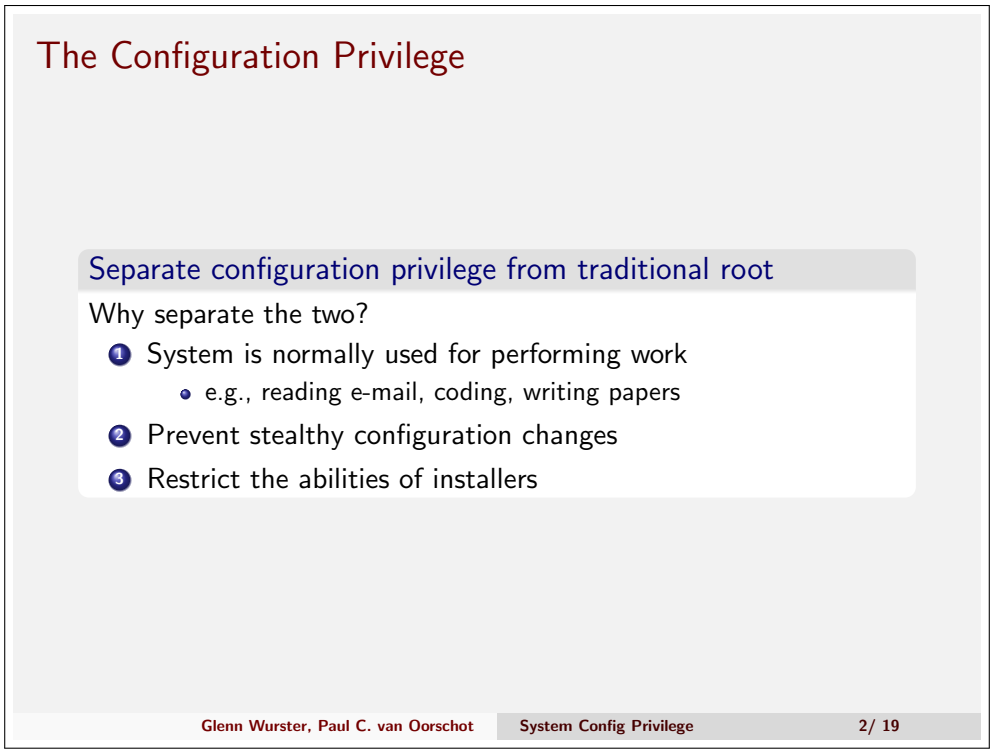
# System Configuration as a Privilege

**Glenn Wurster**, Paul C. van Oorschot
School of Computer Science
Carleton University, Canada

HotSec 2009 — 11 Aug 2009

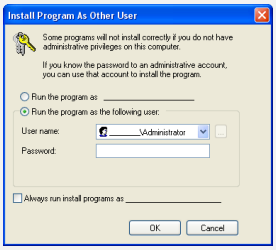



1. Compromised daemons/root applications
   - No limit to configuration changes
2. Dubious installers
   - Sony DRM, Kazaa

