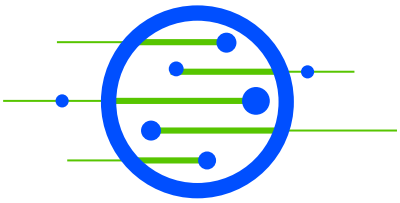

The PlanetLab Boot Manager

Aaron Klingaman
Princeton University

PDN–05–026
January 2005

Status: Ongoing Draft.

The PlanetLab Boot Manager

Aaron Klingaman <alk@cs.princeton.edu>

Abstract

This document outlines the design and policy decisions of a new PlanetLab component called the Boot Manager. The Boot Manager encompasses several systems and all policy regarding how new nodes are brought into the system, how they are authenticated with PlanetLab Central (PLC), what authenticated operations they can perform, and what constitutes a node's identity.

Table of Contents

1. Overview

This document describes the history of and groups several previously separate, undocumented components and policy decisions of the PlanetLab infrastructure into one logical group, which will be called the *Boot Manager*. In addition, specific recommendations are made for changes and additions to these parts to support new features and better security outlined in detail later. These include:

1. How new nodes are added to the PlanetLab system, and the chain of trust that accompanies that addition
2. How to prevent unauthorized nodes from becoming part of the system, and the consequences of that happening
3. How any existing node authenticates itself with PlanetLab Central (PLC), and what operations can it perform
4. What constitutes node identity, and, when this identity should and should not change

Not covered by this document are topics including node to node authentication, or any service or system running after a node is fully booted and the Boot Manager is no longer applicable.

2. Terminology

Before continuing, terms used through this document, including what a site is, what nodes are, and what PlanetLab consists of will be defined. Current organizational structure consists of groups of *sites*, usually a geographical location corresponding one to one with a company or university. These sites have any number of *users* or *researchers*, including a *principle investigator* , or *PI*, responsible for the users, and one or more *technical contacts*. Sites are usually composed of at least two machines running the PlanetLab software, usually referred to as *nodes*. All user and node management operations are done through a set of servers located in one physical location which is known as *PlanetLab Central*, or *PLC*.There are also a set of PlanetLab *administrators*; not necessarily affiliated with a particular site. *PlanetLab* then collectively refers to all sites and their nodes and users, and PlanetLab Central.

3. Background

3.1. How Sites Become Part of PlanetLab

A full discussion and evaluation of the process and security implications of sites becoming part of PlanetLab is outside the scope of this document. It will be assumed that the process is relatively secure, and that user and PI accounts at that site are legitimate. However, it is necessary to provide some basic information about the process.

What does it mean for a site to be part of PlanetLab? Primarily:

1. The site's record (e.g. name, url, geographical location, contact information) is in the PLC database
2. There are a set of users (their email address, password, personal information) associated with the site in the PLC database
3. The ability for those users and PIs to perform some operations at PLC, and gain direct access to the nodes

The process for allowing new sites to become part of PlanetLab has been continually evolving since the beginning of PlanetLab. Initially, the first sites were selected and invited, and record of their existence in PLC was entered in by hand by an administrator. With a site now part of PlanetLab, users and PIs at those sites could then register for accounts to perform operations at PLC. Privileged accounts, such as PI accounts, were enabled by administrators. At the time, this administrative overhead was not a problem given the relatively limited number of total sites.

Over time, parts of these operations have been streamlined. Now, a site can submit all their relevant info on the PLC website, for review and approval by administrators. They also no longer require an explicit invitation. With the creation of the PlanetLab Consortium, there is now an additional paperwork step before a site becomes a member of PlanetLab.

With the introduction of the additional consortium step, the process now exists as:

1. A site either requests to join PlanetLab by contacting administrators over email, or through other external communication
2. Necessary consortium paper work is signed by all parties

3. PI(s) submit connect (join) requests with remaining site and personal information

4. Administrators verify that the PI is who they say they are, and enable their site and accounts at PLC

3.2. How Nodes Become Part of PlanetLab

After a site has been approved and added to PLC, they are required to install and make available to other users at least two nodes (as per current policy).

In the first revisions of the PLC software, nodes were only added to the system by hand. Usually a PI or technical contact would communicate the network settings of the node, and it was then added to PLC by an administrator. This prevented any nodes that weren't part of PlanetLab to be recognized by PLC. No mechanisms existed to ensure that the node's network (effectively its identity) was not hijacked by another machine.

