Linux Kernel Security Overview

Kernel Conference Australia
Brisbane, 2009

James Morris
jmorris@namei.org

Introduction

Historical Background

- Linux started out with traditional Unix security
  - Discretionary Access Control (DAC)
- Security has been enhanced, but is constrained by original Unix design, POSIX etc.
- Approach is continual retrofit of newer security schemes, rather than fundamental redesign

“The first fact to face is that UNIX was not developed with security, in any realistic sense, in mind; this fact alone guarantees a vast number of holes.”

Dennis Ritchie, “On the Security of UNIX”, 1979

DAC

- Simple and quite effective, but inadequate for modern environment:
  - *Does not protect against flawed or malicious code*
- Linux implementation stems from traditional Unix:
  - User and group IDs
  - User/group/other + read/write/execute
  - User controls own policy
  - Superuser can violate policy

“It must be recognized that the mere notion of a super-user is a theoretical, and usually practical, blemish on any protection scheme.”

Ibid.

Extended DAC

- POSIX Capabilities (privileges)
  - Process-based since Linux kernel v2.2
    - Limited usefulness
  - File-based support relatively recent (v2.6.24)
    - May help eliminate setuid root binaries

- Access Control Lists (ACLs)
  - Based on abandoned POSIX spec
  - Uses extended attributes API

Linux Namespaces

- File system namespaces introduced in 2000, derived from Plan 9.
  - Not used much until *mount propagation* provided more flexibility (e.g. shared RO “/”)
  - Mounts private by default
- Syscalls unshare(2) and clone(2) allow control over sharing of resources
- Provides good isolation between processes
- PAM integration
- Used w/ SELinux in kiosk mode

Network Access Control

- Netfilter
  - Packet filtering and mangling framework
  - API allows kernel applications to register by protocol and packet flow point
- IPTables
  - Extensible packet filter for IPv4/IPv6
  - Connection tracking (stateful inspection)
  - NAT
  - Hundreds of contributed matches and targets

Missing Link

- Seminal 1998 NSA paper: *The Inevitability of Failure* describes additional security requirements:
  - Mandatory security
  - Trusted / protected path
  - Assurance
- Difficult work, but we are getting there...

Cryptography

- Historical US export limitations prevented merge of comprehensive cryptography
  - External “kerneli” tree had a crypto API
  - Other projects added own crypto, e.g. FreeSWAN
- Some allowed uses:
  - Hashing
  - RNG

Cryptography

- Crypto API developed rapidly for native IPSec implementation, made it into 2.6 kernel
- Scatterlist API
- Initially synchronous w/ support for basic cipher modes, digests and compressors
- Dynamic crypto algorithm module loading
- Now significantly evolved w/ async, hardware support, ASM, many algorithms & modes

Disk Encryption: DM-Crypt

- Operates transparently at block layer
- Key management with LUKS
- Default is AES-128/SHA-256
- Very nice integration in Fedora; try it!

Disk Encryption: ecryptfs

- Stacked filesystem encryption at VFS layer
- Per-object encryption
- Extensible key management
- Cryptographic metadata stored w/ objects, allows them to be moved to different hosts

Network Encryption: IPSec

- Supports IPv4 and IPv6
- Implemented via generic transform (xfrm) framework:
  - xfrm stack applied to packet based on policy db
- xfrms include: ESP, AH, IPComp, MIP, IPIP
- Utilizes native Netlink sockets for scalability
- Also supports PF_KEY

Memory Protection

- Address Space Layout Randomization (ASLR)
- NX (No eXecute) bit support where available in hardware or via emulation
- GCC stack smashing protector
- /dev/mem & null pointer restrictions
- MAC policy can be applied via SELinux:
  - execheap, execmem, execmod, execstack

Kernel Vulnerabilities

- Note that kernel vulnerabilities may allow attackers to bypass kernel-based security mechanisms.

- See *“Linux Kernel Heap Tampering Detection”*, in *Phrack 66* for a detailed discussion of the topic.

