NVisionCC: A Visualization Framework for High Performance Cluster Security

William Yurcik
NCSA
University of Illinois-UC
Champaign, IL 61820 USA
byurcik@ncsa.uiuc.edu

Xin Meng
NCSA
University of Illinois-UC
Champaign, IL 61820 USA
xinmeng@ncsa.uiuc.edu

Nadir Kiyanclar
NCSA
University of Illinois-UC
Champaign, IL 61820 USA
nadir@ncsa.uiuc.edu

ABSTRACT
Large high performance clusters are gaining popularity as a means of harnessing vast computing resources at low cost using commodity components. Cluster system administrators face difficulty from two related problems. First, while several cluster monitoring solutions collect data on the node state and overall performance of a cluster, few monitors place emphasis on the meaningful visual presentation of this information which is increasingly important as cluster size in nodes grows beyond the human capability to manage using command line tools. Second, while cluster state and performance data have been visually monitored in several systems, there are currently no cluster monitors that visualize cluster security events. We have developed a framework for effective visualization of a high performance cluster security which we describe in this paper. We present GUI screenshots from a security visualization tool based on this framework and discuss our experience using this tool on high performance clusters at NCSA.

Categories and Subject Descriptors
C.2.0 [**Computer-Communication Networks**]: General – *security;* H.5.2 [**Information Interfaces and Presentation**]: User Interfaces*;* K.6.5 [**Management of Computing and Information Systems**]: security and protection – *invasive software*

General Terms
Security, Management, Human Factors

Keywords
NVisionCC, cluster security, high performance security, security visualization, security situational awareness

VizSEC/DMSEC'04, October 29, 2004, Washington, DC, USA.

1. INTRODUCTION
Clustering commodity computer components to provide the supercomputing is the current direction of many high performance computing centers. It is both cost efficient (cycles per dollar) and incrementally extensible. However, the level of management coordination required to provide production service using clusters is significant – a significant amount of energy is devoted to simply making clusters work.

Using command line tools to find cluster state information is reasonable when clusters are within a human-comprehensible range.[1] For instance, while a 12-node cluster may be managed using command line tools, the same command line tools do not scale to clusters with over a thousand or even a hundred nodes – while the exact threshold of human comprehension is debatable the extremes exhibit the point. For this reason visual cluster monitors have begun to appear, the most popular of which we refer to later in this paper. Visual monitoring leverages the inherent processing capabilities of humans estimated to be 150 Mb/screen[2] to allow administrators to manage clusters for performance.

Thus as clusters become larger, they are harder to control and thus require visualization techniques if a human is to remain in-the-loop. As a specific example, at NCSA our clusters range from:

- Titan (160 Nodes)
- Mercury (256 Nodes)
- Platinum (512 Nodes)
- Tungsten (1450 Nodes)

Monitoring for cluster performance is the current state-of-the-art, after all what else would need to be monitored? It turns out that while monitoring clusters for security was proposed by the lead author of this paper as a significant threat in early 2003, there was not a consensus that this was a real problem until the extensive cluster security intrusions of Spring 2004 demonstrated the dangerous state of cluster security was not just a theoretical risk [2,3,4].

[1] For example, KCAP software is Web-based and free, bWatch is based on Tcl/Tk and is free, XPVM can be attached to PVM and is free.

[2] Edward Tufte uses this 150 Mb/screen metric for human visual processing in his popular "Presenting Data and Information" Workshops (Louisville 8/2004).

While in-depth security aspects of clusters are beyond the scope of this paper (we are focusing on visualization), we summarize by stating the following observations:

- In terms of compute cycles, network bandwidth, and available storage, a cluster is an ideal hacker platform for processing malicious jobs (password cracking), launching malicious attacks against Internet sites (denial-of-service attacks) or for warehousing unauthorized files (copyrighted multimedia)
- Many cluster have all their interior nodes Internet-accessible with minimal protection, especially clusters participating in "Grid computing"
- With desire for high performance, cluster traffic filtering is often performed at border routers which provide minimal security protection
- Intrusion detection systems do not distinguish or prioritize cluster nodes from systems on enterprise networks
- A cluster intrusion is often a "class break" , a single compromised cluster node represents a dramatically-increased risk to the rest of the cluster nodes due to the fact that many nodes share identical configurations but also the fact that cluster users tend to coordinate access across security domains so when one security domain is compromised there is often a cascading series of intrusions.