Since the beginning of PlanetLab, there have been little to no restrictions on what machines the PlanetLab software can run on. This is primarily due to the fact that all source code is now available, and it is technically feasible for anyone to bring up a machine that is running the PlanetLab software, or closely resembles it. What is important, however, is when these nodes become recognized by PLC, and then available to the users via PLC. Otherwise, a user would have to go through non-PLC channels in order to find these nodes. Even then, they could not use PLC to run their experiments on the nodes, because PLC does not know about those nodes.

When a node becomes part of PlanetLab, it:

1. Is recognized by PLC as being at the site by its existence in our database

2. The existing node boot mechanisms allow the machine to come online after communicating its identity to PLC

3. Researchers can use the node for their experiments by using administrative interfaces at PLC

Rather than adding each node by hand, the current system instead allows for an entire network subnet to be authorized to contain nodes. When a site joins, a PLC administrator authorizes the subnet the nodes will be on, and any machines on that network are allowed to become recognized by PLC automatically. This had immediate advantages, primarily being one of not requiring overhead for PLC administrators to add each node by hand as was done in the beginning. Given that a common interest was to see PlanetLab grow in terms of number of nodes (as one metric), the assumption was made that allowing any node to come online on an authorized subnet without explicit approval from an administrator or PI would benefit everyone.

3.3. Node Installation

To date, there have been three major revisions of the software that installs a PlanetLab node. Not only have the mechanisms in which the nodes get installed changed, but, under what context the installation is running.

The first revision of the installer was primarily nothing more than a customized RedHat (version 7.3) boot disk, with a PlanetLab specific post script to perform final initialization steps. The network settings, and which packages to install were all stored on the disk, so a custom disk was generated on demand for each node. Anyone with one of these disks could install a PlanetLab node.

The second revision of the installer was released in conjunction the release of the new PlanetLab boot cd. The intention was not necessarily to have the node packages on the cd (as they would quickly go out

of date), but, to provide a mechanism to allow administrators to regain control of a machine, in the event that the node was compromised, or the installed software was corrupted. The nodes were configured to always start off the cd, and, rather than have a custom cd per node, the network settings were stored on a floppy disk. Both the floppy disk and the boot cd were to remain in the machine at all times. The RedHat installer, Anaconda [1], that was used prior to the boot cd was modified to run in the context of this boot cd. This allowed us a great deal of flexibility, as the cd was built so that all it would do was:

1. Bring a full Linux system online, running only off the cd
2. Load any network and other drivers necessary, based on the hardware of the node
3. Configure the network interface with the settings from the floppy disk
4. Contact a special PLC boot server, and download and execute a script.

The boot cd uses HTTPS to contact the boot server, and uses a certification authority (CA) certificate to verify the identity of the machine at PLC. This way, it can be assured that the installation of a particular node is correct, in at least that all packages originated from PLC. The script downloaded by the boot cd for a node depends on the current state of that node, in the PLC database. The PLC database must identify the node in order to accomplish that. That is covered below, in Node Identity.

The third and current version of the installer still runs in the context of the boot cd, but was a complete rewrite to better handle packages, and remove much unneeded complexity in the previous installer.

3.4. Node Identity

In the first revisions of the PlanetLab software, nodes were solely identified by their network settings, primarily, the hostname and the physical address of the network adapter (MAC address). This worked well then, as this set of information was unique, and allowed for the direct mapping of node identity to a physical machine. It was stored this way in the PLC database as well.

As the design of the database progressed, the PlanetLab software needed to identify nodes not by any one aspect of the physical machine, but by a more generic identifier (as this identifier needed to be used internally to refer to other aspects of a node, like which site it is at) - what has been called a node id. Although better in some respects, there are still drawbacks. For example, deleting a node entry from the database and recreating a similar one could result in a new node id, when nothing on the node itself really has changed. These problems are primarily due to a lack of policy being documented, and instead, the implementation details defining the policy.

Currently, when a node requests a script from the boot server as the last step of the boot cd operation, it sends to PLC the output of the program 'ifconfig' (among other data), which contains the network settings the machine was configured with. From the network settings, the primary MAC address is extracted by PLC and used to check the database if the node exists. Here, the MAC address is used to look up a corresponding numeric node id, which is used internally. The MAC address and the node id are tied - if a new MAC address is used, a new node id will be generated. If the node does exist, an appropriate script is sent in response, based on the current node state. Again, this was fine, as long as a node was identified correctly.

