Identity Boxing: Secure User-Level Containment for the Grid

Douglas Thain
University of Notre Dame - Department of Computer Science and Engineering

Today, a public key infrastructure allows grid users to be identified with strong cryptographic credentials and and a descriptive, globally-unique name such as /O=UnivNowhere/CN=Fred. This powerful security infrastructure allows users to perform a single login and then access a variety of remote resources on the grid without further authentication steps. [1] However, once connected to a specific system, a user's grid credentials must somehow be mapped to a local namespace. This creates a significant burden upon the administrator of each site to manage a continuously-changing user list. Large systems have worked around this by employing the old insecure standby of shared user accounts. [2]

Even worse, user identities are not employed consistently across the grid. A single user may be known by a different account name at every single site that he or she accesses, in addition to a variety of identity names given by certificate authorities. In order to access a resource, the user may need to have a local account generated. In order to share resources, each user must know the local identities of users that he/she wishes to share with. However, local identities are often inconsistent or transient, thus preventing any sort of sharing at all.

Ideally, a grid computing system would hide these details from the end user. A user should simply be able to log in and be identified by his or her grid identity without reference to local accounts. If several users wish to share data or resources, they ought to be able to identify each other via their grid identities rather than by arbitrary local names. This ideal is difficult to realize in today's computing systems because of the inflexible nature of the underlying account scheme. Every new user of a grid system must be entered by the administrator into the local account database. Although it is a small burden to do this for one user, it is a full-time job for systems with many thousands of users.

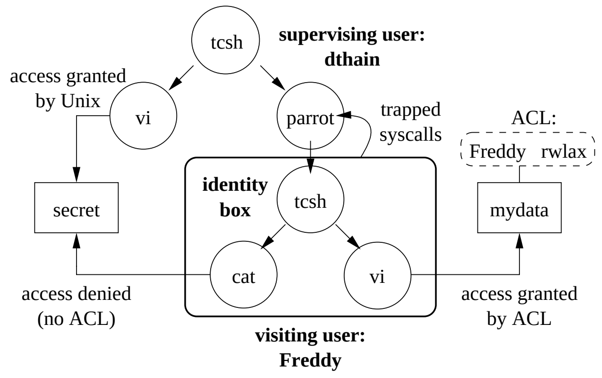
To solve these problems, we introduce the technique of **identity boxing**. An identity box is a well-defined execution space in which all processes and resources are associated with an external identity that need not have any relationship to the set of local accounts. That is, within an identity box, a program runs with an explicit grid identity string rather than with a simple integer UID. As a program executes, all access controls are performed using the high level name rather than the low-level account information. A single Unix account may be used to securely manage several identity boxes simultaneously, thus eliminating the need to services to run as root merely to change identities.

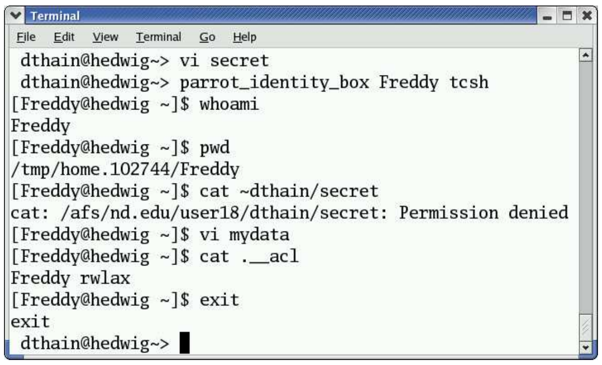
Ideally, identity boxing would be provided by the operating system. However, practical grid computing requires that we live with standard kernels. Thus, we have implemented identity boxing by employing an interposition agent [3]. We have modified Parrot [5] to perform identity boxing by intercepting and modifying system calls through the `ptrace` debugging interface. This allows us to provide identity boxing **securely** at **user-level** on **arbitrary unmodified programs**.