## Installing Applications

### Typical procedure

1. Become the superuser
2. Run the installer

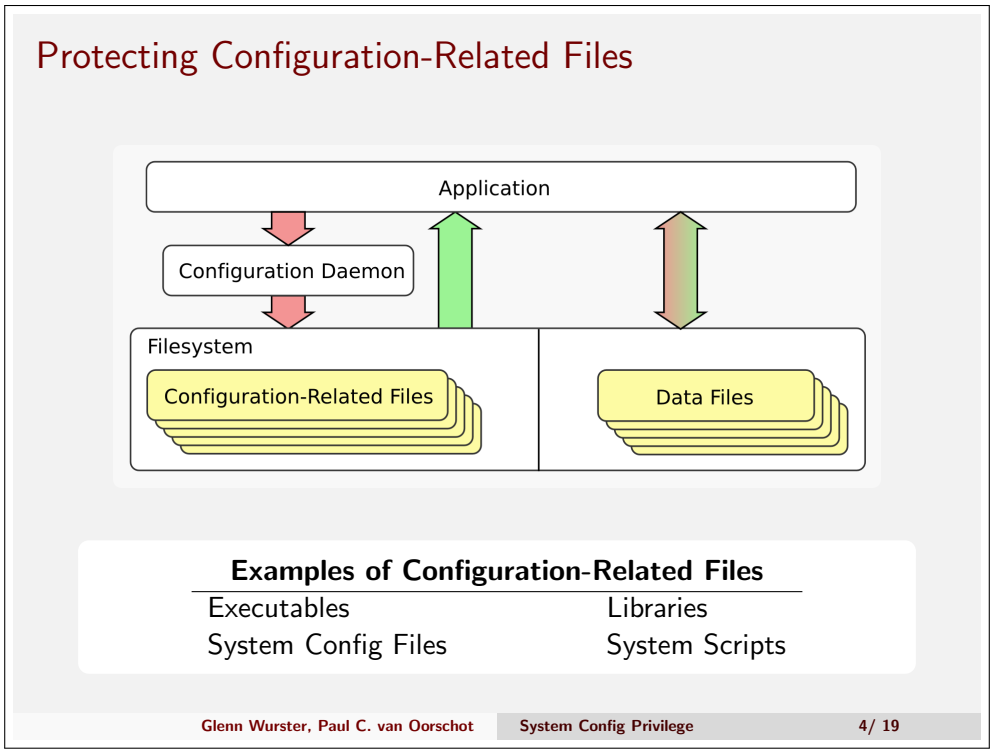



### Problem:

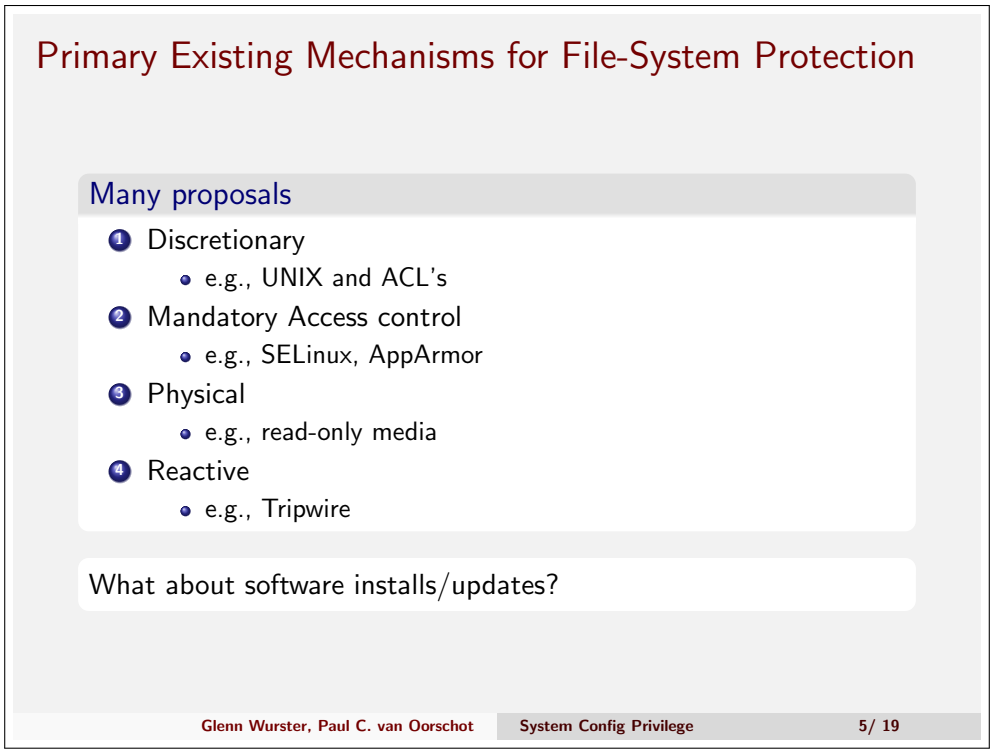
- Granting unrestricted access to the file system



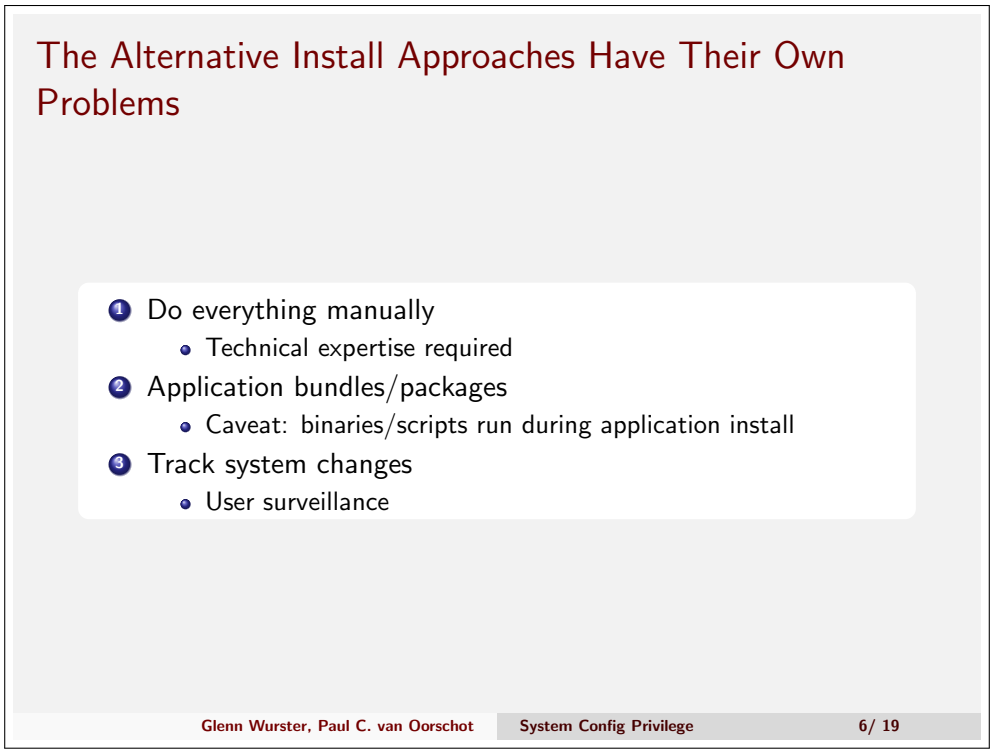
1. User knows when they want to install something
2. Explicitly grants superuser privileges
   - Can run arbitrary code
   - Can overwrite arbitrary files
3. Source of installer is downloaded code?
   - The complete opposite of what we're trying to tell them