The reaction of security engineers to the cluster intrusions of Spring 2004 has been to apply established "best practices" for enterprise security to clusters with renewed vigor (patching, authentication, configuration control, etc. "on steroids"). We posit that applying enterprise security best practices to clusters while beneficial in the short-term is actually the wrong long-term approach because a cluster is fundamentally different. Although the behavior of individual nodes within a cluster may be simple and approached with a growing set of established enterprise security techniques, cluster security is essentially a different problem [13,5]. Effective security monitoring in the context of cluster systems requires tools that evaluate the unique state of the cluster as a whole.

Consider the limitations of current enterprise security tools that evaluate data considered independently of any cluster-specific context while a cluster-aware security tool may be designed to evaluate data within a cluster context – for instance whether a given node is behaving abnormally based on information from sources such as the cluster job scheduler.

The novel contribution of this work is that, to our knowledge and at this point in time (September 2004), **no security tools exist tailored specifically for the unique cluster environment. We present a working cluster security tool based on an extensible visualization framework that fills this gap**. The remainder of this paper is organized as follows: Section 2 identifies and discusses existing cluster monitors that use visualization. Section 3 focuses on the specific question of monitoring for cluster security. Section 4 introduces a novel visualization framework for visualizing cluster security for which we have built a tool currently being deployed on clusters at NCSA. We close with conclusions based on our initial experiences and future plans in Section 5.

2. RELATED WORK

Several tools exist to facilitate the monitoring of computing clusters. One of the earliest of theses was bWatch, a relatively simple monitoring tool written in Tcl/tk, and using rsh as the de-facto communication protocol. Since then, a number of systems have been deployed which offer improved robustness, low overhead, and scalability, in varying degrees. The design of these tools was influenced by the need for effective, lightweight performance monitoring, however, and none of them explicitly support security monitoring and visualization.

Clumon [1,9] is a cluster monitor designed at NCSA. Clumon has the most "traditional" structure of all the tools we describe due to the fact that there is a single logical master node[3] that monitors all compute nodes. One host runs a daemon process, *clumond*, which is responsible for collecting performance metrics from each compute node and storing this data in a central MySQL database. Since all data is written to a central location, multiple programs may easily extract and display the collected information in any number of ways. To this date, the primary monitoring interface used in Clumon is web-based. To collect data from individual compute nodes, Clumon leverages Performance Co-Pilot (PCP) [6], an open source tool for performance tracking developed by SGI. PCP is implemented as a per-node daemon that gathers various performance metrics about a machine and that returns them in response to network queries. PCP runs on all cluster nodes are to be monitored and provides the mechanism by which Clumon is able to collect data.

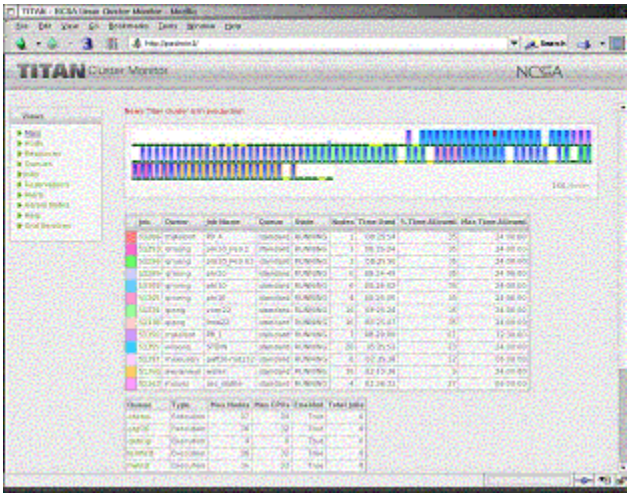

Figure 1. Clumon Main Window

Ganglia [7] is a cluster monitoring tool developed at UC Berkeley in conjunction with the Millennium project [11]. In this system, as in Clumon, cluster nodes run a daemon process, *gmon*, which is responsible for gathering and serving performance and load information to a monitoring host that runs the *gmetad* daemon. Also as in Clumon, Ganglia has a web-based interface which can display performance information for the cluster as a whole, as well as individual hosts. Ganglia