Within an identity box, access control to files and other objects is complicated because grid identities are free-form strings. These do not fit into the existing data structures in the kernel and filesystem that deal with integer UIDs. Our solution to this problem is to abandon the Unix protection scheme and adopt access control lists (ACLs) instead. In each directory, Parrot looks for a file named `.__acl` that describes what actions users can perform on files in that directory. Any program run within an identity box will respect these ACLs. Each entry of an ACL lists an identity and the set of operations that can be performed. Identities may contain wildcards in order to match patterns. For example, this ACL allows /O=UnivNowhere/CN=Fred to read, write, list, execute and administer files in a directory. It also allows any user at UnivNowhere to read and list:

```
/O=UnivNowhere/CN=Fred    rwlax
/O=UnivNowhere/*          rl
```

Visiting users are given a fresh home directory with an appropriate ACL. Newly-created directories inherit the parent ACL. Of course, Parrot cannot





```
 dthain@hedwig~> vi secret
 dthain@hedwig~> parrot_identity_box Freddy tcsh
[Freddy@hedwig ~]$ whoami
Freddy
[Freddy@hedwig ~]$ pwd
/tmp/home.102744/Freddy
[Freddy@hedwig ~]$ cat ~dthain/secret
cat: /afs/nd.edu/user18/dthain/secret: Permission denied
[Freddy@hedwig ~]$ vi mydata
[Freddy@hedwig ~]$ cat .__acl
Freddy rwlax
[Freddy@hedwig ~]$ exit
exit
 dthain@hedwig~>
```

Figure 1. Example of Identity Boxing in an Interactive Session

*An example of identity boxing shown as a schematic and as a shell transcript. The supervising user (*dthain*) creates a file* `secret` *in his home directory. He then creates an identity box for the visiting user* Freddy*, who is not allowed to access* `secret` *because there is no ACL present by default. However,* Freddy *can create a file* `mydata` *in his new home directory, where the ACL has been initialized to give him complete access.*

retroactively place ACLs throughout the file system. When it encounters a directory without an ACL, Parrot enforces Unix permissions as if the visiting user was the Unix user nobody. This ensures that the supervising user's data is protected visitor.

An example of an interactive identity box is shown in Figure 1. Here, the Unix user dthain has created an identity box for Freddy. Note that Freddy does not appear anywhere in the system account list. Freddy attempts to access a file `secret` owned by dthain, but is denied because that file is private to dthain. However, Freddy is given a home directory in which he can work and is allowed to write the file `mydata`. Freddy is also able to set ACLs and share data with other visiting grid users.

A user-level implementation of identity boxing has measurable but not unreasonable overhead. Trapping system calls through the `ptrace` interface increases the latency of system calls by an order of magnitude, due to the increased number of context switches between application, kernel, and agent. Figure 2 shows the overhead of identity boxing on a selection of scientific applications described in an earlier paper. [4] Theses are slowed down by only 0.7 - 6.5 percent. (Although they are more data intensive than other grid applications, they perform primary large-block I/O.) An interactive application such as `make` is slowed down by 35 percent because it make extensive use of small metadata operations such as `stat`. Thus, identity boxing via an interposition agent has overhead that is likely to be acceptable for scientific applications, especially if the technique simplifies security and results in a larger number of resources available to the user.

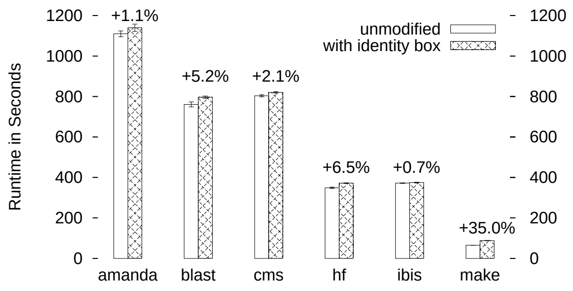


Figure 2. Overhead of Identity Boxing

For a selection of scientific applications, identity boxing imposes an overhead of only 0.7 to 6.5 percent. A syscall intensive `make` *is slowed by 35 %.*

References

[1] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. A security architecture for computational grids. In *ACM Conference on Computer and Communications Security Conference*, 1998.

[2] R. Gardner and et al. The Grid2003 production grid: Principles and practice. In *IEEE Symposium on High Performance Distributed Computing*, 2004.

[3] M. Jones. Interposition agents: Transparently interposing user code at the system interface. In *Proceedings of the 14th ACM Symposium on Operating Systems Principles*, pages 80–93, 1993.

[4] D. Thain, J. Bent, A. Arpaci-Dusseau, R. Arpaci-Dusseau, and M. Livny. Pipeline and batch sharing in grid workloads. In *Proceedings of the Twelfth IEEE Symposium on High Performance Distributed Computing*, Seattle, WA, June 2003.

[5] D. Thain and M. Livny. Parrot: Transparent user-level middleware for data-intensive computing. In *Proceedings of the Workshop on Adaptive Grid Middleware*, New Orleans, September 2003.