Sub-Identity: Security for Mere Mortals

Phil Snowberger and Douglas Thain

Motivation

Users aren't generally capable, or inclined, to ensure the correctness of software that they download and run on their own machines. Users cannot assume any level of security on their desktops unless either they or their administrator takes some measure to compartmentalize, or contain, potentially rogue code.

Current Mandatory Access Control (MAC) solutions such as SELinux are quite technically involved, requiring extensive hand-tuning by security experts. Other solutions such as traditional sandboxing and virtual machines require a high level of user sophistication to operate correctly. As effective as they promise to be, they lack intuitiveness and simplicity.

Concept

- **People have an intuitive understanding of familial and corporate hierarchies, in that they have a sense that higher-ups have control over those lower down.**
- **We propose to introduce *sub-identities* into the operating system. Each new sub-identity has a meaningful name and can be used to enforce access control, perform auditing, or simply isolate sub-processes from each other.**

The Ideal Model, A Lofty Goal

Ideally, every user would be "effectively root" to all of its sub-users. Users should be able to:

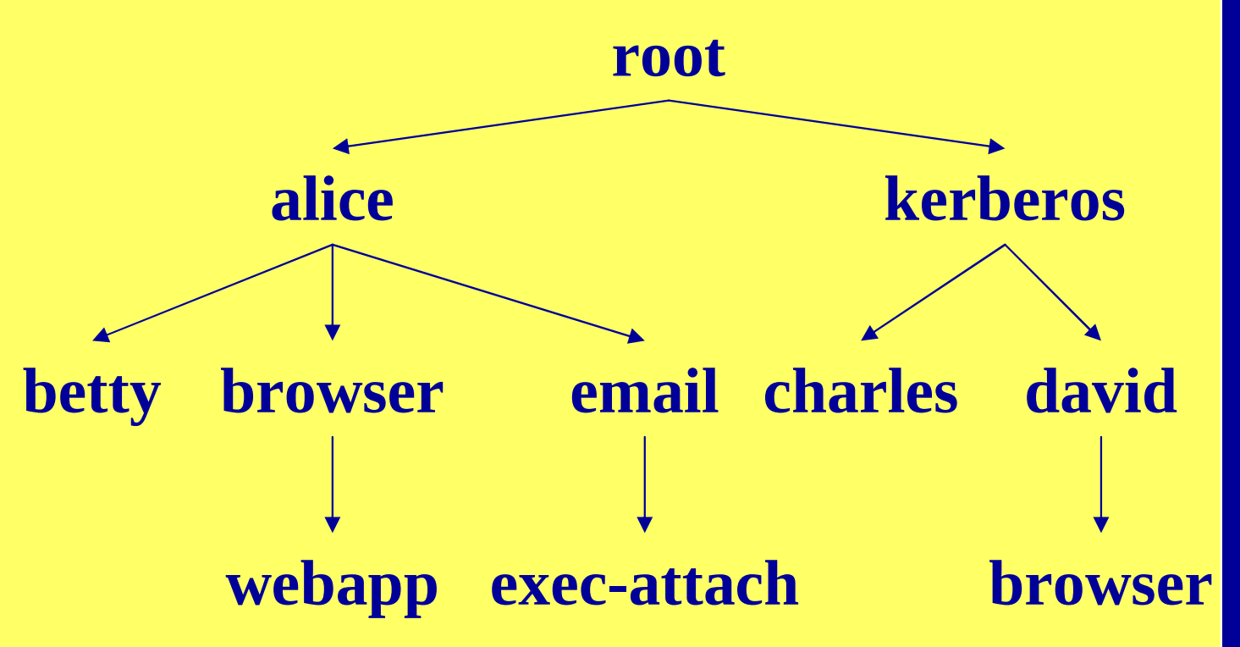
- **send signals to processes owned by their sub-users.**
- **implicitly access files belonging to their sub-users.**
- **debug processes belonging to their sub-users.**

Unix doesn't happen to allow these operations, but some support for sub-identities can be "dropped in" to an existing Linux system.