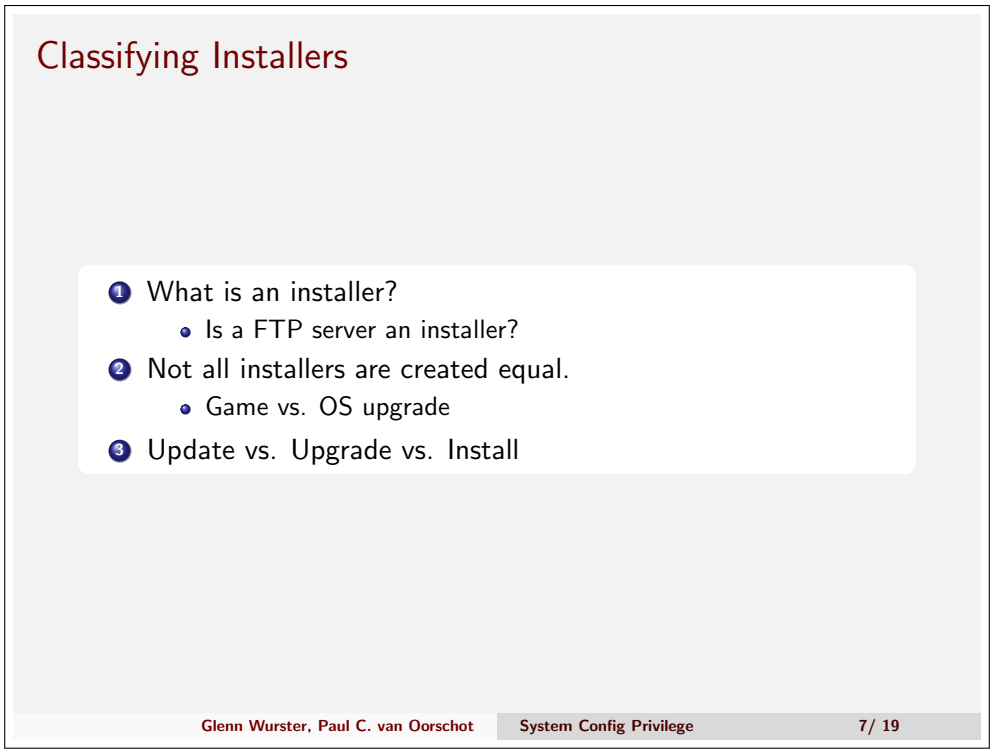

1. Information Flow
   - Read and write to data files
   - Only read from configuration-related files
2. Application Includes Installers
3. Regardless of whether the program is run as root
4. Example of Data file:
   - Your pictures of Montreal



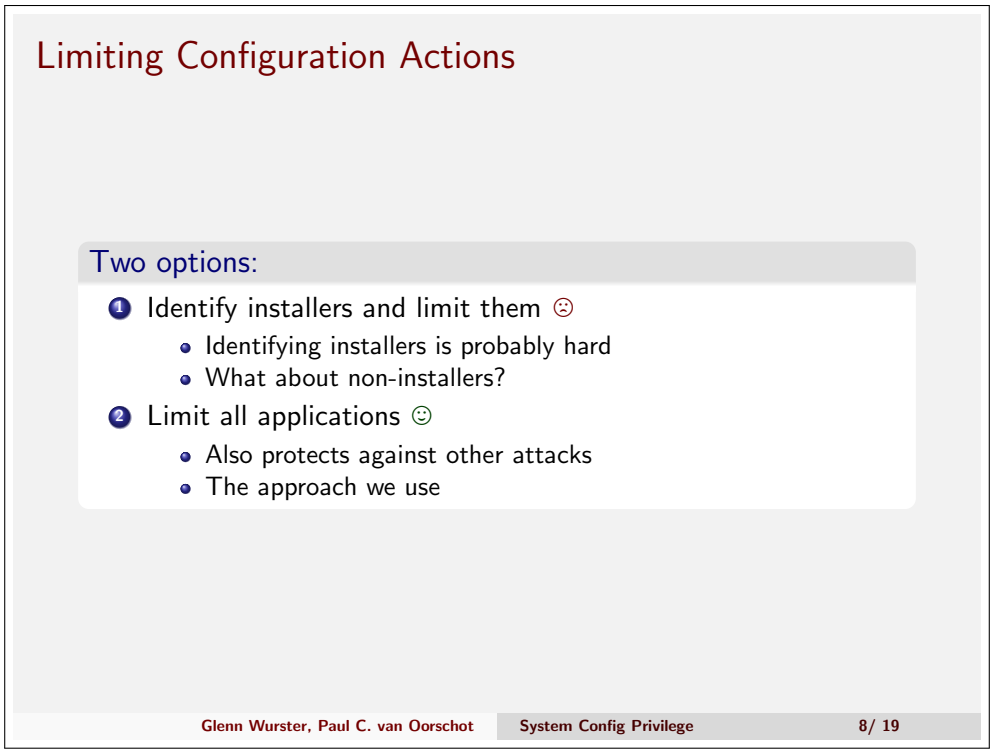

1. Assumption is the system is in a steady state
   - No new applications being installed/removed
2. Need very-high privileges to install/upgrade
   - Read-only media needs to be made writable
   - Tripwire needs manual merging of changes
3. My Observations
   - All handle installs, but were not designed for it.
   - Even MAC systems don't directly tackle the problem of how you get new/updated software onto the system.



