Securing the PlanetLab Distributed Testbed

How to manage security in an environment with no firewalls, with all users having root, and no direct physical control of any system

Paul Brett, Mic Bowman, Jeff Sedayao, Robert Adams, Rob Knauerhase, and Aaron Klingaman
– Intel Corporation

ABSTRACT

PlanetLab is a globally distributed network of hosts designed to support the deployment and evaluation of planetary scale applications. Support for planetary applications development poses several security challenges to the team maintaining PlanetLab. The planetary nature of PlanetLab mandates nodes distributed across the globe, far from the physical control of the team. The application development requirements force every user to have access to the equivalent of root on each machine, and use of firewalls is discouraged. If an account is compromised, PlanetLab administrators needed a way to track the actions of users on the nodes. If an entire node is compromised, then the administrators need a way to regain control despite the lack of physical access. Encryption was built into PlanetLab to ensure confidentiality and integrity of system downloads. A special reset packet, combined with keeping a boot CD in the machine, enables PlanetLab system administrators to remotely regain control of machines if they are compromised and return to the nodes into a safe known state. The Linux VServer implementation is used to provide root access to PlanetLab users for development purposes while isolating users from each other. A network abstraction layer provides accounting of traffic and allows safe access to raw sockets. These mechanisms have proven very useful in managing PlanetLab. After a compromise of large numbers of PlanetLab hosts, control of the PlanetLab network was regained in 10 minutes. The compromise spawned a review of PlanetLab security, which pointed out a number of flaws. The need for a central site for maintaining PlanetLab was cited as a key weakness. Future work includes distributing the functions of PlanetLab's central administrative database and improving integrity checks.

Introduction

The PlanetLab distributed system testbed [1] has a number of unique attributes that make security administration difficult. PlanetLab is a globally distributed network of hosts designed to support the deployment and evaluation of planetary scale applications by distributed systems researchers all over the world. To support their application development efforts, PlanetLab users need root access. PlanetLab systems are used for a variety of network experiments and require unfettered access to and from the Internet. Use of firewalls between PlanetLab hosts and the Internet is strongly discouraged. Moreover, PlanetLab systems are at sites distributed all over the planet and are not under the direct physical control of any of the PlanetLab administrators. At the same time, PlanetLab nodes must be available and usable by the researchers while site and PlanetLab administrators need to be able to respond to security complaints.

This paper describes the security challenges faced by the PlanetLab administration team. It reviews the issues that needed attention, how we dealt with them, and our experiences with the implementation.. We first describe the PlanetLab environment and the system and network requirements of the PlanetLab community. We then describe our design and the implementation of that design. After that, we review our experiences with this design, followed by sections on related and future work.

Environment and Problem Description

In this section, we describe the PlanetLab environment and talk about the key security challenges facing PlanetLab administrators.

PlanetLab Environment

PlanetLab [1] is a distributed-systems testbed, allowing the research, development, and prototyping of new applications and network services. The testbed comprises 433 nodes at 194 sites.[1] PlanetLab is truly "planetary scale" as it is geographically spread across five continents and topologically spread across the Internet, Internet2, and other networks. Because of this geographic and network diversity, the test bed provides researchers with a very "real-world" set of opportunities and challenges; specifically it allows the deployment of, experimentation with, and test/measurement of services in a non-simulated network.

PlanetLab nodes are computers (PCs that meet an evolving set of minimum configuration requirements)

[1]As of the time of this writing; growth continues.

which are hosted by a variety of universities, corporations, and other organizations. The nodes are dedicated running PlanetLab. They run a special version of Linux (detailed later in the paper) and are administered remotely (including patching) by a set of administrators focus on PlanetLab. The set of initial test bed machines was seeded by a grant from Intel Corporation, but now new member organizations must contribute nodes as a condition of joining the test bed. Hosting organizations provide electrical power, physical space, and network connectivity to their nodes. PlanetLab administrators have physical access to almost none of the nodes, and the turnaround on physical work requests on the machines is on the order of days. Administrative and system burden on the hosting organization is deliberately limited (we don't require remote consoles, for example), in order to make joining the testbed as painless as possible.

