Instant Attack Stopper in InfiniBand Architecture

Manhee Lee* Eun Jung Kim* Ki Hwan Yum† Mazin Yousif§
*Department of Computer Science
Texas A&M University
College Station, TX-77840
manhee@cs.tamu.edu
ejkim@cs.tamu.edu

†Department of Computer Science
University of Texas at San Antonio
San Antonio, TX 78249
yum@cs.utsa.edu

§Advanced Component Division
Intel Corporation
Hillsboro, OR 97124
mazin.s.yousif@intel.com

Abstract

With the growing popularity of cluster architectures in datacenters and the sophistication of computer attacks, the design of highly secure clusters has recently emerged as a critical design issue. However, the majority of cluster security research has focused on how to detect and prevent attacks rather than on how to minimize the effect of attacks once detected. The action against detected attacks in the cluster is as important as the actual detection process since no detection mechanism is full-proof in its ability to protect cluster systems without the effective cluster-wide reaction.

In this paper, we propose a scheme, referred to as the Instant Attack Stopper (IAS) that can instantly confront security attacks in a cluster. Specifically we provide detailed implementation methods of IAS in InfiniBand Architecture (IBA) - a new promising communication standard for future System Area Networks (SANs) and clusters. IAS focuses on removing malicious communication on the IBA fabric among processes involved in an attack, which is accomplished through the proposed Security Management Agent (SeMA). We will show IAS deployment in different security levels to meet various security requirements.

1. Introduction

To achieve high performance and high availability in computing, computer clustering is a popular trend observed in industry as well as in academia. This has also spurred a great deal of research to further resolve issues related to performance, scalability, and quality-of-service (QoS). In addition to these design objectives, the critical nature of many Internet-based services mandates that these systems should be robust to attacks from the Internet. However, researchers have paid more attention to performance and cost efficiency than to security. As a result, numerous security loopholes of cluster servers come to the forefront and subsequently the design of a secure cluster has recently become a critical issue.

To defend against attacks exploiting those vulnerabilities, cluster systems usually depend on *firewalls* or the Intrusion Detection Systems (IDSs). Even though they can defend against a lot of attacks, they cannot prevent all the possible threats for many reasons. The following two reasons are reported recently. First, a firewall or the IDS can be vulnerable themselves. Geer reported that several products of famous security companies such as Check Point Software Technologies, Symantec, and Zone Labs have "potentially dangerous flaws that could let hackers gain control of systems, disable computers, or cause other problems" [6]. Wool quantified configuration errors of Check Point FireWall-1 in several sites [11]. Surprisingly, most sites have lots of errors. Almost 80 percent of the firewalls allow both "Any" service and insecure accesses to the firewalls so that they could have been broken into easily. The second reason is that if a hacker illegally gets a legitimate user password, he can enter cluster systems without any restriction. For example, a hacker infiltrated and compromised lots of computers in supercomputer networks at Stanford University after getting a legitimate user password recently [13].

Once a hacker breaks into a cluster, the impact of the attack within the cluster would be severe. That is because one infected system, which is believed to be trustworthy, may instantly paralyze the whole cluster through the high speed network. To prevent these attacks, system-alone security tools are currently being adopted due to the lack of cluster-specific security tools. However, as Yurcik *et al*. insist, cluster security is different from security of multiple systems [8, 12].

Usually the studies on cluster security have focused on how to prevent and detect attacks rather than on how to confront the detected attacks. The action against a detected attack in the cluster is as important as the detection because the cluster-wide detection may be useless without an effective cluster-wide countermeasure. Despite the importance of the action against the detected attacks, to our knowledge, little research has been done before. In this paper, we propose the Instant Attack Stopper (IAS) scheme to respond security attacks instantly in the cluster. We define *conspirators* as a group of processes residing in one or more nodes and actively taking part in currently occurring attack. We focus on

how to search for other conspirators after finding a conspirator. Based on the observation that conspirators need to communicate with each other through a cluster interconnect before an attack, we speculate that a process is also a conspirator if it has communicated with other conspirators. IAS blocks its traffic temporarily and issues alert to each node's attack analyzer to make it investigate this case. IAS can be implemented in many different ways, but in this paper we provide detailed implementation methods in InfiniBand Architecture (IBA), a new promising communication standard for future system area networks (SANs) [16].