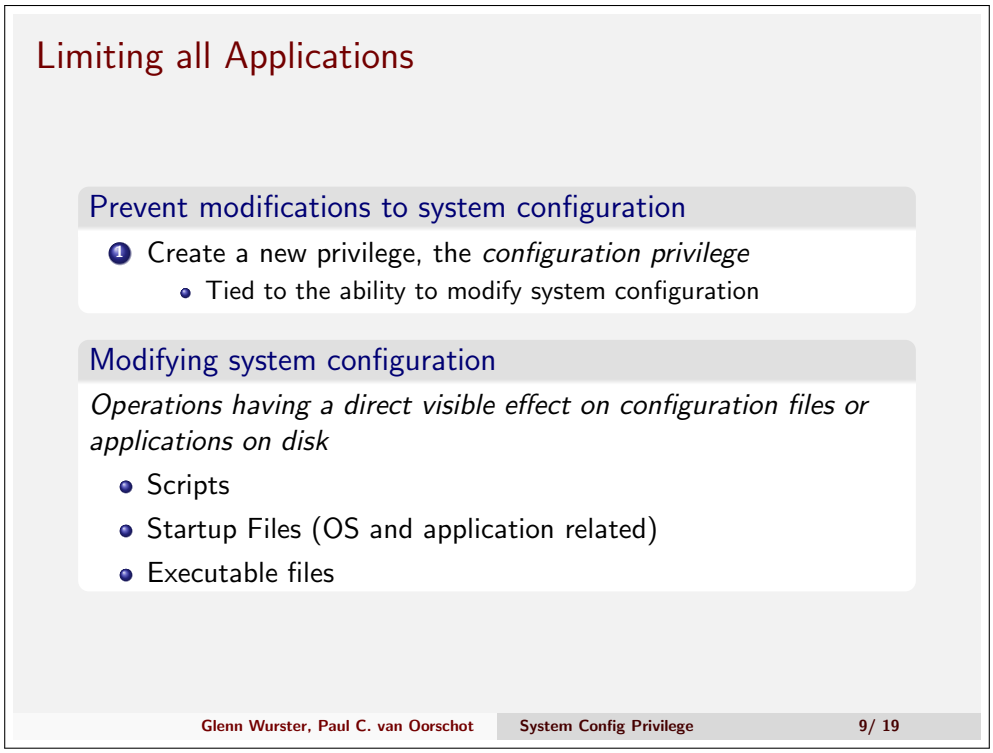

1. Manual Install:
   - Thick book of instructions
   - No installer run with superuser privileges
2. Application bundles:
   - Linux package managers - still run scripts with root
   - Apple bundles - don't know about scripts
3. Tracking Changes:
   - System Checkpointing, Sandboxing
   - User must run tools before install