Security Challenges

Because of the nature of the research done on PlanetLab – requiring unfettered access to the network and frequently resulting in non-standard traffic patterns – the nodes are generally positioned outside of the hosting organization's firewall. A consequence of this is that nodes are typically not protected by any kind of filtering of inbound traffic, and lack of the outbound filtering permits all kinds of traffic, some of which will be interpreted as hostile, to be sent out.

PlanetLab users perform distributed systems research. Accomplishing this frequently requires great flexibility on the part of the system – for example, requiring root access to perform certain functions, or wishing to use the system in odd ways (or replace part of the system with their own code). At the same time, the node must remain stable enough for use. Moreover, researchers don't want other experiments affecting their experiment (as well as the converse).

PlanetLab nodes are administratively very complex – they consist of machines in different networks and administrative domains, providing access to researchers who are at arbitrary locations in other administrative domains and who do odd, non-standard things. Keeping control of the nodes is a difficult problem. That control rests with a centralized group of PlanetLab system administrators who develop and maintain the base operating system (including patches for security and for PlanetLab functionality) and the associated management utilities.

Host organizations also frequently have requirements that they be able to control nodes on their network. One concern that host organizations have is that PlanetLab nodes would be used for nefarious purposes. Sites would need ways to audit PlanetLab usage to help them deal with any possible complaints that received about PlanetLab node behavior. Despite the trust that sites have shown by hosting PlanetLab nodes, we anticipated that at some point PlanetLab nodes could be compromised. We needed a mechanism be able to remotely regain control of hosts when this happened. Nodes would need to be brought to a safe known state for forensics and for removal of vulnerabilities. Also, the ability to remotely power cycle a node would not be enough. While that functionality is extremely useful for remote managing machines when they get hung, it does not implicitly put the PlanetLab node into a state where it can be debugged remotely.

Summary

The security key problems faced are summarized as the following:

- Create a full and rich development environment where users have tremendous flexibility while being isolated from each other and the native OS environment.
- Make it possible, even comfortable for sites to host PlanetLab nodes despite possible complaints about node behavior and the fact that the local site does not fully control the node
- Be able to regain control of PlanetLab nodes even if they are compromised.

Related Work

There are a number of large distributed/network testbeds that deal with similar issues. Emulab [2] is a network testbed that has many of the same concerns as PlanetLab. Emulab uses the FreeBSD jail [3] to isolate experiments in a type of virtual machine. PlanetLab differs from Emulab in that PlanetLab emphasizes the development of services and APIs and also aims to be a deployment platform for services. Also, while Emulab nodes talk primarily to each other, PlanetLab nodes are encouraged to and often do communicate with non-PlanetLab nodes, making the need for an audit trail of PlanetLab node more critical.

As we describe later, PlanetLab uses a virtualization technology to isolate users from each others while giving them a very flexible environment, not unlike Emulab's use of a chroot jail. Related work in the virtualization area are Xen [4] and Denali [5]. A number of PlanetLab nodes are even running on top of Xen.

Other work has been done to manage an environment where users have tremendous flexibility and need the equivalent of root. Leon, et al., [6] discuss how they manage an environment where all users have root. Like the environment described, PlanetLab takes advantage of having sophisticated users who are willing and capable of managing their own environment. PlanetLab is not intended as a desktop environment where users perform activities such as receive mail.

The Grid has been compared to PlanetLab in [7]. PlanetLab differs in that it is more network centric vs. compute-centric than the Grid. Many PlanetLab applications, such as network measurement [8] and content

[1]The PlanetLab consortium is hosted by Princeton University, and their staff serve as centralized PlanetLab system administrators.

distribution [9, 10], rely on geographic and network dispersal to be effective, while rarely being CPU intensive. Dispersal of Grid resources tend to be more accidental than intentional. Also, Grid resources are shared (compared to dedicated PlanetLab node) and typically more heterogeneous than PlanetLab nodes and are not centrally managed like PlanetLab. The network-centric application mix of PlanetLab, particularly with network measurement and content distribution, makes the audit trail requirement more pressing.

