, S. Smalley, "Linux Security Modules: General Security Support for the Linux Kernel", in the *Proceedings of the 2002 USENIX Security Symposium*.
- [29] W. Yurcik, X. Meng, G. Koenig, J. Greenesid "Cluster security as a unique problem with emergent properties", *The 5th LCI International Conference on Linux Clusters: The HPC Revolution 2004*, Austin, TX, May 18-20,