As more and more clusters are using IBA for inter-node communication, the security enhancement in IBA will become critical. That is the reason for our focus on IBA. More specifically, there are many advantages in implementing IAS in IBA. First, IBA already has a well-defined management architecture where IAS can be easily integrated. Second, the response time of IAS in IBA will be much faster than IAS in the OS. Third, IAS in IBA matches the recent research trend to make IBA more secure. By enforcing the security of IBA, a cluster based on IBA will have a certain level of security without depending on OS vendors' implementation. In addition, since IAS can be well integrated into upcoming security infrastructures such as Security Enhanced Linux (SELinux) and the Distributed Security Infrastructure (DSI), IAS will contribute to enhancing cluster security [8, 17].

The remainder of this paper is organized as follows. Section 2 reviews some related work and Section 3 briefly explains the user-level communication and IBA. With this background, in Section 4, we propose the IAS in IBA and state its advantages over the OS-level IAS. The detailed implementation methods are also discussed. Section 5 describes how IAS can be applied to meet various security requirements, including methods to integrate IAS with SELinux and DSI, followed by conclusions in Section 6.

2. Related Work

Foster, *et al*. [5] proposed a communication library allowing programmers to communicate securely in the geographically distributed computing environment. It provides well-organized security in the Grid, but it cannot be directly applied to cluster security because Grid security is a kind of the distributed security on the Grid middleware. Connelly and Chien have done a forefront research directly related to the cluster security in [3]. They focused on providing remote procedure calls (RPCs) with confidentiality in tightly coupled components applications. They applied traditional security functions such as transposition, substitution, and data padding on the marshalling layer. Their performance analysis showed that encryption can operate in clusters. Dimitrov and Gleeson [4] presented security enhancement methods in three levels: network host interfaces, SANs, and protocols for interconnecting many SANs. Their approach can be a good systematic guideline for enhancing the security of cluster systems based on the Myrinet or Virtual Interface Architecture (VIA). As a specific topic in cluster security, Lee *et al*. [7] proposed Deterministic Distance Packet Marking scheme to identify a Denial of Service (DoS) attacker in cluster interconnects.

More recently, there are two noticeable ongoing studies on cluster security. Pourzandi, *et al*. proposed a new security model called Distributed Security Infrastructure (DSI) [8, 9]. DSI provides a cluster-wide security space allowing the fine-grained security enforcement on distributed applications. DSI is based on providing a process-level resource and access control and its implementation is available on [14]. Yurcik, *et al.* are doing another promising research [12]. They claimed that high bandwidth connections, extensive computational power, and massive storage capacity of a cluster can be used for another attack and as a repository for illegal contents. They also introduced a new concept called "emergent security properties," to identify security characteristics unique to a cluster, which can be used to develop a unified monitoring tool for a cluster.

The last research we should not miss is Security Enhanced Linux (SELinux) [17]. Like DSI, SElinux is providing higher security availability by enforcing mandatory access control over all subjects and objects in the system but not in the cluster. Red Hat Inc. recently announced that Fedora 3 core would have SELinux as a default option [15]. Even though it is system-level security, we think SELinux will give significant changes in cluster security because many clusters are Linux-based cluster systems.

3. InfiniBand Architecture

The current low cost clusters and early clusters are using traditional communication protocols such as TCP/IP where the operating system (OS) processes all works for communication from packetizing data to checking errors. This limits the cluster performance significantly because the OS has to spend substantial amount of CPU time as a cluster needs to communicate at high speed [2]. To remove this overhead, several user-level communication protocols were proposed [1]. Among them, InfiniBand Architecture (IBA) is a new communication standard to design SANs for scalable and high performance clusters. In this section, we describe IBA briefly.