Linux Security Modules (LSM)

- Framework for integrating access control schemes
- Hooks located at security-critical points in the kernel, pass security-relevant information to LSM module, which can veto the operation
- Avoids races when making security decisions
- Restrictive interface: can only further confine access, not loosen it

SELinux

- Flexible fine-grained MAC scheme w/ least privilege, confidentiality, integrity, isolation, information flow control; exploit containment
- Composition of multiple security models under single analyzable policy
- Currently ships with: Type Enforcement, RBAC and MLS/MCS
- Clean separation of mechanism and policy to meet very wide range of usage scenarios

Simplified Mandatory Access Control Kernel (SMACK)

- Simple labeling of subjects and objects to provide flexible MAC
- System labels define hierarchical limits
- Admin-defined labels can be any short string
- Policy is written as triples:

  ```
  Subject Object [—rwxa]
  ```

AppArmor

- Not currently in kernel
- Path name access control scheme to confine applications
- Aims to solve security usability by expressing policy with a familiar abstractions, e.g.:
  - File access controls defined with path names and names of common operations
  - POSIX capabilities described by name

TOMOYO

- Path-based MAC scheme developed by NTT research
- Aims to solve security usability with automatic real-time policy generation
- Enforces previously observed behavior in learning mode
- Domains are trees of process invocation
- Rules apply to domains

Labeled Networking

- NetLabel
  - CIPSO
    - Legacy labeling using IP options
  - IPSec
    - Labeling of Security Associations
- Secmark
  - Utilizes iptables
  - Generic labeling (SMACK & SELinux use it)

Network File Systems

- Labeled NFS
  - NFSv4 extension
  - Prototype code
  - Also need to extend RPC security
  - IETF process ongoing
- NFS ACLs
  - Support for Linux ACLs and NFSv4 ACLS
    - See talk by Greg Banks at LCA

Anti-Malware

- Good userspace solutions
- People still want kernel scanning
- fsnotify
  - Generalized file notification framework
  - Consolidate dnotify & fsnotify
  - Useful for HSM
- TALPA
  - File access scanning API for AV modules

Integrity & Platform Security

- TPM (Trusted Platform Module)
  - Cryptographic processor, RNG, storage for keys and measurements
- IMA (Integrity Measurement Architecture)
  - Static integrity verification of code
- TXT (Intel Trusted Execution Technology)
  - DRTM (Dynamic Root of Trust Measurement); trusted launch, hardware security enhancements
- VT-d (device virtualization)
  - Needed to secure IO devices

Audit

- Developed for certification (e.g. CAPP)
- Audit framework generates events:
  - User sessions & configuration changes
  - Syscalls
  - LSM decisions
- Useful for forensics and deterrence
- SELinux, SMACK et al use it for detailed reporting
- Netlink API for audit daemon, IDS

Seccomp

- Secure computing mode
  - Extremely lightweight sandboxing for untrusted code
  - Application enters mode with fixed set of restricted syscalls (read, write, exit, sigreturn)
- Proposal to convert into generic syscall filter
  - Historically problematic area

High Level View

- State of the art: Fedora 11
  - Kiosk Mode as example
- Known mitigations
- Certifications
  - RHEL: LSPP, CAPP, RBACPP at EAL4+
  - Not a separate product, all upstream and open
- Security features standard and generalized

Future Directions

- Continued refinement and hardening
  - Working towards “Inevitability” goals
- Extensible models
  - Consistent policy for entire computing environment
- Cloud Computing

Challenges

- Multiple security models hindering adoption

- Convincing people of the value of security:
  - enable features
  - report problems
  - help improve usability

Resources

- Linux Kernel Security Wiki
- LSM Mailing List
- LWN Security page

Questions?

Useful URLs

Kernel Security Wiki
    http://security.wiki.kernel.org/

LSM Mailing List
    http://vger.kernel.org/vger-lists.html#linux-security-module

LWN Security Page
    http://lwn.net/Security/

“The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments”
    http://csrc.nist.gov/nissc/1998/proceedings/paperF1.pdf

LSM Usenix Paper
    http://www.usenix.org/event/sec02/wright.html

Kernel Memory Protection
    http://lwn.net/Articles/329787/

Linux Security Model Comparison
    http://tomoyo.sourceforge.jp/wiki-e/?WhatIs#comparison

Useful URLs ...

SELinux
    http://selinuxproject.org/
“Have You Driven an SELinux Lately?” (OLS paper on current state)
    http://namei.org/ols-2008-selinux-paper.pdf
“Anatomy of Fedora Kiosk Mode”
    http://namei.org/presentations/fedora-kiosk-mode-foss-my-2008.pdf
“SELinux Memory Protection Tests”
    http://people.redhat.com/drepper/selinux-mem.html
“A seatbelt for server software: SELinux blocks real-world exploits”
    http://www.linuxworld.com/news/2008/022408-selinux.html

SMACK
    http://schaufler-ca.com/

AppArmor
    http://en.opensuse.org/Apparmor

TOMOYO
    http://tomoyo.sourceforge.jp/

“POSIX file capabilities: Parceling the power of root”
    http://www.ibm.com/developerworks/library/l-posixcap.html

“POSIX Access Control Lists on Linux”
    http://www.suse.de/~agruen/acl/linux-acls/online/

Useful URLs ...