[3] There can be more than one node involved in monitoring. *Clumond*, the log database, and the http server can all run on separate computers. Logically speaking, however, monitoring is done from a single point.

differs from Clumon significantly in implementation as well as in design, however. Communication between nodes is done over TCP connections with an XML-based protocol. One interesting implication of the use of a hierarchical XML-based language is that the monitoring host can aggregate and serve information in the same format as individual hosts, allowing for a hierarchical monitoring structure. In this way, several physical clusters can make up one logical "meta-cluster." Storage is managed by periodically writing data to a fixed-sized Round Robin Database (RRDTool) in which samples are periodically reduced in granularity as they age [8]. While this helps contain the space explosion that can result with large volume log files, the trade-off is that these databases become less suitable for use in security-related monitoring tasks.

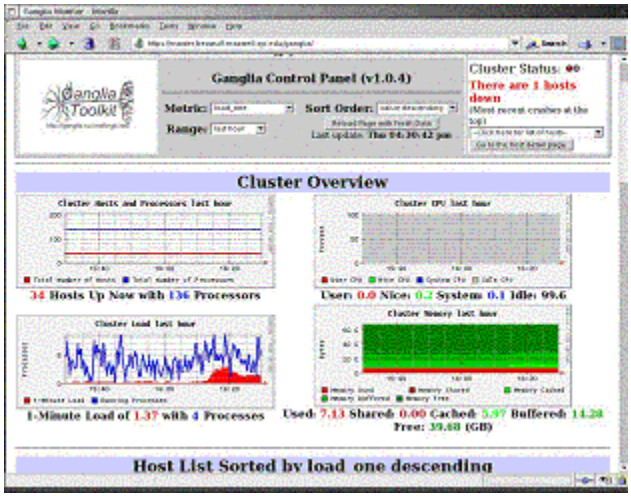

Figure 2. Ganglia Main Window

Supermon [10] is a cluster monitoring system designed to be fast and lightweight. The authors of Supermon claim that it is possible to sample the state of a given machine at a rate of up to 6600 samples per second. A per-host daemon (*mon*) serves information to a monitoring host. Similarly to Ganglia, the *mon* cluster monitoring daemon can itself serve data to another monitoring host, allowing for a hierarchical monitoring structure. Although Supermon can allow for very fine-grained monitoring of a cluster, the project has a narrower scope than either Clumon or Ganglia since it is focused on efficient data collection, not storage or presentation. No integrated web interface exists for Supermon.

3. CLUSTER SECURITY MONITORING

Security "Best Practices" from enterprise security that have been forcibly applied to the cluster environment include automated software configuration and patching, private network designs, network and host intrusion detection systems, firewalls, and authentication schemes. This has resulted in increased noise to the security analyst who cannot discern alerts from clusters from other network system alerts thus defeating the guidance that a security monitor must do no harm – it must not increase the possibility of compromise. Instead any security information gathered from nodes must be verified in the context of the whole cluster.

Many of the traits of cluster performance monitoring discussed in Section 2 would superficially appear to also apply equally well to cluster security monitoring. It must first be decided what data reliably portrays the state of security at both the cluster and node levels. When monitoring a cluster, it may be desirable to send alerts triggered by some event. An example of an event may be the violation of some type of policy. In the area of performance, policy is comparatively easy to specify. Optimal performance can be specified as a set of bounds within which certain observed metrics must remain. Alerts can be fired, if necessary, when these bounds are violated. It is still possible to have each host observe its state and alert a monitoring daemon if necessary but two difficulties arise. First, determining whether a host has been compromised is much harder than insuring performance bounds are respected. While precautions can be taken not to expose a host more than is necessary, a single scalar value that represents the security posture of a cluster does not exist. It is similarly impossible to automatically detect all possible vulnerabilities on a host and repair them. By restricting logins to the head nodes of a cluster, and by constraining users in the type of jobs they can execute, reliable detection of malicious programs becomes easier. However, a cluster lockdown is not always desirable or even feasible.