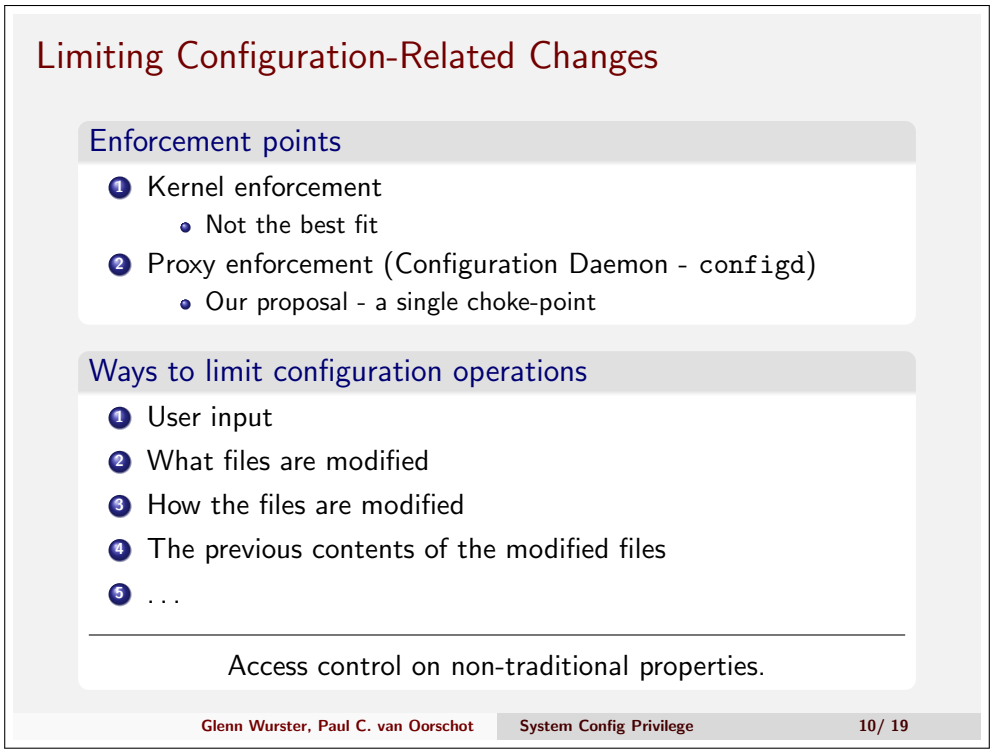

1. What is an installer
   - Hard question
   - Prone to subversion by malware
2. Different installers need to do different things
   - Security Updates change little
   - OS Upgrades change much
3. Any attempt to subclass the installer space?
   - Apparently not
4. Not sure we need to classify installers
   - Configuration privilege still a contribution without classification



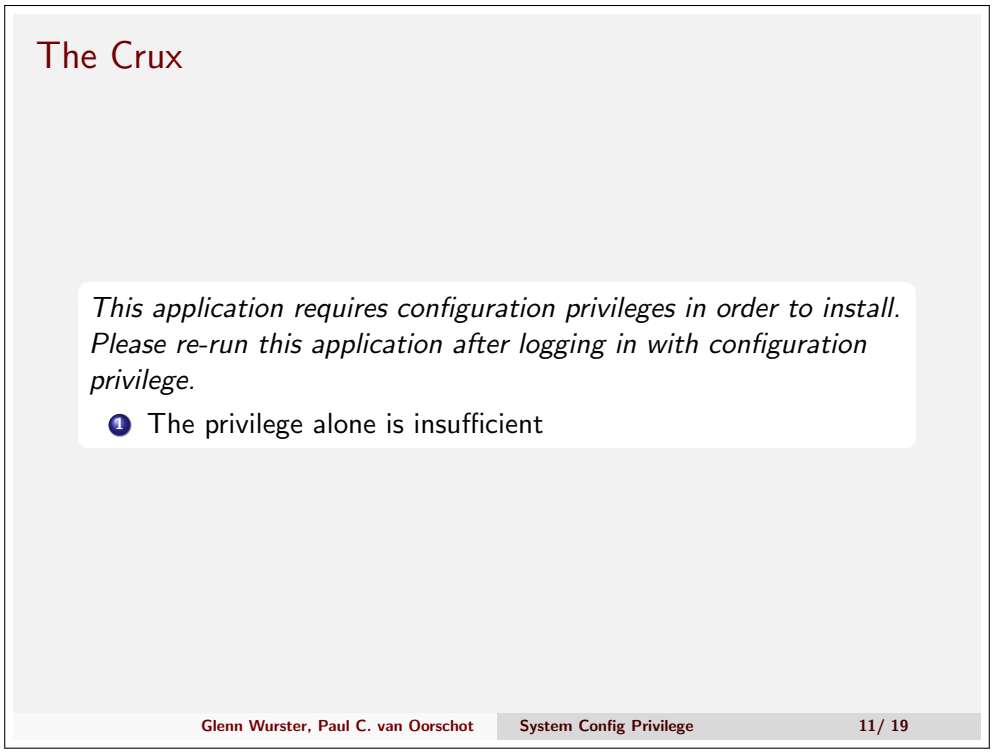

1. From Previous: Identifying what is an installer is hard
2. Alternative install approaches (manual, packages, tracking) required identifying installers
3. Non-installers should not be allowed to configure the system
4. Want to limit dangerous configuration operations regardless