"Implementing Native NFSv4 ACLs in Linux"
http://lca2009.linux.org.au/slides/79.tar.gz

“Applying mount namespaces”
http://www.ibm.com/developerworks/linux/library/l-mount-namespaces.html

“Disk encryption in Fedora: Past, present and future”
http://is.gd/16012

“Limiting buffer overflows with ExecShield” (2005)
http://www.redhat.com/magazine/009jul05/features/execshield/

“Linux Kernel Heap Tampering Detection”
http://phrack.org/issues.html?issue=66&id=15#article

“System integrity in Linux”
http://lwn.net/Articles/309441/
“Linux kernel integrity measurement using contextual inspection” (LKIM)
http://portal.acm.org/citation.cfm?id=1314354.1314362

Intel TXT Site
http://www.intel.com/technology/security/

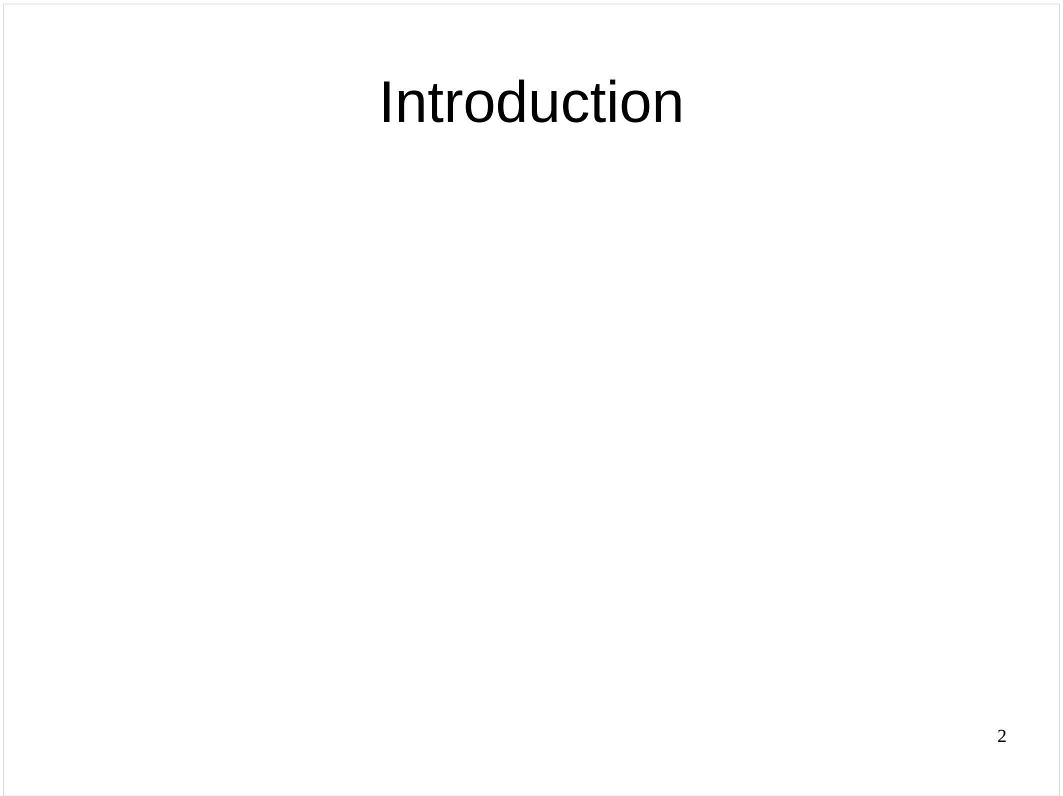
IBM TCPA Resources
http://www.research.ibm.com/gsal/tcpa/tcpa_rebuttal.pdf