Security Design and Implementation

There are many points of control that must be managed to provide top-to-bottom control of the system. The first are the users who must use the system's resources in a reasonable way, as not to provoke sites into removing the hosts. Next, resource utilization, such as network, CPU, and disk space, for each user must operate within certain bounds. The platform must be remotely controlled. The booted operating system and base execution environment must be installed securely and controlled remotely. This section describes each of these pieces and how they are controlled and secured.

AUP

The PlanetLab users operate within the limits of a published Acceptable Use Policy (AUP). All PlanetLab users get access to PlanetLab by first creating an account at PlanetLab Central. One step in this signup is the user's acceptance of the AUP. Since PlanetLab is a testbed for experiments in new Internet technologies, it is difficult to enumerate specific limits within which users must operate. Of course, malicious activity, attempts to subvert the PlanetLab security and authorization system, illegal activities, excessive node use and activities that exceed the usual limits of network propriety are called out, but the general rule is "do no harm." The AUP instructs PlanetLab users to ask what activities would cause network and resource alerts in their own site and then consider the same sort of limits on the remainder of the PlanetLab modes.

User Isolation

To provide a rich development environment to users yet provide user isolation, we modified the base OS of PlanetLab nodes [11]. PlanetLab administrators use a lightweight virtual machine abstraction provided by the Linux VServer [12] implementation. Each research group getting access to a node receives a chrooted virtual Linux machine, which we will call a vserver. The user API effectively becomes Linux. VServers virtualize machines at the system call level, above the kernel. Virtualizing at this level allows us to scale to 1000 virtual machines at the cost of weaker isolation, something not possible with other VMM implementations like VMWARE [13] or Xen [4]. To use the PlanetLab network, researchers get "slices" of the infrastructure. Slices are collections of accounts on some set of nodes across the network. These accounts on a node are isolated within vservers, with the exception of some administrative slices.

Network isolation is achieved through a "safe raw sockets" implementation [14], part of the SILK package, which is derived from Scout [15]. This implementation provides controlled access to the network stack by what appears to be raw sockets without granting root privilege. It also isolates traffic, preventing individual virtual machines from snooping on each other's traffic. In addition to isolating network traffic, SILK provides CPU guarantees and enforces usage policy. The Linux Traffic control facility [12] is used to manage the bandwidth utilization and implement bandwidth policies. We allow site administrators to set the amount of bandwidth that each PlanetLab node can use.

SILK also provides network traffic auditing capability. SILK tags each packet with the ID of the VServer that sent it and provides an administrative port for snooping outgoing traffic. We also created blacklists that would prevent a PlanetLab node from contacting some set of IP addresses. PlanetLab administrators install these blacklists, and local site administrator can request that nodes or networks be placed on them. Care needs to be taken in the installation of blacklists to prevent nodes from being made totally inaccessible.

Reporting

PlanetLab's geographic distribution makes it ideal for mapping the Internet. It seems that many researchers first build a "Hello world" application that pings other PlanetLab and non-PlanetLab nodes to discover timing and connectivity information. Repeated pings, IP address space scanning and port scanning are just the activities that set off Snort [16], and other network monitoring tool alarms. Even some "well designed" probing applications (i.e., with built in flow restriction to avoid complaints) have set off alarms. This implies that some sites have very tight restrictions on probing and mapping activities.

To handle an inappropriate traffic incident, we need to map the reported activity from the network traffic to the experimenter. A traffic report usually contains a time and a source and destination IP address. Additionally, traffic reports relate to an incident in the past. We found that most conventional traffic monitoring tools are for watching current traffic and not recording and querying past traffic.

These problems (mapping, delay and distribution) led to the development of tools which have each node collecting information on its own network traffic (in and out), saving that information and eventually reporting that information to a central repository.

As mentioned above, the kernel's network stack was enhanced with SILK to return information on which IP addresses to which the slivers were communicating. An administrative application named "netflow" analyzes this information every five minutes and calculates the ''flows'' – the connections from a source to a destination by some slice. This information is saved to a file. These files are kept on the node and are eventually copied to PlanetLab Central where they are available for analysis if problem reports arrive.