While Clumon, Ganglia, and Supermon cluster monitors have GUIs that provide an overall view of cluster performance with the ability to 'drill down' to examine raw data of a particular host, there is little security information that can be gleaned from these visualizations. Cluster performance is a relatively simple thing to visually represent with statistics such as cpu and memory usage comparing the actual value measured versus an operating range. In addition, it is easy to aggregate these statistics. For example, to view overall cluster utilization, averaging cpu utilization values over the all nodes in the cluster may be sufficient. Since many performance numbers are similarly one-dimensional, more detailed performance displays are also easy to interpret.

4. THE "NVisionCC" VISUALIZATION FRAMEWORK

We believe that process monitoring is perhaps the most far-reaching source of information about cluster security because an attacker cannot gain access to cluster nodes without running at least some processes. The appearance of unexpected processes on a cluster node is a strong indicator that the security of the node has been compromised. Thus our goal is to enable a human to monitor the many processes on a large cluster.

This goal is made feasible since many clusters consist of several distinct types of nodes with a large degree of homogeneity within node types as depicted in Figure 3. For example, the nodes that make up a cluster typically can be grouped into categories such as head nodes (used by end-users to access the cluster, compile software, and submit and monitor jobs), compute nodes (allocated to users to run serial or parallel jobs), storage nodes (containing the disk space used to hold datasets for cluster jobs), and management and monitoring nodes (typically only accessible to cluster administrators).

NVisionCC (where CC = Cluster Computing) is a cluster security monitoring tool we have developed based on the requirements we have described [12]. NVisionCC is implemented as an extension to the Clumon monitoring system currently in use at NCSA. Its design, therefore, is also similar

to that of Clumon. Our tool is a security analysis module that may be installed on a cluster monitoring host, and therefore able to communicate with *clumond*. The security module will than analyze the collected data, looking for security events. This program structure is advantageous for both reliability in monitoring and ease of implementation. By running from a central location, we retain the Clumon data collection and storage facilities but add the ability to verify the correctness of this data from a trusted location (the monitor host) on the network.

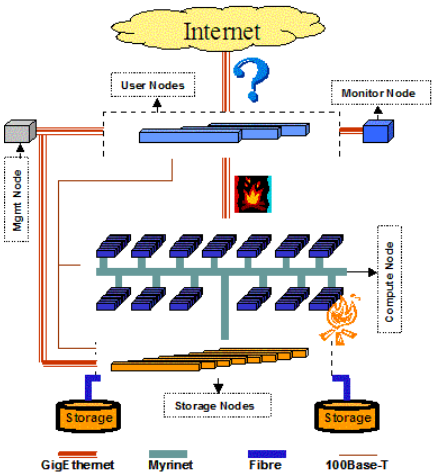

Figure 3. General Cluster Architecture

Figure 4 shows the NVisionCC system architecture. Process information is collected from each node by *clumond* and stored into a MySQL database. The process monitor fetches the process data from this database for analysis and puts the results back into the database. Users have a central management console for process monitoring by accessing a web server which retrieves security logs from the database directly.

In addition, the cluster monitoring community needs to accept the tool and incorporate it into their security monitoring practices. A modular design will also provide a low-hassle upgrade path for clusters already running Clumon. Confining security analysis to a single module also eases design. We envision our tool performing a series of passes over various data metrics collected from the cluster. Leveraging Clumon allows NVisionCC to depend on the data collection and logging facilities of that tool, while allowing us to focus on effective analysis and display of security information.