Invisible Things Labs
http://theinvisiblethings.blogspot.com/

Linux Kernel Security Overview

Kernel Conference Australia
Brisbane, 2009

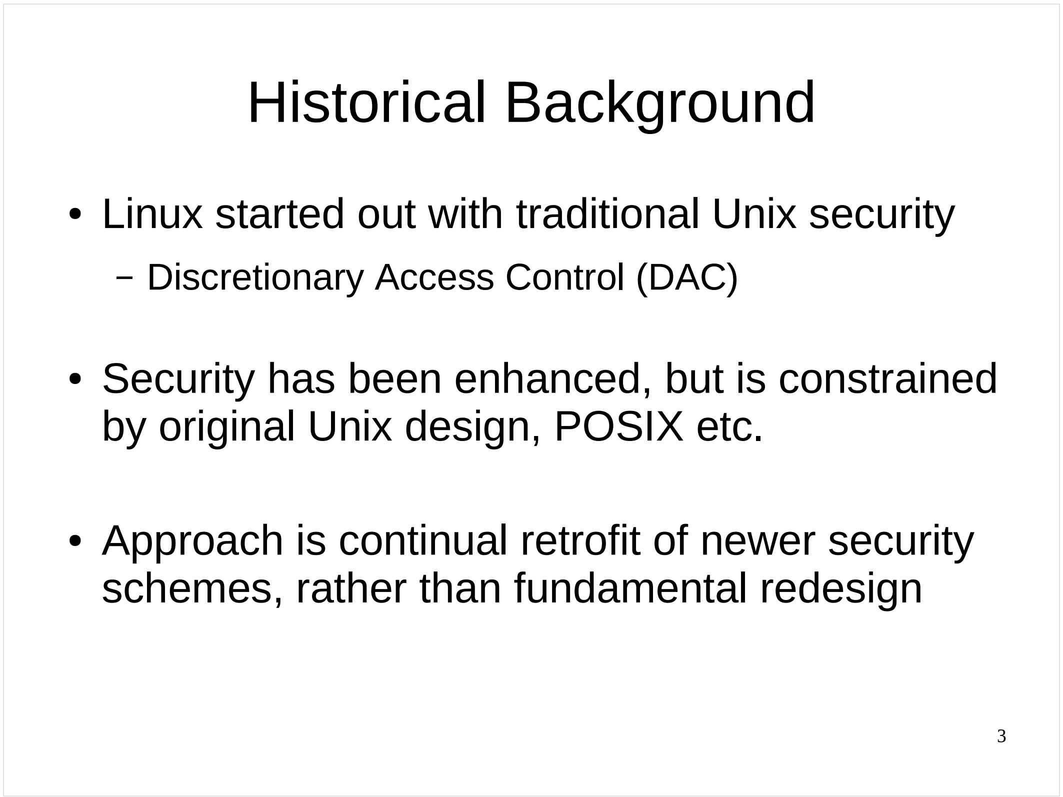
James Morris
jmorris@namei.org



Introduction

- Discuss scope and purpose of talk:
  - Provide a high-level overview of Linux kernel security
  - Cover significant security subsystems
  - Historical background and rationale
  - Development model & (lack of) overall design
  - Security has been retrofitted
    - Pros and cons

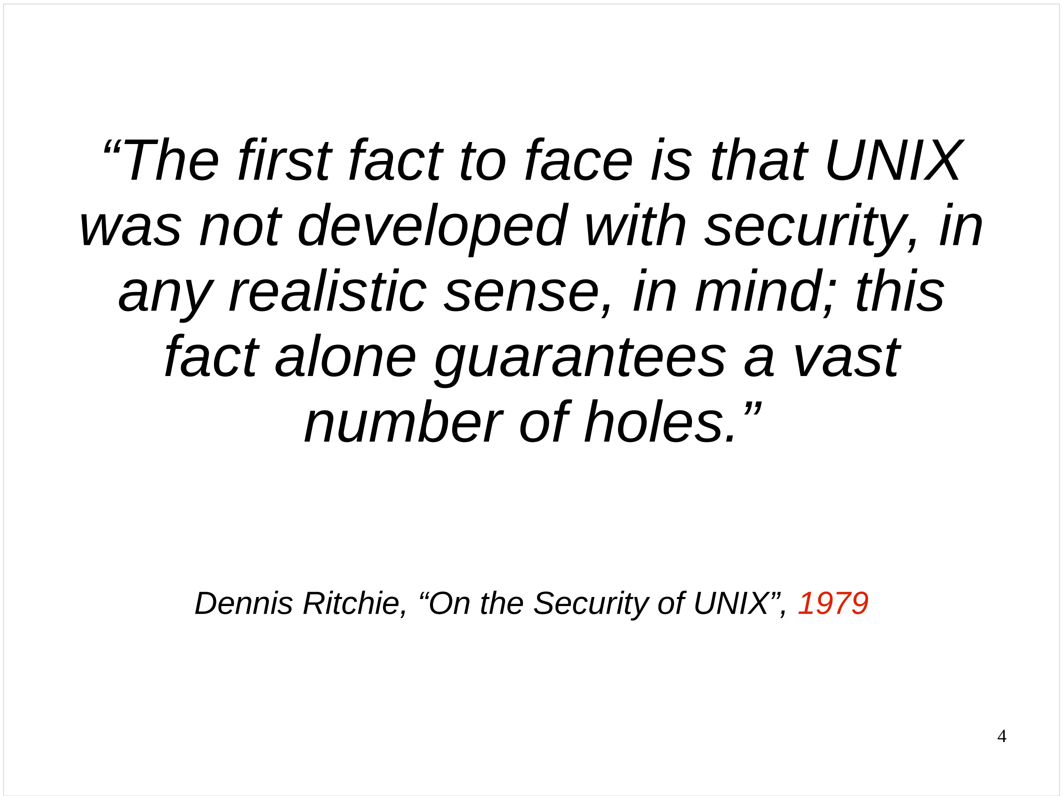
  - Understanding of why, not just how
  - Build understanding of current system and directions
  - Useful for developers, admins, researchers etc. as a starting point
- Also only talking about in-tree unless otherwise noted
- The scope is kernel security: this does not cover general Linux security, which needs a large book!