This flow information is then made available on each node and from PlanetLab Central. Each PlanetLab node runs a web server on the standard web port (80) that gives information about PlanetLab, about the node and allows browsing through the flow information. This allows local administrators of sites hosting PlanetLab nodes to respond to traffic and security alerts. Through the web page, an administrator can search for the reported destination IP addresses and trace this to the email address of the researcher. If a remote (not at the PlanetLab site) network administrator receives reports about the traffic from a PlanetLab node, that administrator can contact the experimenter directly. In this way, the reporting facility removes PlanetLab administrators from the chain of contacts regarding a perceived incident, reducing the time to respond to security complaints. Given the exposure of this web server (no passwords or accounts are needed to access it), web pages are implemented in simple HTML (no java, javascript, or PHP) with no user text input required for selecting looking at traffic patterns.

The requirement on the PlanetLab infrastructure for providing this service is maintaining a mapping between service operators (the researchers). This means a verifiable system of user identities and the monitoring system that records the user of resources and who is using them. PlanetLab thus has a extensive system of monitoring resource use and this monitoring system is tied into a system that authenticates users and which provides a path back to the email of a responsible person for any resource use.

Preventing and Dealing with Compromises

We knew that there was a substantial chance that our exposed network of nodes could be compromised. To deal with that possibility, we configured our machines to boot only from a CD in a machine. Once the machine boots, it downloads via an SSL secured connection a gpg signed script to execute as the next phase of the boot process. These scripts are used for remote re-installation, normal booting, and placing the node into a ''debug'' mode in which the network stops all traffic except ssh connections. Since scripts for normal reboot are downloaded from a central location, we can upgrade the kernel versions used on the hosts without having to update the CD. During debug mode, should the connection to the PlanetLab central website become unavailable for any reason, the node will reboot and retry the connection at 15 minute intervals. With the debug mode, we can bring nodes into a safe known state while preserving disk information for forensics.

The Linux kernel on PlanetLab nodes has been modified to reboot when it receives an ICMP trigger packet with a unique 128 bit payload which is generated for each machine and is re-generated each time a machine reboots. We choose 128 bits to make exhaustive search attacks very difficult against a single node. Since each node has a unique packet for reboot, replay attack is ineffective against the nodeentire PlanetLab network. At worse, a replay attack will only cause a single node to be rebooted. If desired, the machine can be forced to come up into a special debug mode, to which as described above, limits access while allowing for forensics. While effective, this software reboot mechanism suffers from the problem that the machine must have a working network stack, and connectivity to the internet in order to ensure a reboot. With the widescale filtering of ICMP traffic following the SQL Slammer worm, we now recommend that PlanetLab sites install remote power switches on their nodes.

Experiences

Many of the security features implemented in have proven very useful. Our reporting mechanisms have defused many incidents after a network experiment triggered an overly sensitive Intrusion Detection System (IDS). Remote control and access made recovery from a system compromise quick and effective. This section will describe some of the incidents and successes of the PlanetLab security and control mechanisms.

User Behavior and AUP

Since the current direct users of PlanetLab are researchers who generally understand operation on the Internet, there have not been many incidents that required enforcement of the Acceptable Use Policy (''AUP''). We have not yet had to revoke any access from user, which would be the ultimate penalty associated with AUP violation.

Problems with PlanetLab user behavior have been studied [17] and fall into two categories: program failure and accidental network traffic alerts. Building distributed, decentralized applications is hard and, of course when you have lots of projects building them, there will be bugs. PlanetLab Central will receive reports or will notice excessive node resource use (e.g., no file descriptors) or excessive network traffic (e.g., too many external computers accessed or excess volume) and PlanetLab Central sends email to the researcher. In all cases, the researchers have responded to the situation.

Measuring the Internet generates lots of probes and pings. A simple mapping experiment, generating a small amount of data and performing a straightforward measurement set off alarms at many locations. In this case, and in others, a measurement experiment has the same network traffic profile as a worm looking for hosts to infect (probing port 80 is a feature of CodeRed/NIMDA). It is against the PlanetLab Acceptable User Policy to generate ''disruptive'' network traffic but it's sometimes hard to know what type of traffic would be considered disruptive.