In order for any security tool to be successful, it must first be effective and easy to use. Figure 5 shows the NVisionCC Main Process View for a 160 node cluster. NVisionCC analyzes the processes found on each cluster node with a profile for the class of each node (head, compute, storage, etc…). There are four types of process alerts: (1) a denied process is running, (2) an unknown (suspicious) process is running, (3) the number of instances of a certain process is out-of-range, and (4) a process that should be running is not running.

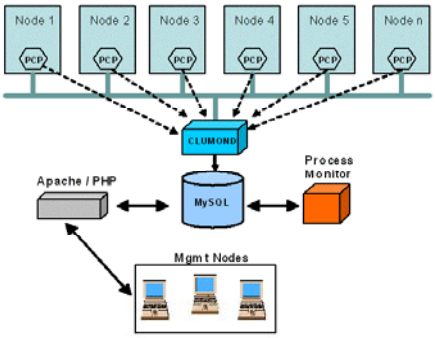

Figure 4. NVisionCC System Architecture

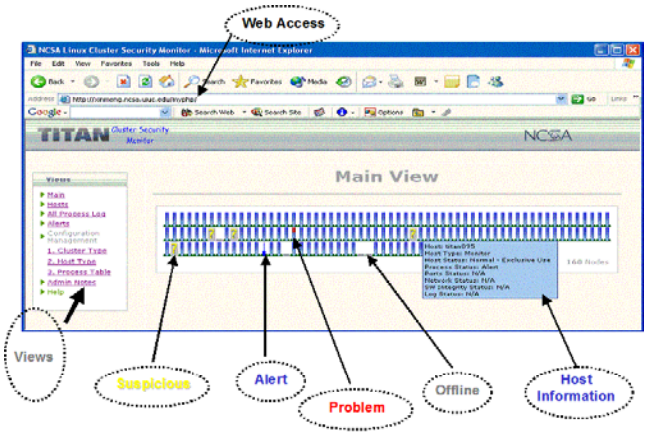

Figure 5. An Annotated NVisionCC Main Process View

The primary visualization design elements we incorporate are:

- All the nodes within an entire cluster are shown on one screen adjacent in space not stacked in time nor scrolled at the bottom.
- Overview of entire cluster with drill down to areas of interest and raw data details on demand at the individual node level.
- Smallest effective differences in shape and color indicate information.
- Different icons show different levels of process security status: critical, bad, suspicious, and normal.

A cluster computing environment is constrained so it is feasible to enumerate/classify the processes we expect on different types of cluster nodes (~40 processes/node at NCSA) and those processes that should not be found (e.g. web). NVisionCC provides different views of a cluster for different purposes. The Process Alert View (Figure 6) provides decision-making assistance about the security tate of a cluster by consolidating alerts from all cluster nodes into a prioritized list for the entire cluster. An individual Host Level View (Figure 7) is available by clicking on a host icon in the Main Process View.

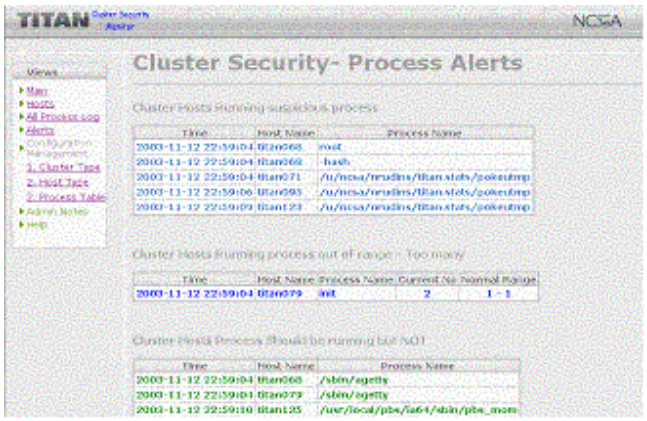

Figure 6. NVisionCC Process Alert View – All Nodes

Figure 7. NVisionCC Process Alert View – Single Host

5. CONCLUSIONS

Good design is clear thinking made visible. We present a framework design for effective visualization of complex security in large cluster environments. While NVisionCC provides compelling pictures, this would be worthless if it did not benefit users in more easily identifying security events as suggested by preliminary results.