Historical Background

- Linux started out with traditional Unix security
  - Discretionary Access Control (DAC)
- Security has been enhanced, but is constrained by original Unix design, POSIX etc.
- Approach is continual retrofit of newer security schemes, rather than fundamental redesign

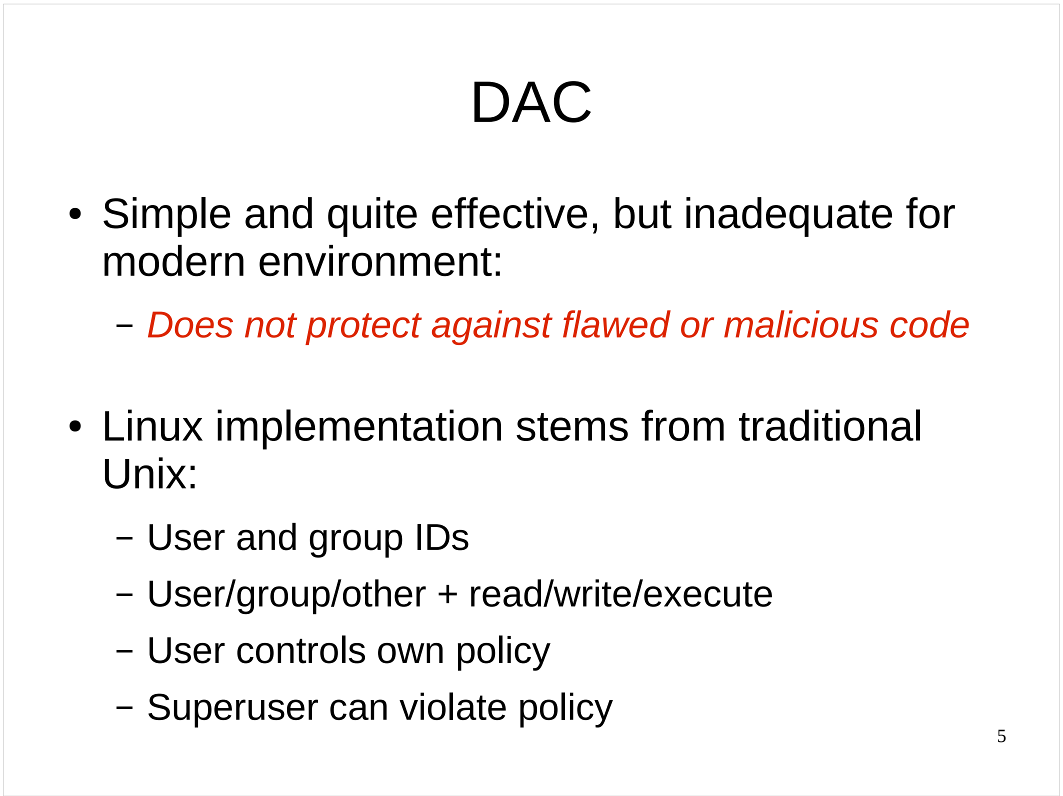
- Llinux security constrained by these factors, can’t re-design / break userland



“The first fact to face is that UNIX was not developed with security, in any realistic sense, in mind; this fact alone guarantees a vast number of holes.”

Dennis Ritchie, “On the Security of UNIX”, 1979

- There are references on this back to at least 1975...