A Toolkit Approximation

We have implemented an approximation of the sub-identity model in the form of a toolkit of setuid binaries that can be installed on any Linux system.

```
/etc/subusers
root:alice,psnowber
alice:betty,browser,email
browser:webapp
email:exec-attach
```



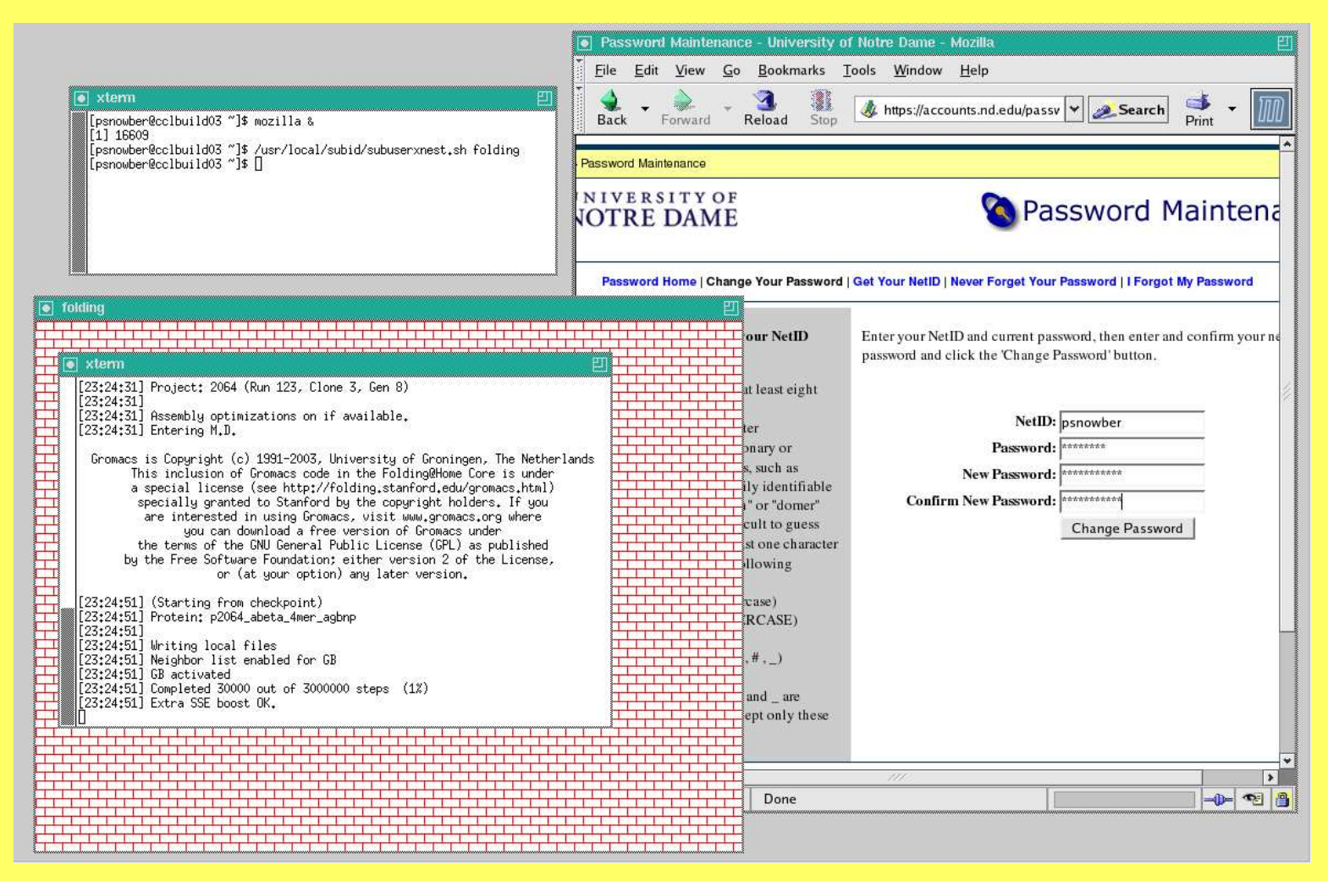
Here we show the identity relationships in a hypothetical system. Alice has a normal system account, whereas Charles and David authenticate through Kerberos. Each user has a normal entry in the /etc/passwd file.