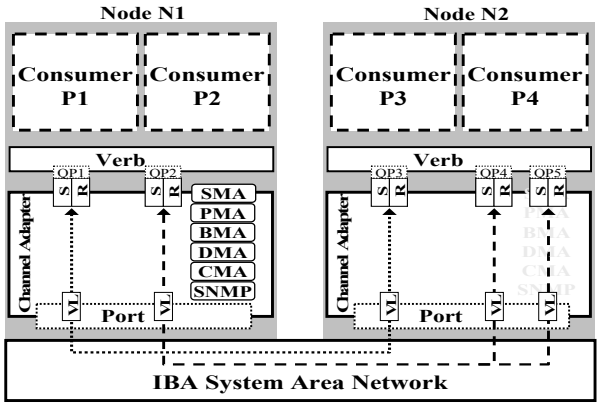


Figure 1. InfiniBand Architecture

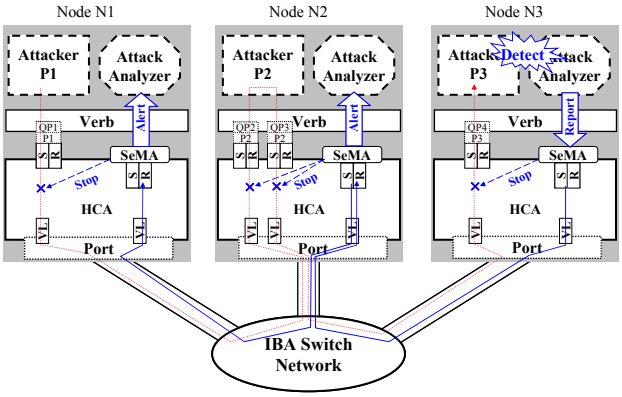


Figure 2. Instant Attack Stopper in IBA

Channel Adapter (CA). Physically a CA is similar to Network Interface Cards (NICs) in terms of terminating a link and supporting hardware-level signaling. Unlike NICs, CA handles all transport-level functions offloaded from the OS. Processing nodes are attached to an IBA network through Host Channel Adapters (HCAs) and I/O nodes can be attached to a network through Target Channel Adapters (TCAs). In this paper, we use CA representing both of them.

Queue Pair (QP). A QP is a transport-level entity implemented in a CA. One Send and one Receive queues are created together and identified by Queue Pair Number (QPN). Information about a QP is stored in a QP context. There are five types of QPs: Reliable Connected (RC) QP, Unreliable Connected (UC) QP, Reliable Datagram (RD) QP, Unreliable Datagram (UD) QP, and Raw QP. Connected QPs (RC and UC), can communicate with only one QP designated at the creation time, while Datagram QPs (UC and UD) can communicate with any UC and UD QPs, respectively. For example, in Figure 1, the connected QPs, QP1 and QP3, can transfer packets between them. In contrast, a datagram QP, QP2, is communicating with QP4 and QP5. Reliable QPs (RC and RD) provide error free, in-order communication, while Unreliable QPs (UC and UD) do not. A Raw QP is providing a way to communicate with non-IBA destinations.

Verbs. A verb is a loosely defined API of functions supported by a CA. Consumers (or applications), the OS, and the drivers communicate with a CA using verb calls. The IBA specification describes their input and output parameters with execution effects.

Management Architecture. There are many management components in IBA. The most important one is the subnet management performed by a Subnet Manager (SM) and a Subnet Management Agent (SMA). An SM's roles include discovering a network topology, assigning an address to each port, and establishing possi-ble paths between nodes. To manage a subnet, an SMA in each CA is communicating with an SM by sending and receiving Subnet Management Packets (SMPs), which are a type of Management Datagram Packets (MADs). Also, there are a group of managers collectively referred to as the General Services Managers (GSMs) that include Performance Manager (PM), Baseboard Manager (BM), Device Manager (DM), Communication Manager (CM), and SNMP tunneling Manager. Figure 1 shows their corresponding agents (PMA, BMA, DMA, CMA, and SNMP) in a CA.