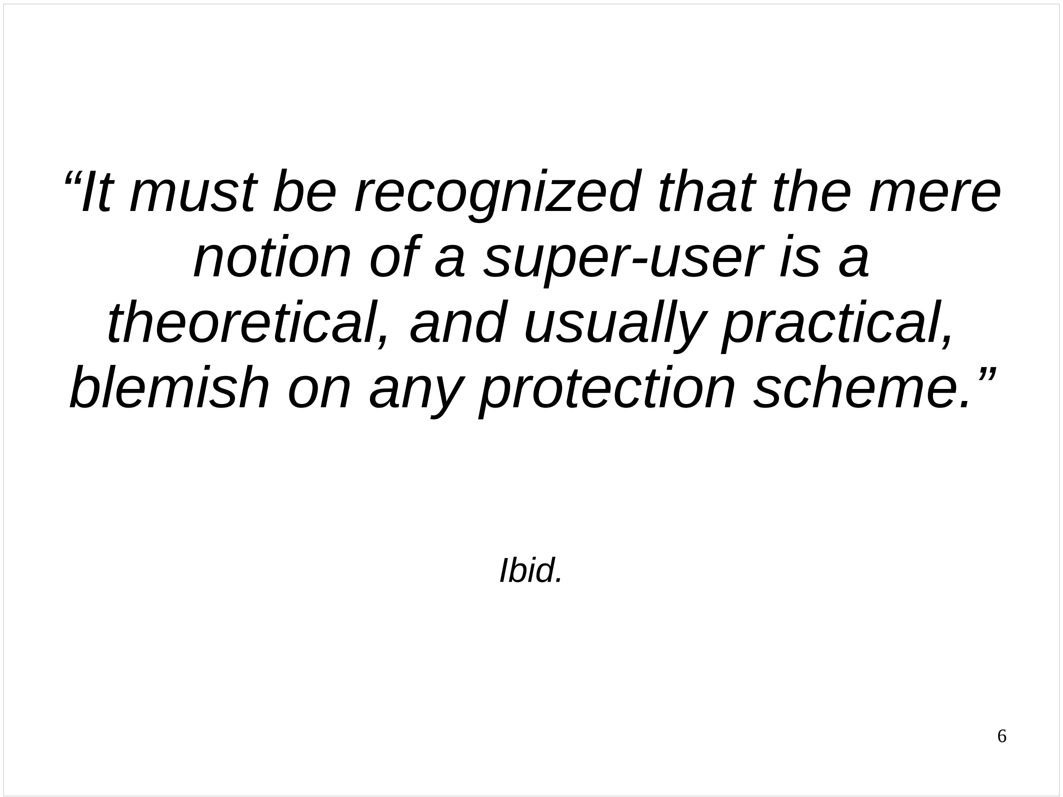


DAC

- Simple and quite effective, but inadequate for modern environment:
  - *Does not protect against flawed or malicious code*

- Linux implementation stems from traditional Unix:
  - User and group IDs
  - User/group/other + read/write/execute
  - User controls own policy
  - Superuser can violate policy