3.5. Node Authentication

What does a node (or PI, for that matter) have to do to prove that it is one of the real, or legitimate, PlanetLab nodes? At first, this was not an issue because the nodes were added to the system by administrators, and all communication paths led only from PLC to the nodes. Everything was downloaded from PLC, including information about what experimenters can use the system, what packages to install for updates. For this, a node only needed to send enough information in the request to identify itself with

PLC. From the PLC point of view, it did not matter which node downloaded the packages for a node, so long as the node was identified correctly and received the packages it was supposed to. This was acceptable since the node was added to PLC by hand, thus it was already 'authenticated'. During this period, a number of assumptions were made:

1. That a rogue node with the same network settings would not be a problem, as the site technical contacts could prevent or detect that

2. The ability to check to ensure a particular node was already authenticated was not done (aside from assuring that the host's public ssh key fingerprint did not change from one login to the next)

As more previously manual steps became automated, a number of situations came up in which a node would need to initiate and perform some operation at PLC. There is only a small set of these operations, and are limited to items such as, adding a node to the system (under a previously authorized subnet), changing the 'boot state' (a record of if the machine is being installed, or is in a debug mode) of a node, or, uploading the logs of an installation.

To handle this new node authentication, a 32 byte random nonce value was generated and sent to PLC during node boot time (at the same time the network settings are sent). The nonce value in the PLC database for that particular node is updated if the node is identified correctly, and is used for authenticating subsequent, node initiated operations. Then, for example, when a node install finished, a node could request it's state updated, and all it would need to do would be to resend its network settings, and the original nonce for authentication. If the nonce in the database matched what was sent, then the requested operation was performed.

The problem here is obvious: now, any node that can be identified is essentially automatically authenticated. For a node to be identified, it has to be in the database, and, new nodes can be automatically added on any authorized subnets without intervention of an administrator or tech contact. With this system, it is trivial to add a rogue node to the system, even at a different site that was not originally authorized, because the whole system is based on what a node sends PLC, which is trivial to spoof.

4. Recommendations

4.1. How PLC Will Identify Nodes

Before any suggestions on what to change regarding the node identity policy can me made, the question, what makes a node a node, should be answered. This primarily depends on who is asking. From an administrators point of view, a node could be tied to a particular installation of the software. Reinstall the node, and it becomes a new node with a new identity. However, from an end user's perspective, the machine still has the same network address and hostname, and their software simply was removed. For them, changing the node identity in this situation does not make any sense, and usually causes them unnecessary work, as they have to re-add that machine to their experiment (because, as far as the PLC database is concerned, the node never existed before then). This question is particularly import for several reasons:

1. It gives users a way to identify it, in order to use it for their research

2. The node identity could be used by other external systems, as a universal identifier

The following recommendation is made for a new node identity policy. Rather that tie node identity to some attribute of the physical machine, such as its hardware configuration as is currently, instead, PLC will assign an arbitrary, unused identity to the node upon its creation, and that identity will be stored locally at the node (most likely on an external medium like floppy disk). Then as long as that identity is still

on the node, any hardware or software changes will not necessarily require a change of the node identity. This will then allow PLC, if necessary in the future, to change the node identity policy as needed.

The following policy will apply to this new node identity:

1. In the past, a tech contact was able to change the network settings on a node automatically by updating the network configuration floppy. Now, these changes will have to be done at PLC (with the option of assigning a new node identity). Thus, the node's network settings (excluding MAC address), are tied to the identity.

2. Attempting to move the node identity to another machine will halt that machine from being used by researchers until the change is dealt with by either a PLC administrator or a site technical contact. If approved, the node would reconfigure itself appropriately.

3. A node identity cannot be reused after the node has been deleted from the PLC database.

4. The node identity will not change across software reinstalls, changes of the harddisks or network adapters (as long as the network settings remain), or any other hardware changes.

Given the current design of the PLC database, there is still a need to use, at least internally, a numeric based node identifier. Other software and APIs available to researchers also use this identifier, so the question becomes whether or not the above policy can be applied to it without significantly changing either the PLC software or the researcher's experiments. Answering this question is beyond the scope of this document, and is left as implementation decision.

4.2. Authenticating Node Identity

It is clear that the previous model for authentication will need to change, which assumes with identity comes authorization, to one where a node can present its identity, then authenticate it as a separate step in order to become authorized. During the boot process, a node can still send sufficient information to identify itself, but, a new system is required to prove that what it sends in fact does come from the node, and not someone attempting to impersonate the node. This is especially important as node identities are made public knowledge.