4. Instant Attack Stopper (IAS) in IBA

4.1. Basic Design

IAS (Instant Attack Stopper) is a scheme to stop conspirators from sending attack traffic to a cluster. IAS is based on an observation that to make a conspiracy, conspirators need to communicate with each other before an attack. In other words, if they want to conspire an attack, they need to communicate through a cluster interconnect. From this observation, we can suspect that a process is a conspirator if the process is communicating with other conspirators. IAS blocks its traffic temporarily and issues alert to each node's attack analyzer to make it investigate this case. And then ISA traces back other processes with which this conspirator is communicating. This operation continues until there are no more conspirators.

Figure 2 explains how IAS works. A conspirator, P1, is steering P3 through P2 to attack N3. We assume the attack analyzer in N3 detects an attack from P3. Immediately, the analyzer reports an attack to the Security Management Agent (SeMA). SeMA blocks QP4's traffic while sending a MAD to the SeMA in N2. In the same way, it alerts to the analyzer and blocks communications of QP2 and QP3. At the same time, it sends another MAD to N1. All attack traffic is finally blocked when the SeMA of N1 blocks QP1 and alerts to its attack analyzer.

It appears to be too strict that a process is regarded as a conspirator only because it is communicating with another conspirator. However, its security will be fortified more because communicating with a conspirator might hamper a normal operation of the process and even change the process itself into a real conspirator by exploiting its vulnerabilities. In addition, after a while, if the process turns out to be good-mined, it can continue its original operation in the absence of conspirators because true conspirators and their QPs will be removed by attack analyzers.

4.2. Implementation Methods

IAS can be implemented in OS-level (or, application-level) and in IBA-level. When a process wants to use QPs, it should call a verb function, which can be monitored by the OS. The OS can get all information about QP communications such as QP numbers and remote CA addresses. Therefore, it can know all local QPs' destination QPs and their CAs' addresses. With this information, it can send alerting messages to other OSs to trace back conspirators. It has some advantages over IBA-level implementation because it does not require changes in the IBA specification and may provide better manageability than IBA.

Despite the difficulty to modify the IBA specification, we think IBA-level IAS has significant advantages over OS-level IAS. First, IBA already has a well-defined management architecture. Management components at each CA can be used to implement IBA-level IAS and furthermore IBA-wide managers can have extensive functionalities to enhance IBA-wide security. Second, the reaction time of IBA-level IAS will be much faster. During a trace-back, some OSs under attack may not react quickly. IBA-level IAS can process MADs with a higher priority even when its OS works improperly. Third, IBA-level IAS matches recent research to make IBA more secure. By enforcing the security of IBA itself, a cluster based on IBA will have a certain level of security not depending on OS vendors' implementation.

We propose two new management components in the GSM: Security Manager (SeM) and Security Management Agent (SeMA). An SeM will be in charge of enforcing security inside IBA such as changing security policies in the Security Management Agent (SeMA). An SeMA in each CA responds to a request from SeM and at the same time cooperates with other SeMAs and the OS to cope with security attacks[1]. An SeMP MAD is a packet format of GMP MAD, which can be implemented rather easily by defining a new Management class, SecMgt, an identifier telling which management agent should handle the MAD. SeMP MAD contains a QP number, which is considered to participate in an attack, as an additional data into existing MAD contents.

Upon receipt of the SeMP MAD, the SeMA should search for conspirators to which the QP belongs. We assume that if one QP among multiple QPs created by a process is alleged to join an attack, its owner process is regarded as a conspirator and the other QPs to be used for the attack. This assumption is based on the fact that a process is almost the smallest entity to perform one job. Therefore, IAS should trace back through the other QPs as well. For example, in Figure 2, QP2 is found through QP3 and QP1 also through QP2. To implement this, each QP context should contain its owner process ID, or PID. In addition, the PID should be included in input parameters of the QP verb call when the process creates a QP. An attacking process may deliberately use a wrong PID. The OS can prevent this simply by comparing the input PID and the actual PID of the calling process.