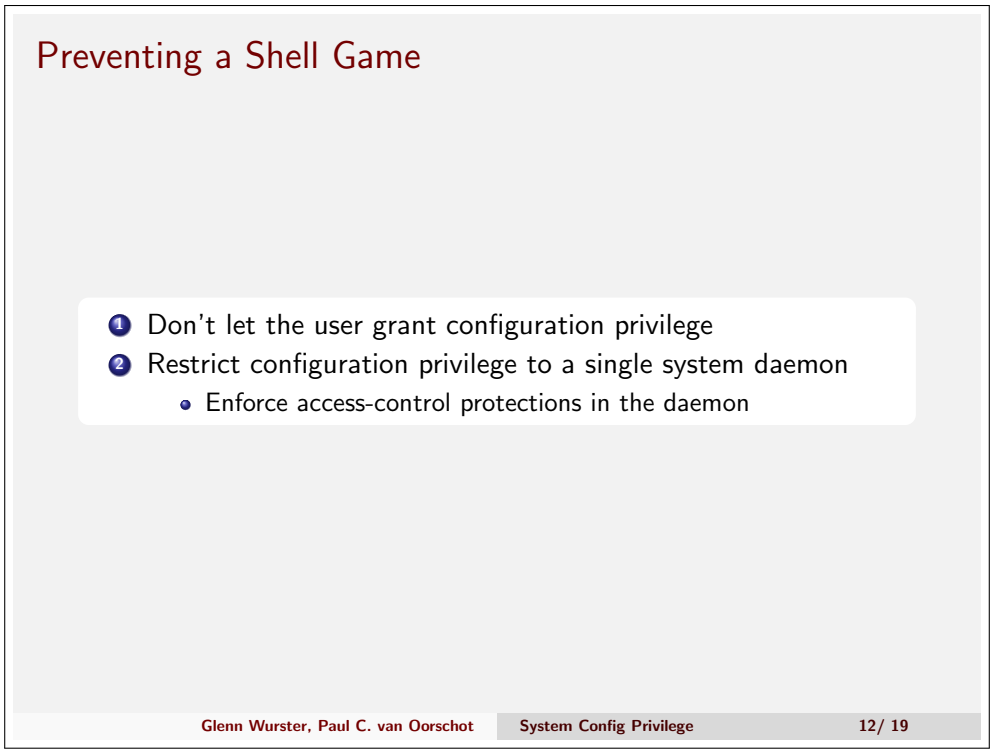

1. Privilege required to modify system configuration
   - Files on disk
2. Identify configuration changes and limit them
3. Privilege required to modify configuration files on a system




1. Kernel enforcement limits choices
   - Rootkit-resistant disks
   - Code-signing
2. Configd
   - A daemon which responds to configuration requests
3. User-input is intentionally broad
   - Traditional Keyboard/Display
   - USB Keys
   - Location Sensor
   - Biometrics




1. Problem:
   - Developers pop up a dialogue box
   - Users follow the instructions
   - We've just shifted the problem.
2. Not sufficient to just create the privilege
   - Must specify how the privilege is used
3. Restrict in a way that developers can't convince the naive user to run with elevated privileges.




1. Reduce the effectiveness of social engineering
2. A single system daemon
   - Make it hard to get around

## How Much Complexity Should Be Exposed?

**Desired target audience:**

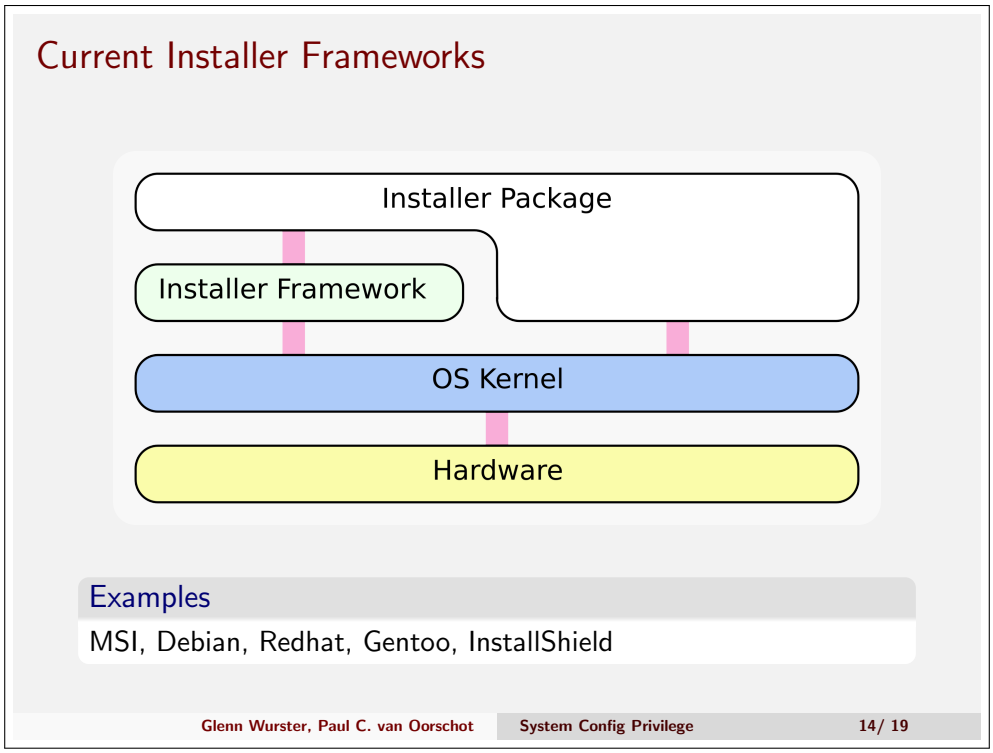
Non-technical users

vs.