NVisionCC has three additional unique characteristics desirable for cluster management not directly related to visualization. First, it is tailored specifically for cluster security. All other cluster monitoring tools are focused on performance. Second, NVisionCC is near-real-time. All data is collected and analyzed at configurable intervals so the security status of a cluster can be constantly assessed. Third, NVisionCC provides archive logging to study transient security events longitudinally. Data is stored in a central database so data can be queried and visualized historically. The desired effect is to fuse data from these heterogeneous sources at different levels into a centralized system for accessing cluster behavior.

As a result of experience we have accumulated during early deployments of NVisionCC, we plan to develop and integrate more security modules into NVisionCC. Four security modules being developed are (1) port scanning, (2) file integrity checking, (3) networking analysis, and (4) log analysis. Port scanning checks ports across all nodes in a cluster. File integrity checking validates installed software and monitors file changes. Network analysis monitors traffic to/from a cluster. Log analysis parses historical data for patterns of security interest. The first two modules have been developed and are currently being testing. We believe fusing all security modules in the NVisionCC visualization framework will increase accuracy, reduce false positives, but most importantly will dramatically increase operational effectiveness.

6. ACKNOWLEDGMENTS

A special thanks to Joseph P. (Joshi) Fullop, IV for partnering with us on this work. We also acknowledge the subsequent contributions of "Cluster-Sec" Team Members Greg Koenig, Adam Lee, and Mike Treaster who have built upon this initial framework. Some figures in the paper are derived from an LCI 2004 conference presentation by Joe Greenseid [13]. Lastly, we thank the following NCSA cluster system administrators for their guidance: (alphabetically) Tim Bovet (Copper), Karen Fernsler (Tungsten), Eric Kretzer (Platinum and Titan), Dan Lapine (Mercury), and Thomas E, Roney (TRECC) along with their Technical Program Manager Wayne Louis Hoyenga.

7. REFERENCES

[1] *Clumon – The Cluster Monitoring System.* <http://clumon.ncsa.uiuc.edu/>

[2] Hacker Attacks Prompt Academic Supercomputers to Block Outside Access Temporarily. *The Chronicle of Higher Education*, April 15, 2004.

[3] Hackers Breach Clusters of U.S. Supercomputers. *Arizona Daily Star*, April 15, 2004.

[4] Hackers Breach Research Systems, But Data Kept Safe. *ComputerWorld*, April 19, 2004.

[5] Gorsuch, N., Linux Cluster Security. *2nd LCI Intl. Conference on Linux Clusters*, 2001.

[6] *Performance Co-Pilot Home Page.* <http://oss.sgi.com/projects/pcp/>

[7] Massie, M.L., Chun, B.N., and D. E. Culler, The Ganglia Distributed Monitoring System: Design, Implementation, and Experience. *Parallel Computing*, Vol 30 Issue 7, 2004.

[8] *RRDtool Home Page*. <http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/>

[9] Roney, T., A. Bailey, and J. Fullop, Cluster Monitoring at NCSA. *2nd LCI Intl. Conference on Linux Clusters*, 2001.

[10] Sottile, M.L. and R.L. Minnich, Supermon: A High-Speed Cluster Monitoring System. *IEEE Cluster Computing Conference*, 2002.

[11] UC Berkeley Millenium Project. <http://www.millennium.berkeley.edu/millennium/>

[12] Yurcik, W., Meng, X. and G.A. Koenig, A Cluster Process Monitoring Tool for Intrusion Detection: Proof-of-Concept. *IEEE Local Computer Networks Conference (LCN),* 2004.

[13] Yurcik, W., Koenig, G.A., Meng, X., and G. A. Koenig, Cluster Security as a Unique Problem with Emergent Properties: Issues and Techniques. *5th LCI Intl. Conference on Linux Clusters*, 2004.