Then, the SeMA should send SeMA MADs to remote QPs. It is clear to which QP the SeMA should send a MAD if the source QP is an RC or an UC, since the destination QP is fixed. On the contrary, RD and UD QP can communicate with more than one QP. Therefore, if the SeMA wants to trace back all the QPs, the CA should provide history information, which can be implemented by an array of memory. However, with the limited memory in the CA, it seems impractical. Otherwise, at the very least, RD or UD QP context can store one destination QP number of the previous or the current communication, resulting in a trace-back through one destination QP. This might limit the security strength of IAS, but the security level can be increased by using different process information, which will be discussed in the next section.

After sending the SeMA MADs, SeMA makes an interrupt to alert OS to initiate an attack analyzer. For the attack analyzer to take care of the attack, it needs to get the PID and later removes the associated QPs or releases the blocks made by SeMA. To support these operations, several new verb calls are necessary as shown in Table 1. An additional verb for the attack analyzer to report an attack to the SeMA is necessary as well. This will be the first verb call by an attack analyzer to trigger a series of trace-backs. Table 1 summarizes all the expected modifications to IBA specification.

Table 1. InfiniBand Modification for IAS

Category & Type	Name & Description
MAD: Management Class	***SecMgt***: Security Manager. Identifying which management agents should handle this MAD. Stored at the second byte of MAD.
MAD:	***DestinationQP***: QP number, which is considered to

[1] The roles of SeM and SeMA implementing IAS in this paper are limited even though there is a large room to extend for enhancing the security of IBA. Note that only SeMA is used in IAS.

SeM Specific Field	join an attack on other nodes. SeMA traces back other conspirators which this QP's process is communicating with.
QP: Context	***PID***: Process ID of QP's owner application.
QP Verb: Create Input Parameter	***PID***: Process ID of QP's owner application. OS should check its validity
Verb: SeM Verb Calls	***ReportAttack***: Attack analyzer call to report an attack to SeMA in CA. PID should be in the input paratmeters.
	GetAttackPID: Attack analyzer call to get a PID from SeMA in CA. Return value is PID.
	ReleaseBlock: Attack analyzer call to release blocks from QPs associated with a PID. PID should be in the input parameters.
HCA: Interrupt	***AttackAlert***: SeMA makes an interrupt to initiate attack analyzer to investigate an application associated with a PID.

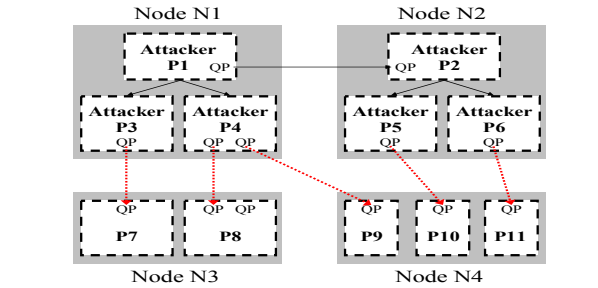
5. Multi-level Security in IAS

In cases multiple processes are related with an attack, IAS may not stop all attack traffic, since ISA blocks only QPs associated with one conspirator in each node. Due to the limited information in a QP context, it is not trivial to find out all processes related to the attack. In this section, we will describe two levels of IAS enhancement to search the processes and the integration of IAS with two new promising system structures: Secure-Enhanced Linux and Distributed Security Infrastructure [8, 17].

5.1. Sibling Process Level

Sibling processes, created by the same parent process, often do the same work especially when they execute the same program. If one sibling is declared as a conspirator, it is highly probable that the others are also conspirators. However, the basic design of IAS cannot block those siblings. For example, if the attack analyzer detects an attack in P9, the basic IAS can find only two more processes, P4 and P8 as shown in Figure 3 (b).

The attack analyzer can speculate other conspirators more aggressively by looking into its sibling processes. Upon receipt of an interrupt from a CA, the attack analyzer gets PID from the CA and finds siblings executing the same program. If any, the attack analyzer makes additional *ReportAttack* verb calls with the newly found PID. With this application, more conspirators can be found as shown in Figure 3 (c).