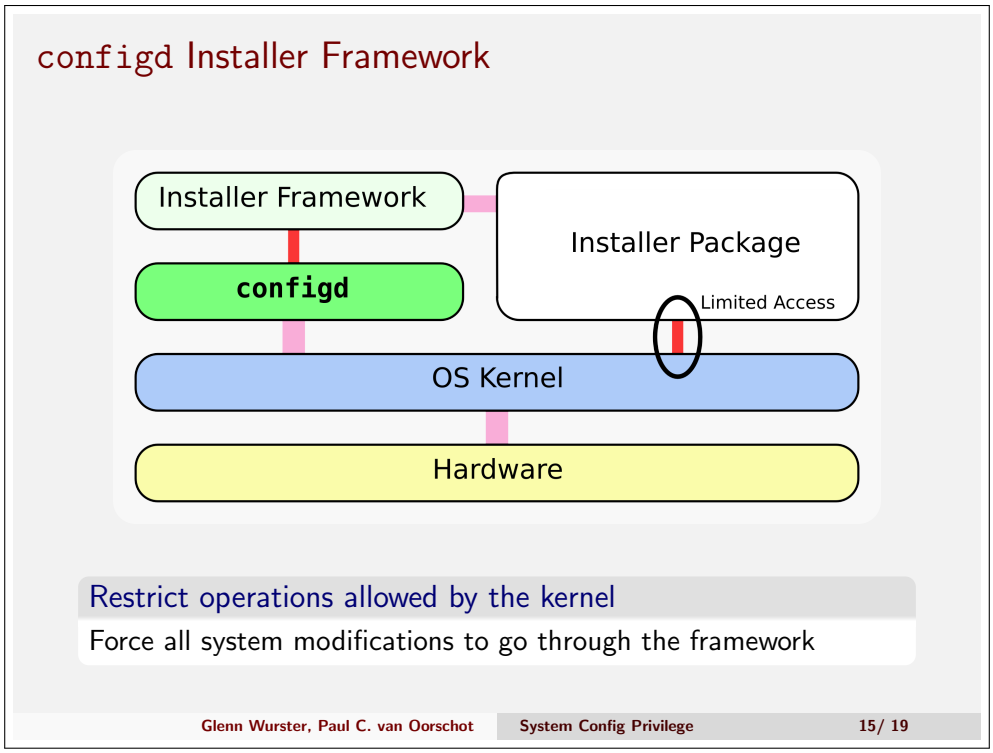
*Fine-grained access control solutions which require users to get involved with the fine grains usually fail due to usability issues.*



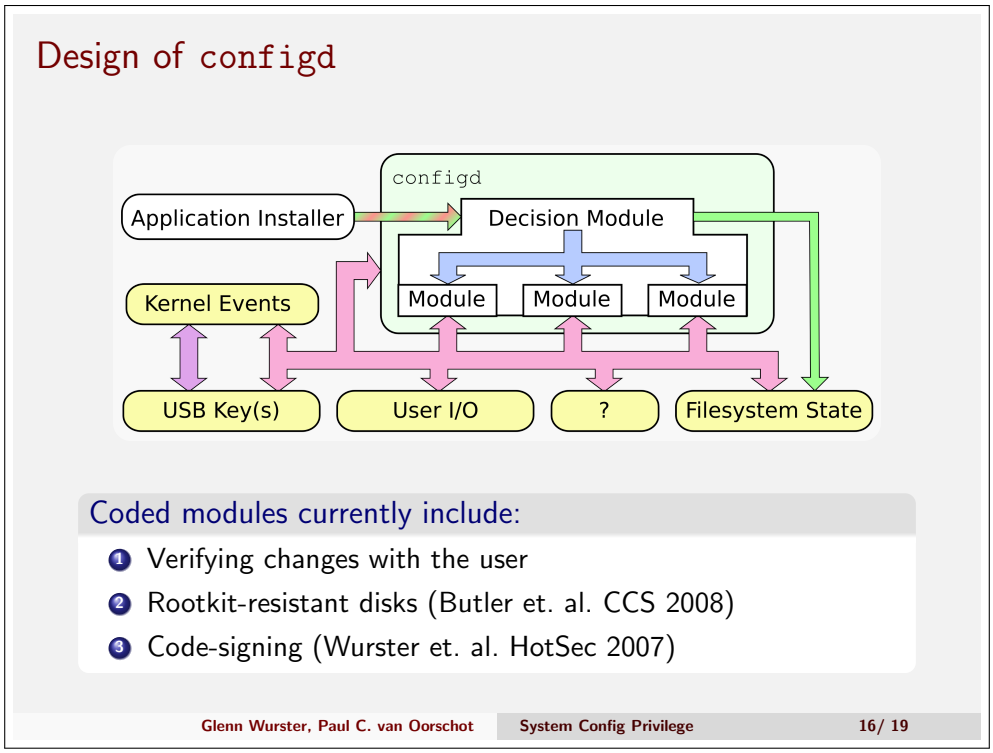
1. The collection of parts in a drill
   - More useful than the drill, but harder to use
2. Going past determining all the pieces we need
3. Involves determining how the pieces work together
4. Decrease the complexity exposed to the user