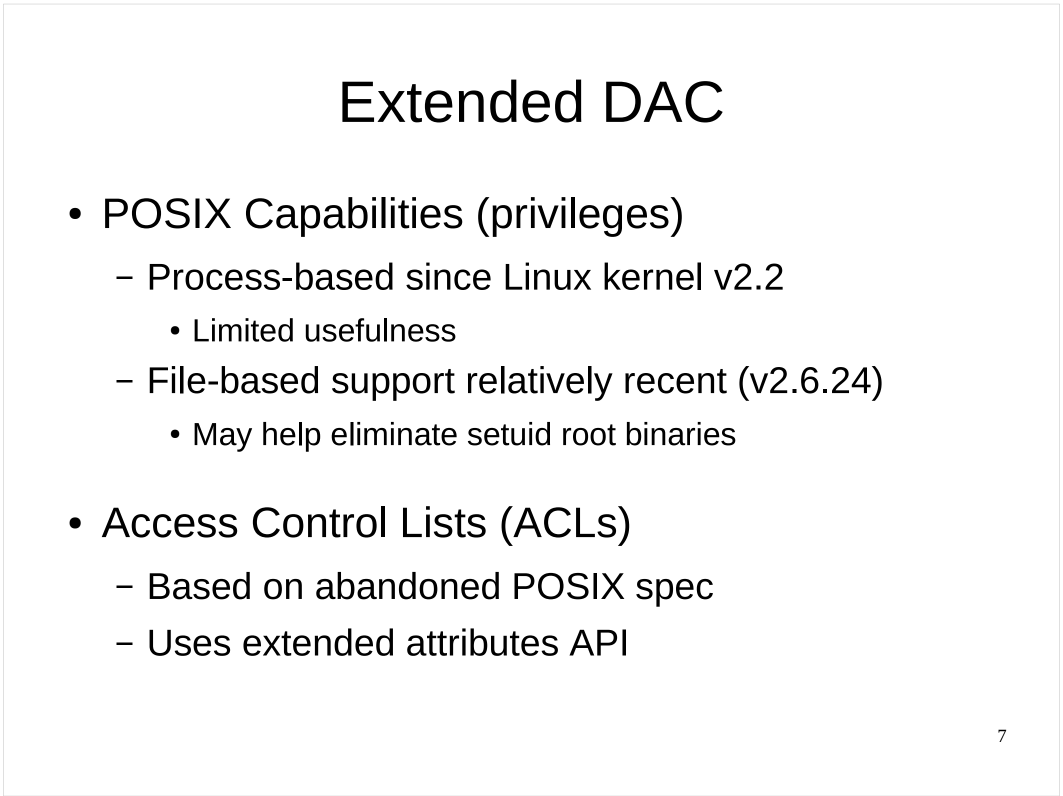
- Unix DAC has been very successful due to its simplicity (although still trips people up...).
- “root” is allowed to violate security policy
- Basically: all or self security w/ abbreviated ACLs
- Not good enough because:
  - ref: NSA paper “The Inevitability of a failure”
- Consider that in practice, all software has bugs; some of those bugs may be security issues, therefore it is prudent (and historically accurate) to assume all software has security bugs; DAC simply cannot provide effective protection as the security policy is controlled by the flawed software!
- People assume MAC means “trusted systems” and that they don’t need it; MAC can and has been generalized....



“It must be recognized that the mere notion of a super-user is a theoretical, and usually practical, blemish on any protection scheme.”

Ibid.

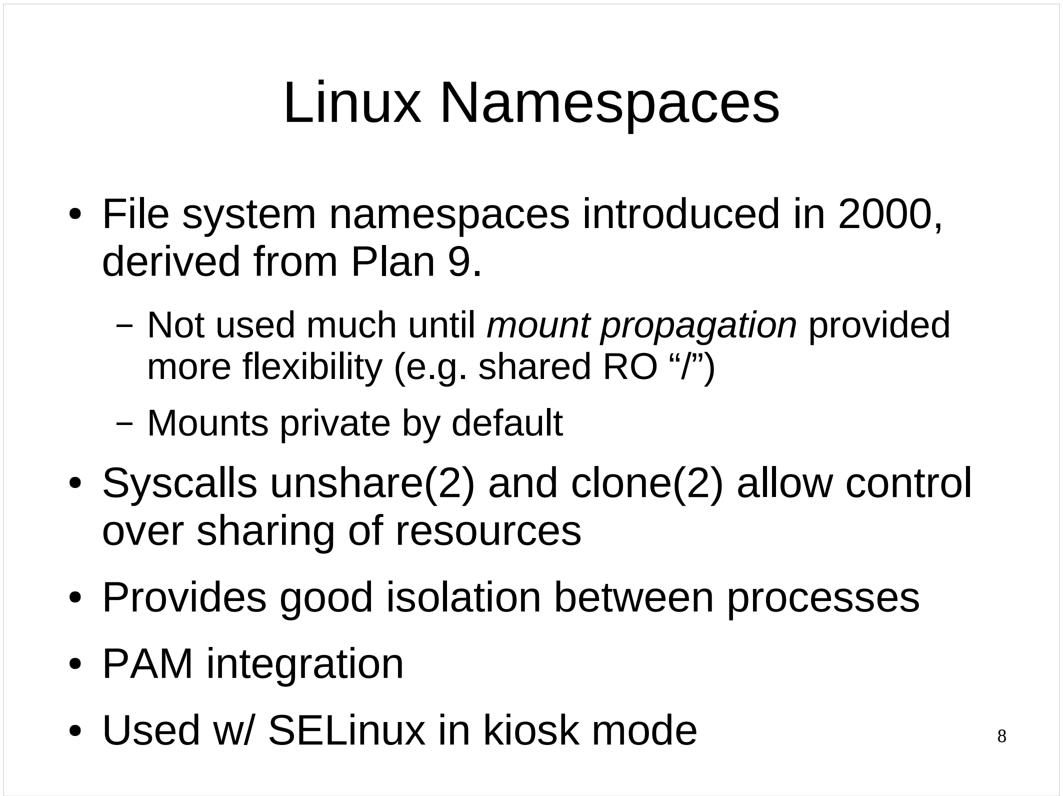
- This was recognized as a problem 30 years ago!