There have been many incidents of "attack" or "worm" reports that were traced back to a measuring or topology experiment. The resource monitoring system allows forwarding of the reports to the researchers and, in all cases, the researchers responded appropriately to the situation.

Monitoring

We put a lot of energy into providing ways for network and security administrators to determine for themselves what researcher generated traffic and how to contact them. While these facilities did reduce the workload from dealing with security complaints about PlanetLab node behavior, we continue to have problems from overzealous intrusion detection systems. Some organizations set up intrusion detection systems (IDS) to trigger on relatively innocuous things as a traceroute. One organization went as far as to threaten lawsuits if behavior persisted, and this tactic proved successful in getting a number of PlanetLab hosts pulled off networks. Sometimes complaints were justified, as some researchers experiments generated what would have to be interpreted as an attack – large numbers of connections attempted to a range of IP addresses in a domain. In any case, we anticipate that poor experiment behavior and overly sensitive IDS will continue to cause problems.

Compromise and Recovery

We had anticipated that at some point, PlanetLab nodes would be compromised, and we did have an incident where large numbers of PlanetLab nodes were compromised. The early implementation of PlanetLab had accounts that were not virtualized – they had access to the native operating system. An SSH key to a non-virtualized account was compromised, and that key was used to log into a number of nodes. Since the account was not isolated within a VServer, the attacker used his access to the native operating system obtain root. When we received notice that a number of PlanetLab hosts had been rooted, we used the reboot feature of PlanetLab nodes to force all of PlanetLab into a known safe state in 10 minutes.

We took a number of actions in response to the compromise. Forensic analysis, enabled by debug mode, determined that the nodes were rooted using a vulnerability that we had plans to patch. We had just begun to roll out a version that was not vulnerable to the exploit when we were attacked. We also eliminated general purpose slices that were not isolated with VServer accounts. At the same time, we made all user slices dynamic and eliminated static VServer slices. Slices would not be assigned by default to all nodes, which would limit the access to nodes if a slice private key were compromised. In addition, slices would have a finite lifetime. They would not last indefinitely, and would need to be renewed. This idea brings us closer to the idea of least privilege for slices – slices would only be instantiated on nodes that they needed and only for as long as they were needed.

A Security Review

We had fixed the more obvious problems, but what about problems we had not yet anticipated? Another response to the incident was to have more eyes looking at the problem, so we conducted a review of PlanetLab security. We wrote a summary of PlanetLab's architecture and implementation and had it reviewed by a variety of security researchers and practitioners and PlanetLab users. This review proved quite valuable, finding a variety of vulnerabilities and areas of improvement at both the implementation and architecture levels.

One key problem found was the dependency on a single instance of PlanetLab Central for key PlanetLab's operations, such as slice creation and deletion and software updates. A compromise there would lead to a compromise of all PlanetLab nodes. A DOS attack on PlanetLab Central, while not rendering PlanetLab unavailable, would make many key PlanetLab functions unusable.

Another problem regarded resource management. We need better ways to monitor and manage resources such as slices, CPU, disk, and bandwidth utilization. A runaway process could render nodes unusable and take up so many resources that it would be nearly impossible to log in and fix the problem. In addition, much of resource management and the security associated with it is hard to use. Slices can only be created the principal investigator at site. As a site PI this is usually a busy professor, this leads to a tendency of the PI to share his password, and we have evidence of this. Also, the dynamic slice mechanism provides no warning when slices will expire and all of our user's work will disappear. As a result, there is a perverse incentive to create slices that live as long as possible. Some of PlanetLab users had a contest to see who create the longest lived slice. Many of the reviewers mentioned that security that is hard to use will usually be worked around, as demonstrated.