Authentication in distributed systems is a fairly widely researched problem, and the goal here is not to build a new mechanism from scratch, but rather to identify an existing method that can be used to fulfill our requirements. Our requirements are fairly simple, and include:

1. The ability to trace the origin of a node added to PlanetLab, including the party responsible for the addition.

2. Authenticating requests initiated by nodes to change information at PLC. These requests involve little actual communication between the nodes and PLC, and the overhead for authenticating each request is small given the number and frequency of them. This also means the need to open an authenticated channel for multiple requests will not be necessary.

Given the public nature of PlanetLab, the need to encrypt data during these system processes to prevent other parties from seeing it is not necessary (also, simply hiding the details of the authentication process is not a valid security model). Assuring the requests are not modified during transmission is necessary, however. A public/private key pair system could be used, where each site would be responsible for generating a private key, and signing their node's identity. PLC could then have a list of all public keys, and could validate the identities. However, this is not recommended for several reasons:

1. It places an additional burden on the site to generate and keep secure these private keys. Having a private key for each node would be unreasonable, so one key would be used for all nodes at a particular site.

2. By using one key for all nodes, it not only increases the cost of a compromised key (all identities would have to be resigned), but, use of the key to add unauthorized nodes could not as easily be detected.

3. Differences in versions of the software used to generate keys would have to be handling, increasing the complexity of supporting a system at PLC

To fulfill the above requirements for node identity, the recommendation is made to use a message authenticate system using hash functions and shared secrets such as in [2]. In such a system, the shared secret (or refered to as key, but not in the public/private key pair sense), is as simple as a fixed size, random generated number. Of primary importance in such a system is the control and distribution of the key.

Securing a key at PLC is relatively straight forward. Only a limited number of administrators have direct access to the PLC database, so keys can be stored there with relative confidence, provided access to the PLC machines is secure. Should any of these keys be compromised, all keys would need to be regenerated and redistributed, so security here is highly important.

However, securing the secret on the client side, at the node, is more difficult. The key could be placed on some removable media that will not be erased, such as a floppy disk or a small usb based disk, but mechanisms must be in place to prevent the key from being read by anyone except the boot manager and the boot cd processes, and not by any users of the machine. In a situation like this, physical security is a problem. Anyone who could get access to the machine can easily copy that key and use it elsewhere. One possible solution to such a problem is to instead make the key a combination of two different values, one stored on the floppy disk, the other being a value that is only known to the PI, and must be entered by hand for each message authentication. Then, in order to compromise the entire key, not only must the attacker have physical access to the machine, but would have to know the other half of the key, which would not be recorded anywhere except in the PLC database. This ultimately cannot work because of the need for human intervention each time a node needs to be authenticated.

Ultimately, the best solution for the circumstances here is to leave the entire key on the disk; leave physical security to the individual sites; and put checks in place to attempt to identify if the key is being reused elsewhere. As before, the post-boot manager system (running the real PlanetLab kernel), can be configured to prevent the floppy disk from being read by any logged in user (local or not).

If the key was identified as being reused elsewhere, appropriate actions would include deleting the key from the PLC database (effectively halting any use of it), and notifying the technical contacts and PIs at the site. If necessary, they could regenerate a new keys after corrective actions had been taken.

4.3. Adding New Nodes

It is important to have control over the process for which nodes are added to the PlanetLab system, and to be able to derive which party is responsible for that machine at any point in the future. This is because several different parties come to PLC for the list of nodes, and PLC needs to provide a list that only includes nodes that have been authorized. For one, the researchers who are looking to run experiments need to identify a set of PlanetLab machines. Two, non-PlanetLab related people who may have traffic related concerns or complaints, and are trying to track down who is responsible for a node and/or the researcher's experiment.

It is possible to envision at least several scenarios where having a non-authorized node in the PLC database would be a problem. One of which would be a researcher inadvertently using a rogue node (those who installed it could easily have root access) to run an experiment, and, that experiment being compromised across all of PlanetLab, or the results from their research being tampered with. Another could

include a rogue node being used for malicious purposes, such as a spam relay, and the (initial) blame being directed at PLC, simply because of the association.