Relative Complexity

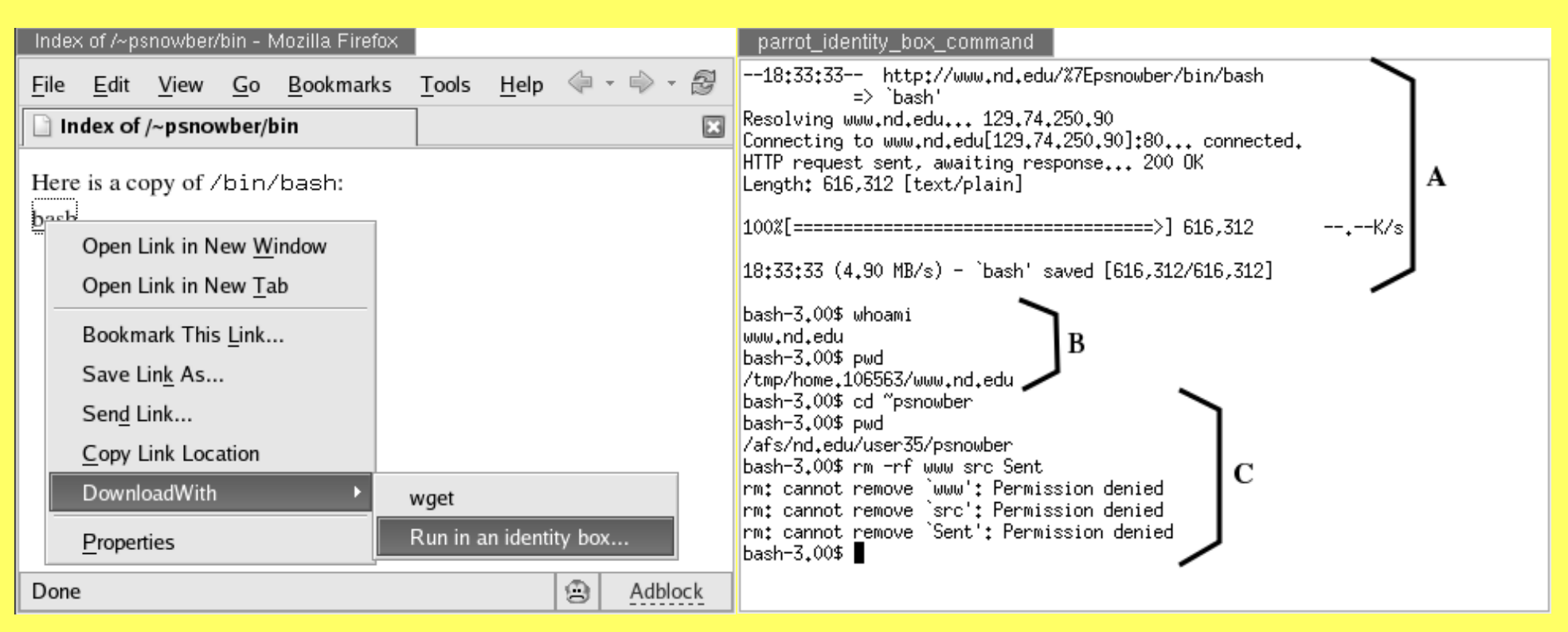
	Sub-Identity (toolkit)	Virtual Machine (VMWare)	Sandbox (Janus)
Lines of Code	885	*Millions?*	8743
Ops to set up	**Install toolkit** Call *subuseradd*	Create new image Install new OS into image **Set up network bridge** Install program	**Install kernel module** Identify system calls Define minimal policy
Ops to run	Call *subusersudo*	Run program in image	Run program in sandbox Iteratively update policy, if insufficient
Ops to retrieve	Call *subuserchown*	Extract through network	*none*

The relative complexity of various compartmentalization solutions is shown. Steps in bold text require root privileges.

Applications



A "Safe Window" is shown running an Xnest. Everything contained in the window is run as a sub-user. Potentially unstable or rogue X applications can be run inside it without compromising the security of the main X server.