Related to the resource management issues is the need for better intrusion detection and prevention. While we have worked to improve the isolation of slices from each other and then real operating system, if a PlanetLab slice is compromised, the attacker has a large amount of resources available to him. We need ways to detect resource misuse and intrusion. Also, we need better ways to authenticate and authorize users. Relying on a single database of information run by a single organization to authenticate and authorize users is not likely to scale or be secure. As the number of users and organizations using PlanetLab increases, it is unlikely that PlanetLab administrators could revoke them when those users leave an organization. Instead, a federated scheme, where access is granted to some institutions and those individual institutions manage who has valid access, is more likely to be successful in the long term. PlanetLab does not allow easy ways for slices to authenticate and authorize each other. As a result, some users have poor practices such as leaving SSH private keys on nodes. One reviewer pointed out the use of any reusable password is not vulnerable to attack. The attacker who compromised PlanetLab set up sniffers that listened not to network interfaces but to TTY ports. In this case, SSH does not help as data streams are decrypted when the attackers are listening to them, so if users actually did put pass phrases on their SSH private keys, those pass phrases could be compromised.

Future Work

A large focus of future work is on the dependencies on a single centralized facility for much of PlanetLab's operations. Key dependencies are in PlanetLab Central are being analyzed, and a more formalized threat assessment matrix will be created. Creation of a Red team for more formally and thoroughly analyzing security weaknesses is also being considered. Integrity checkers such as chkrootkit [18] and rkdet [19] and other IDS like features are being evaluated and tested. Longer term, the architecture for PlanetLab management is being studied to make it more secure and scalable.

Conclusion

PlanetLab's security mechanisms have worked relatively well so far. The VServer mechanism effectively gives PlanetLab users a whole virtual machine to use and configure while isolating them from each other and the native operating system. The PlanetLab user account system allows network and security administrators a way to determine the source of problematic traffic. While PlanetLab has hosted hundreds of projects and researchers, and a major compromise was dealt with swiftly and effectively using PlanetLab's reboot mechanisms. A review of PlanetLab's architecture and implementation has yielded a number of areas of improvement, such as the vulnerability of having a single point of control, the need for better resource management, and the need for improvements in authentication and authorization.

Author Information

Jeff Sedayao is a staff engineer in the Planetary Services Strategic Research Project and in Intel's IT Research Group. He focuses on applying PlanetLab and PlanetLab developed technologies to enterprise IT problems. Sedayao joined Intel in 1986, where he designed and implemented almost all aspects of Intel's Internet connectivity, including routing, firewalls, mail, proxying, and DNS. After leaving Intel's IT organization, he worked in Intel's Online Services venture, designing firewall configurations, managing network services, and providing consulting services on security, mail, and DNS. Sedayao has participated in IETF working groups, published papers on policy, network measurement, network and system administration, and authored the O'Reilly and Associates book, *Cisco IOS Access Lists*. He has recently written PlanetLab Design notes on port usage and IP address usage.

Mic Bowman is a senior researcher within Intel's Virtualization Platform Lab. He received a Ph.D. in Computer Science from the University of Arizona. Bowman joined Intel's Personal Information Management group in 1999. While at Intel, he has developed personal information retrieval applications, context-based communication systems, and middleware services for mobile applications. He is currently a Principal Investigator for Intel's Planetary Services Strategic Research Project. Prior to joining Intel, Bowman worked at Transarc Corporation, where he led research teams that developed distributed search services for the Web, distributed file systems, and naming systems.

Paul Brett joined Intel in 2000 as part of Intel's Online Services group. He is currently focused on PlanetLab, a global test bed for developing, deploying and accessing planetary-scale services. From 1988 to 2000, Brett worked on the design and implementation of dependable systems for air traffic control. He is a graduate of the UKs Open University, where he earned a First Class Honours degree in systems engineering of software- based systems.

Rob Knauerhase is a staff research engineer in the Planetary Services Strategic Research Project. He joined Intel in 1993 and has been involved in the research and development of mobile networking, handheld/mobile computing, distributed computing, Internet technologies and middleware, and static and runtime compiler environments. He holds 11 patents, with approximately 60 more pending. Rob is a Senior Member of the IEEE and IEEE Computer Society and has been an adjunct professor at Portland State University.

Aaron Klingaman is a member of the research staff at Princeton University. He joined Intel in 2001 after receiving a B.S. degree in Software Engineering, with honors, from the Oregon Institute of Technology. His initial research focused on interactive television. His current interests include management of remote distributed computing and network resources. Extracurricular projects include work in the areas of firmware development and fractals. Klingaman currently works on supporting and developing PlanetLab's infrastructure.