As shown previously, simply authorizing an entire network is insufficient, as the ability to identify who authorized an individual node on that subnet is unknown. Having the PlanetLab administrators add all nodes by hand incorporates too much overhead, given the number of nodes and the current growth of PlanetLab. This also places the administrators in a state where they may not have the contact information for the responsible party. A decent compromise will be to require either the PIs or technical contacts at each site to enter in their own nodes using the existing PLC interfaces. Given that one of the existing steps for bringing a node online involves generating a floppy-based network configuration file on the PlanetLab website, this process can be extended to also add record of the nodes with little additional impact to PIs and tech contacts. At this point, the per-node shared secret and a node identity necessary for node authentication would be generated and saved at PLC as well.

4.4. How To Remove Nodes

There may be the need for an administrator, PI, or technical contact to remove a node from the system. This can be done simply by removing the node record from the PLC database, thereby preventing it from successfully authenticating at boot time. In addition, a node could be effectively disabled (but not removed), by deleting the private key for that node from the database. Once restarted, it would not be able to come back online until a new key is generated.

4.5. Node Installation

The node installer shall be integrated into the Boot Manager, rather than continue to be a standalone component. This will allow the boot manager, when appropriate, to invoke the installer directly.

5. Conclusion

As outlined above, this new system effectively encapsulates a new policy for node identity, and a new mechanism for verifying the node identity and authenticating node-initiated PLC changes. In total, the boot manager collectively will consist of:

1. A set of interfaces at PLC that are used to perform authenticated, node-initiated changes.
2. A set of interfaces at PLC that are used to add new nodes to the system.
3. A package downloaded by the boot cd at every boot, which used to install nodes, update configurations, or boot nodes, using the interfaces above.
4. The policy for identifying nodes, and when that identity should change.

Given the above recommendations, the boot strap process and the chain of trust for adding a new node now exists as detailed below. A site, a principle investigator, and a tech contact are assumed to be already present, and authorized.

1. The technical contact downloads a boot cd for the new node. Since the HTTPS certificate for the public web server is signed by a trusted third party, the image can be verified by either ensuring it was downloaded via HTTPS, or by downloading the PlanetLab public key and verifying a signed copy of the cd, also available on the website.
2. The now validated boot cd contains the CA certificate for the boot server, so any host initiated communication that is using this certificate on the cd can be sure that the server is in fact the Plan-

etLab boot server.

3. The PI logs into their account on the PlanetLab website, also over HTTPS and verifying the SSL certificates. Once logged in, they use a tool to generate a configuration file for the new node, which includes the network settings and node identity. During this configuration file generation, record of the nodes existence is entered into PLC, and a random, shared secret is generated for this machine. The shared secret is saved in the PLC database, and is also included in this configuration file.

4. Both the cd and the new configuration file (on a floppy disk), are inserted into the machine. The machine is configured such that it always starts off the cd, and never the floppy disk or the machines hard disks.

5. After the boot cd finishes bringing the machine online, loading all hardware and network settings from the floppy, it contacts the boot server using HTTPS and the certificate on the cd, and downloads and executes the boot manager.

6. The boot manager then contacts PLC to get the current state of the node it is currently running on.

7. Based on this state, the boot manager can either continue booting the node (if already installed), install the machine if necessary, or take any other action as appropriate. Since this is a new machine, the installation will be initiated.

8. After successful installation, the boot manager needs to change the state of the node such that the next time it starts, it will instead continue the normal boot process. The boot manager contacts PLC and requests a change of node state. This request consists of the node identity, data pertaining to the request itself, and a message authentication code based on the shared secret from the floppy disk and the request data.

9. The boot manager, in order to authenticate the request, generates its own message authentication code based on the submitted data and its own copy of the shared secret. If the message authenticate codes match, then the requested action is performed and the boot manager notified of success.

10. If the node is already installed, and no actions are necessary, the machine is booted. To protect the shared secret on the floppy disk from users of the machine, the kernel during runtime cannot access the floppy disk. At this point, control of the system is removed from the boot manager and run-time software takes control.

Any action the boot manager may need to take that requires some value to be changed in PLC can use the steps outlined in 8 through 10. As an extra precaution to prevent unauthorized nodes from booting, the process in step 7 should also use the authentication steps in 8 through 10.

Given that the shared secret on the floppy disk can only be accessed in the cd environment (when the boot manager is running and the boot cd kernel provides floppy disk access), any operation that a node can perform that results in a change in data at PLC must be performed during this stage. During runtime, a node can still present its identity to PLC to receive node-specific packages or configuration files, but all interfaces that provide these packages or files cannot change any record or data at PLC.

Bibliography

[1] *Anaconda [http://rhlinux.redhat.com/anaconda]*.

[2] *Message Authentication using Hash Functions - The HMAC construction*. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Spring 1996.