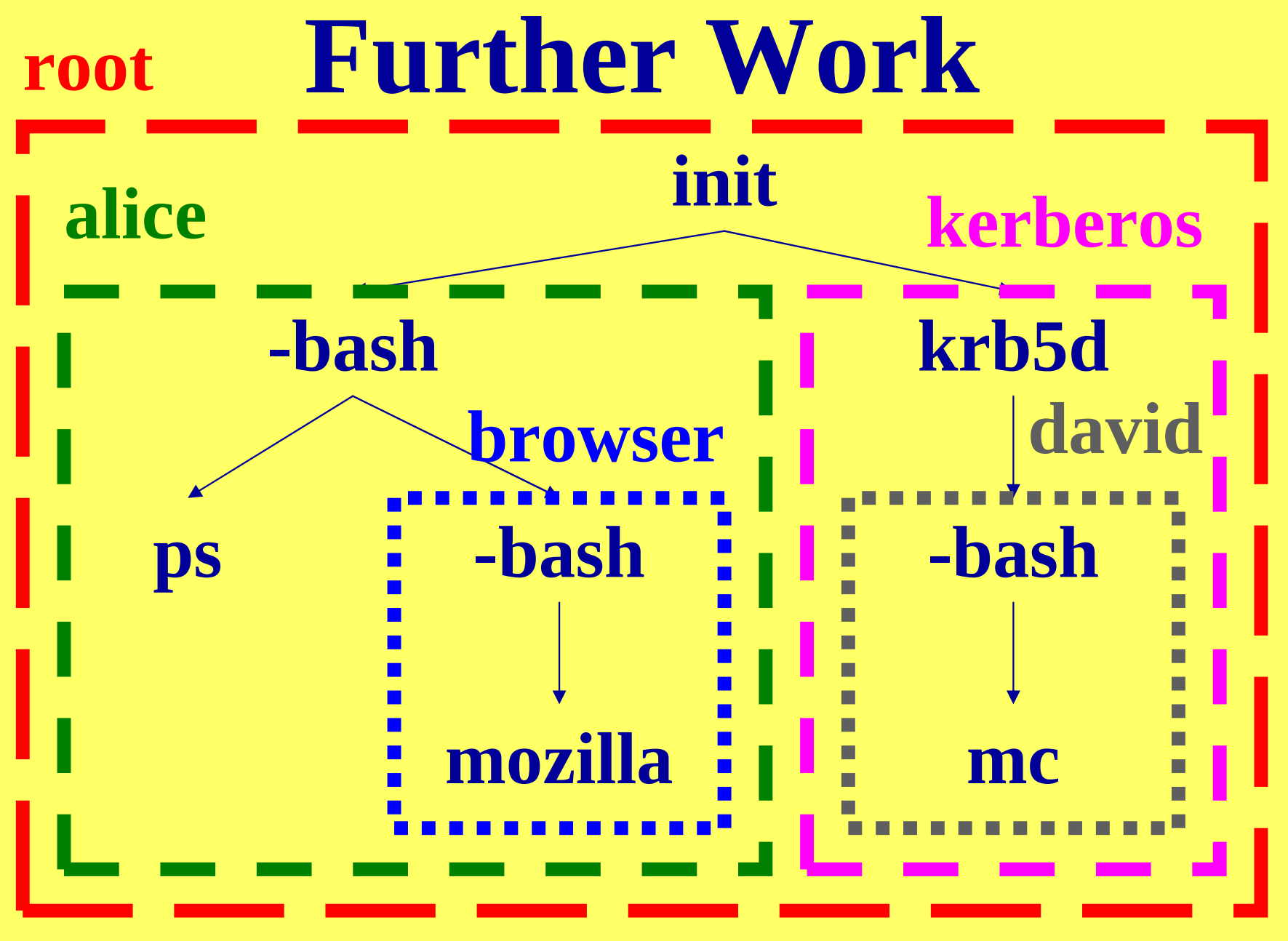


Here is a copy of /bin/bash:
bash

```
--18:33:33--  http://www.nd.edu/%7Epsnowber/bin/bash
           => `bash'
Resolving www.nd.edu... 129.74.250.90
Connecting to www.nd.edu[129.74.250.90]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 616,312 [text/plain]

100%[====================================>] 616,312        --.--K/s

18:33:33 (4.90 MB/s) - `bash' saved [616,312/616,312]

bash-3.00$ whoami
www.nd.edu
bash-3.00$ pwd
/tmp/home.106563/www.nd.edu
bash-3.00$ cd ~psnowber
bash-3.00$ pwd
/afs/nd.edu/user35/psnowber
bash-3.00$ rm -rf www src Sent
rm: cannot remove `www': Permission denied
rm: cannot remove `src': Permission denied
rm: cannot remove `Sent': Permission denied
bash-3.00$
```

A quick modification to a Firefox extension allows downloadable applications to be run seamlessly inside a security domain that is created on the fly. The callouts show the application being downloaded, the application under typical usage, and an unsuccessful attack being carried out.

Further Work



- **Process hierarchy: alice's `ps` command would only show the processes inside her box.**
- **Disk / chroot hierarchy?**

Presented at the USENIX Security Symposium, 3 August 2006.

This work was supported by NSF Grant CNS05-49087 "SGER: Enabling Electronic Self Defense with Dynamic Identities" and by an Arthur J. Schmitt Graduate Fellowship.

Sub-Identities: Towards Operating System Support for Distributed System Security, Philip Snowberger and Douglas Thain, Technical Report 2005-18, University of Notre Dame, Computer Science and Engineering Dept, October 2005.

Web address: http://www.cse.nd.edu/~ccl