(a) Attack example

P9->P4->P8

(b) Basic search

P9->P4->P8
|->P3->P7

(c) Sibling process level

P9->P4->P8
|->P3->P7
|->P1->P2->P5->P10
|->P6->P11

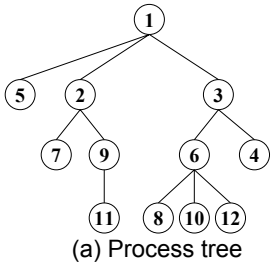
(d) LCA level

Figure 3. Multi-level Security in IAS

5.2. LCA (Lowest Common Ancestor) Level

For stronger security, we can search upward for parent processes of conspirators. For example, in Figure 3 (a), P1 masterminding the current attack can be blocked because P1 is the shared parent of two conspirators, P3 and P4. Finding shared parents appears to be simple but actually not because P1 is also a child process of another parent process and it is difficult to decide whether P1's parent is a conspirator. To decrease the possibility of blocking innocent processes, we search for parent conspirators as conservatively as possible. Therefore, we propose to consider a process as a conspirator when at least two conspirators share the process as the lowest common ancestor (LCA). The LCA problem is a well-known classic problem finding the deepest node in a tree that is a parent of two given nodes [10]. In addition, all the children processes of the conspirator are also considered as conspirators. For example, in Figure 4 (a), the LCA of node 8 and 12 is node 6 and that of node 4 and 8 is node 3. In our scheme, if node 4, 8 and 12 are involved an attack, their LCA and its children processes, node 3, 6 and 12, are considered as conspirators. Within this level, all conspirators in Figure 3 can be found as shown in Figure 3 (d).

It is, however, a significant overhead to maintain and update a process tree all the time because security attacks are sparse while the creation and death of processes are dynamic. Therefore, we propose to make a process tree and find a LCA at every attack. Figure 4 (b) describes the LCA pseudo code and an example. After sorting all the processes with PID, it increases the counters of parents of all conspirators. If P4, P8, and P12 are conspirators in Figure 4 (c), the counters of the LCA, P3, will have the same with the number of conspirators. *findlowest()* finds



(a) Process tree

```
sort(p-list, PID);
While (conspirators ≠ ∅) {
cons =select(conspirators);
ptr = find(p-list, cons);
while (ptr.PID != 1) {
      ptr.count++;
      ptr = find(p-list, ptr.PPID);
 }}
findlowest(p-list);
```

(b) LCA pseudo code

Input: p-list & P4, P8, P12

PID	1	5	2	3	7	9	11	6	4	8	10	12
PPID	0	1	1	1	2	2	9	3	3	6	6	6
Cnt												

Output: sorted p-list & LCA P3

PID	1	2	3	4	5	6	7	8	9	10	11	12
PPID	0	1	1	3	1	3	2	6	2	6	9	6
Cnt			**3**	1		2		1				1

(c) Input & output of LCA pseudo code

Figure 4. Process Tree & Pseudo Code

the deepest child among shared parents and PPID is the parent PID. This code is bounded to the sorting algorithm of which lower bound is O(*nlogn*) with *n* processes. Consequently with additional computing costs the security can be fortified substantially.

5.3. IAS Implementation in SELinux and DSI

SELinux is providing higher security availability by enforcing mandatory access control over all subjects and objects in the system [17]. Recently, SELinux has been adopted as a default option to Fedora Core 3 [15]. SELinux defined two policy-independent labels: security context and the security identifier (SID). The SID is an integer mapped to a security context.

Most processes forked by one user will have the same SID with their parent processes'. If IAS blocks all processes with the same SID, it will be too harsh because they can execute totally different programs. Therefore, we propose to use SID along with techniques explained in the previous two subsections. If the search space for a LCA is limited to processes sharing the same SID, we can save some computing cost. At the same time, it can make a fine-grained blocking scheme possible.