Robert Adams began working with computers long before he earned a B.S. in Computer Science at Oregon State University. After receiving his degree, he spent ten years in Silicon Valley, where he worked on multi-processor operating systems for large and small computer systems. Adams joined Intel in 1986, initially writing drivers for the first port of Unix to Intel's 386 processor. Later, he moved to Intel's new Multimedia Systems Technology Group, where he was a principal developer of video and audio conferencing systems, eventually winning Intel's Achievement Award for leading the team that wrote the company's first software video codecs. He also has worked in Intel's Architecture Laboratory developing multiple technologies, including cable modem networking, statistical text analysis, peer-to-peer networking and Web infrastructure. Adams holds more than 18 patents and has served on various Internet standards committees. His current work focuses on the deployment of PlanetLab, a global test bed for developing, deploying and accessing planetary-scale services.

References

[1] Peterson, L., T. Anderson, D. Culler, and T. Roscoe, "A Blueprint for Introducing Disruptive Technology into the Internet," *Proceedings of HotNets I*, October 2002.

[2] White, Brian, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar, "An Integrated Experimental Environment for Distributed Systems and Network," *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation*, December 2002.

[3] Kamp, P. H., and R. N. M. Watson, "Jails: Confining the Omnipotent root." *Proceedings of the Second International SANME Conference*, May 2000.

[4] Barham, Paul, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Hegebar, Ian Pratt, and and Warfield, "Zen and the Art of Virtualization," *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, October 2003.

[5] Whitaker, Andrew, Marianne Shaw, and Steven D. Gribble, "Denali: A Scalable Isolation Kernel," *Proceedings of the Tenth ACM SIGOPS European Workshop*, September 2002.

[6] De Leon, Laura, Mike Rodriguez, and Brent Thompson, "Our Users Have Root!" *Procedings of the Seventh Large Installation System Administration Conference (LISA '93)*, November 2003.

[7] Ripeanu, Matei, Mic Bowman, Jeffrey Chase, Ian Foster, and Milan Milenkovic, "Comparing Globus and PlanetLab Resource Management Solutions," PDN-04-018, February 2004.

[8] Spring, N., D. Wetherall, and T. Anderson, "Scriptroute: A Public Internet Measurement Facility," *USENIX Symposium on Internet Technologies and Systems*, 2003.

[9] Wang, L., K. Park, R. Pang, V. Pai, and L. Peterson, "Reliability and Security in the CoDeen Content Distribution Network," *Proceedings of the USENIX 2004 Annual Technical Conference*, June 2004.

[10] Freedman, M., E. Freundenthal, and David Mazieres, "Democratizing Content Publication with Coral," *Proceedings of NSDI '04: First Symposium on Networked Systems Design and Implementation*, March 2004.

[11] Bavier, Andy, Mic Bowman, Brent Chun, Scott Karlin, Steve Muir, Larry Petersen, Timothy Roscoe, Tammo Spalink, Make Wawrzoniak, "Operation System Support for Planetary-Scale Network Service," *Proceedings of NSDI '04: First Symposium on Networked Systems Design and Implementation*, March 2004.

[12] *Linux VServers*, http://www.linux-VServer.org .

[13] *VMWARE*, http://www.vmware.com/ .

[14] *Safe Raw Sockets*, http://www.planet-lab.org/raw_sockets .

[15] Mosberger, D. and L. Peterson, "Making Paths Explicit in the Scout Operating System," *Proceedings of the Second OSDI Conference*, Oct 1996.

[16] Roesch, Martin, "Snort – Lightweight Intrusion Detection for Networks," *Procedings of the Thirteenth System Administration Coference (LISA '99)*, November 1999.

[17] Adams, Robert, "Distributed System Management: PlanetLab Incidents and Management Tools," PDN-03-015, November, 2003.

[18] Murilo, N. and K. Steding-Jessen, *Chkrootkit*, http://www.chkrootkit.org/ .

[19] *rkdet*, http://vancouver-webpages.com/rkdet/ .