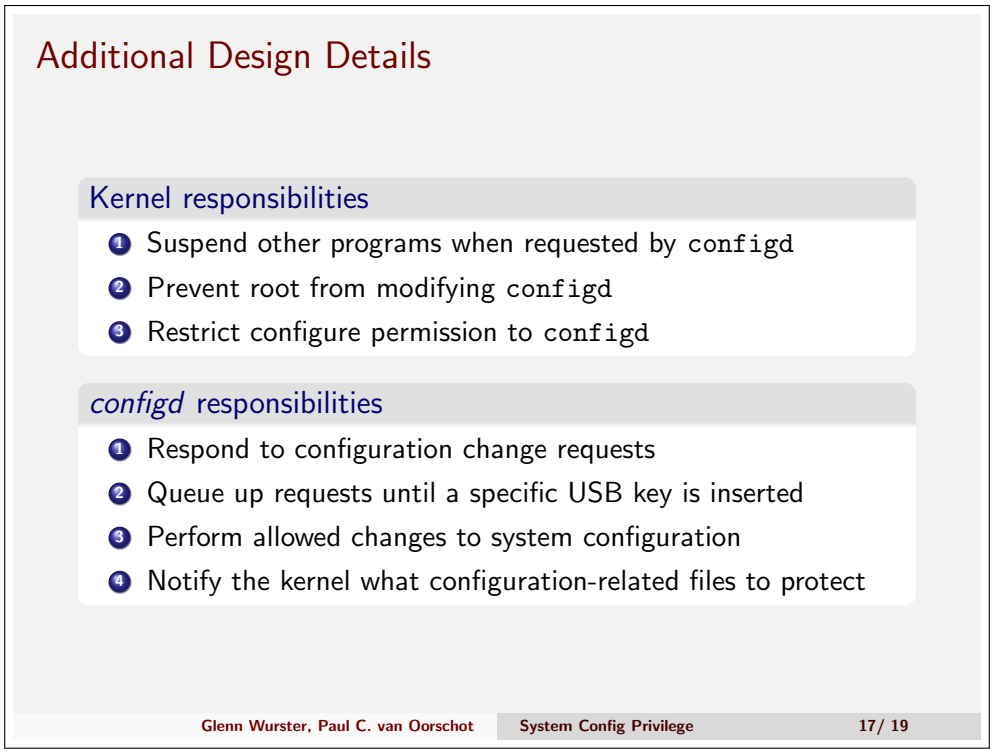

1. Installers can bypass most frameworks by talking directly to the OS kernel
   - Including all frameworks I'm aware of
2. OS does not restrict activities
   - Installer is run as superuser



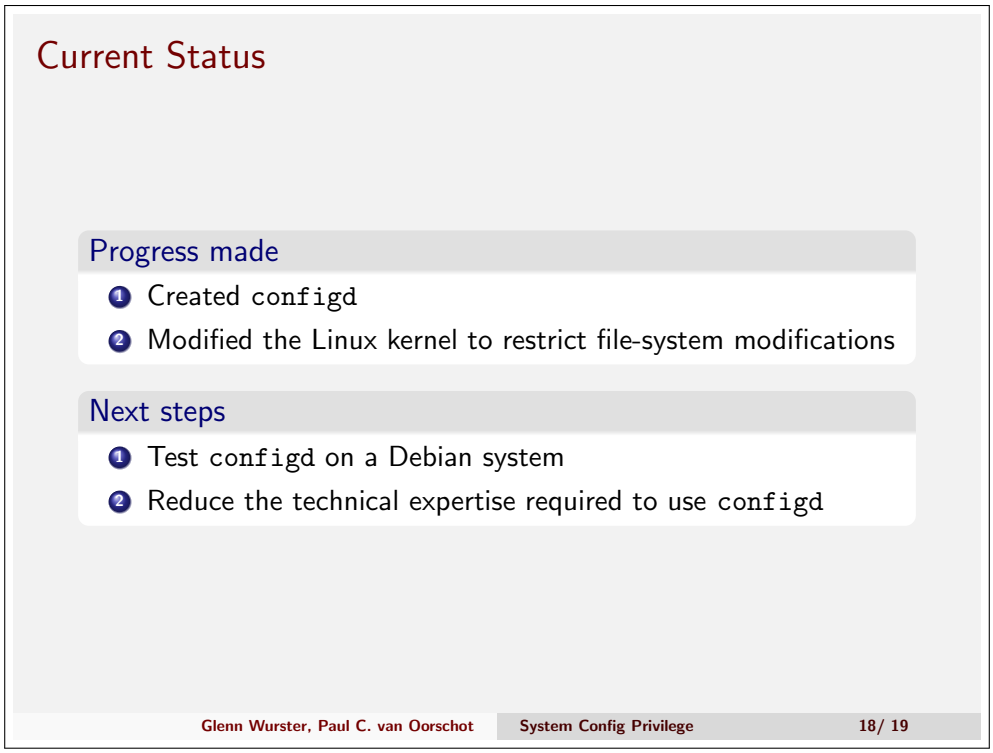

1. Applies to all applications
2. Interface with OS kernel is limited
3. Configd can reject requests for configuration changes




1. Each Module:
   - Examines elements of system state
   - Rejects, Allows, or Postpones making a decision
2. Base Configd also responds to system events
3. Rejected requests do not not modify file-system state
4. Modules work together
   - e.g., Code-signing combined with Rootkit-resistant disks
5. Mix of modules is not right yet.
6. Experimenting with asking the user
7. Extensible to try out new ideas
   - Module list is not designed to be modifed at run-time




1. Kernel:
   - Prevents others from getting configure permission
   - Protects the configd process
   - Protects configuration-related files
2. Configd:
   - Deals with requests for configuration changes
   - Notifies the kernel what files are configuration related
3. Prevent race conditions by suspending other tasks




1. Done:
   - Configd framework and modified Linux kernel
2. In progress:
   - Integrating **`configd`** into a Debian system
   - Start to determine what the right mix of modules should be

## Questions