The other promising security architecture is the distributed security infrastructure (DSI) [8]. Pourzandi proposed a security model in the cluster to provide the cluster-wide process-level security by assigning the same security context to a group of processes acting as one application. Every process and resource has *security context identifier* (ScID) and any access to the resource is determined by comparing ScIDs of the requesting process and the resource. A conceptual zone with the same ScID is referred to as a *security zone*, which is different from SELinux wherein a security context is effective only in one system.

Since all objects in one security zone have the same security characteristics, if any of them are defined as conspirators, all objects in the security zone should be considered as conspirators as a whole. To do this, IAS can use the ScID instead of the PID. The ScID will simplify the searching operation of IAS while enhancing the security level significantly.

6. Conclusions

In this paper, we proposed the Instant Attack Stopper (IAS) scheme to react to security attacks instantly in the cluster and specifically we provided detailed implementation methods in IBA. The Security Management Agent (SeMA) in each CA is a key component in IAS. It receives and sends security management MADs and processes required operations to QPs and the OS. Since General Service Management (GSM) is well defined in IBA, our proposal is expected to be implemented rather easily. In addition, to enhance security level more, we proposed several ways to apply IAS: sibling process level, lowest common ancestor level, SELinux, and Distributed Security Infrastructure. These options make IAS more applicable to real systems and more helpful to enhance the cluster security.

7. References

[1] R. A. Bhoedjang, T. Rühl, and H. E. Bal, "User-Level Network Interface Protocols," in *IEEE Computer*, 31(11):53-60, November 1998.

[2] J. Chase, A. Gallatin, and K. Yocum, "End-System Optimizations for High-Speed TCP," in *IEEE Communications Magazine,* Vol. 39, no. 4, pp. 68-74, April 2001.

[3] K. Connelly and A. Chien, "Breaking the Barriers: High Performance Security for HighPerformance Computing," in *New Security Paradigms Workshop '02*, 2002.

[4] R. Dimitrov and M. Gleeson, "Challenges and New Technologies for Addressing Security in High Performance Distributed Environments," in *Proceedings of the 21st National Information Systems Security Conference* 457-468, 1998.

[5] I. Foster, N. Karonis, C. Kesselman and S. Tuecke, "Managing Security in High- Performance Distributed Computations," *Cluster Computing*, Vol 1, issue 1, pp. 95-107, 1998.

[6] D. Geer, "Just How secure Are security Products?," *Computer*, Volume 37, Issue 6, Jun. 2004.

[7] M. Lee, E.J. Kim, and C.W. Lee, "A Source Identification Scheme against DDoS Attacks in Cluster Interconnects," in *Proceedings of International Workshop on Network Design and Architecture (IWNDA) 2004*, 2004.

[8] 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.

[9] M. Pourzandi, I. Haddad, C. Levert, M Zakrewski, and M. Dagenais, "A Distributed Security Infrastructure for Carrier Class Linux Clusters," *Ottawa Linux Symposium, 2002.*

[10] B. Schieber and U. Vishkin, "On Finding lowest common ancestors:simplifications and parallelization," *SIAM Journal on Computing*, 17:1253-62, 1988.

[11] A. Wool, "A Quantitative Study of Firewall Configuration Errors," *Computer*, Volume 37, Issue 6, Jun. 2004.

[12] W. Yurcik, G.A. Koenig, X. Meng, and J. Greenseid, "Cluster Security as a Unique Problem with Emergent Properties: Issues and Techniques," In *Proceedings of Linux Revolution 2004*, 2004.

[13] "Attackers infiltrating supercomputer networks," http://news.com.com/2100-7349-191024.html

[14] Distributed Security Infrastructure, http://disec.sourceforge.net/

[15] Fedora, http://fedora.redhat.com/

[16] InfiniBand Trade Association, "InfiniBand Architecture Specification, Volume 1, Release 1.1," November 2002. Available from http://www.infinibandta.org.

[17] SELinux, http://www.nsa.gov/selinux/