Extended DAC

- POSIX Capabilities (privileges)
  - Process-based since Linux kernel v2.2
    - Limited usefulness
  - File-based support relatively recent (v2.6.24)
    - May help eliminate setuid root binaries
- Access Control Lists (ACLs)
  - Based on abandoned POSIX spec
  - Uses extended attributes API

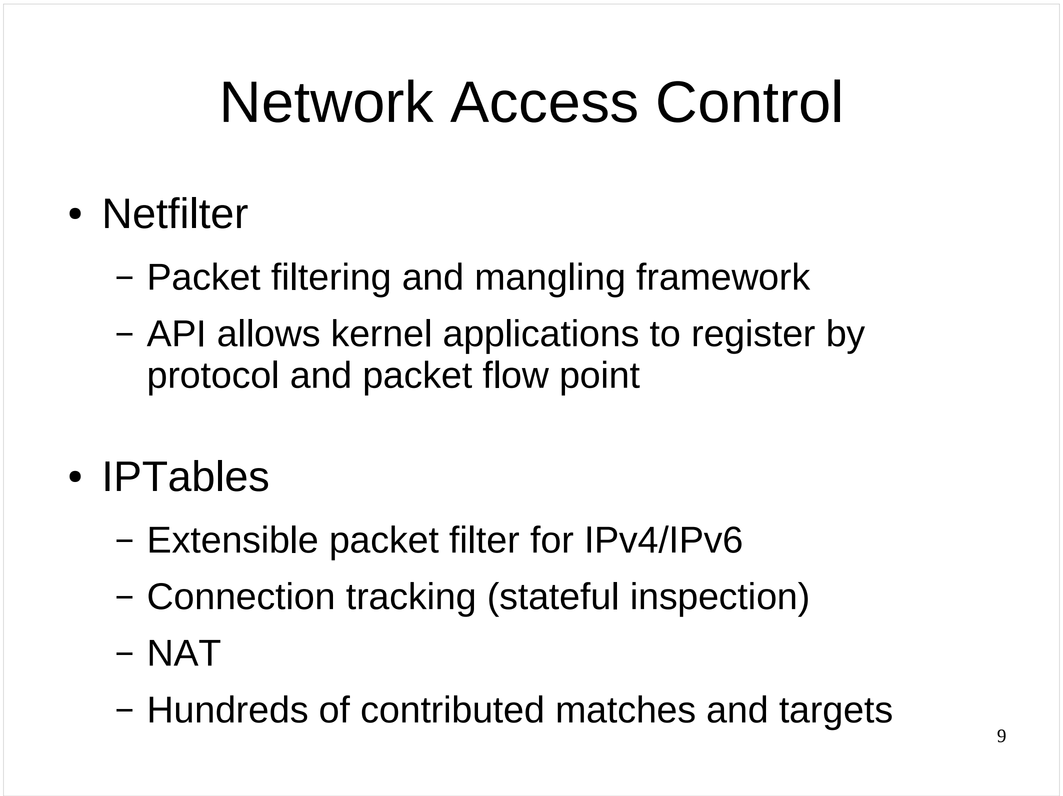
- Let’s solve DAC by... **adding more!**
- Proc caps have had some use w/ sendmail & ntpd
- Useful as annotations, has had some limited use (sendmail, ntpd), also involved in security issue
- File caps much more useful, but still to see distro adoption
- ACLs much more fine-grained and powerful than Unix perms, also subtle and complex; many different implementations
- See paper by Andreas Grünbacher
  also LCA slides on NFS ACLs by Greg Banks
- Capabilities not sufficient fundamentally: don’t take object security into account; fixed security model re. Inheritance & propagation which hinders least privilege; don’t protect trustworthy app from untrusted input (no information flow control!)
- setuid becoming less of an issue now than DBUS, which needs MAC



Linux Namespaces

- File system namespaces introduced in 2000, derived from Plan 9.
  - Not used much until *mount propagation* provided more flexibility (e.g. shared RO “/”)
  - Mounts private by default
- Syscalls unshare(2) and clone(2) allow control over sharing of resources
- Provides good isolation between processes
- PAM integration
- Used w/ SELinux in kiosk mode

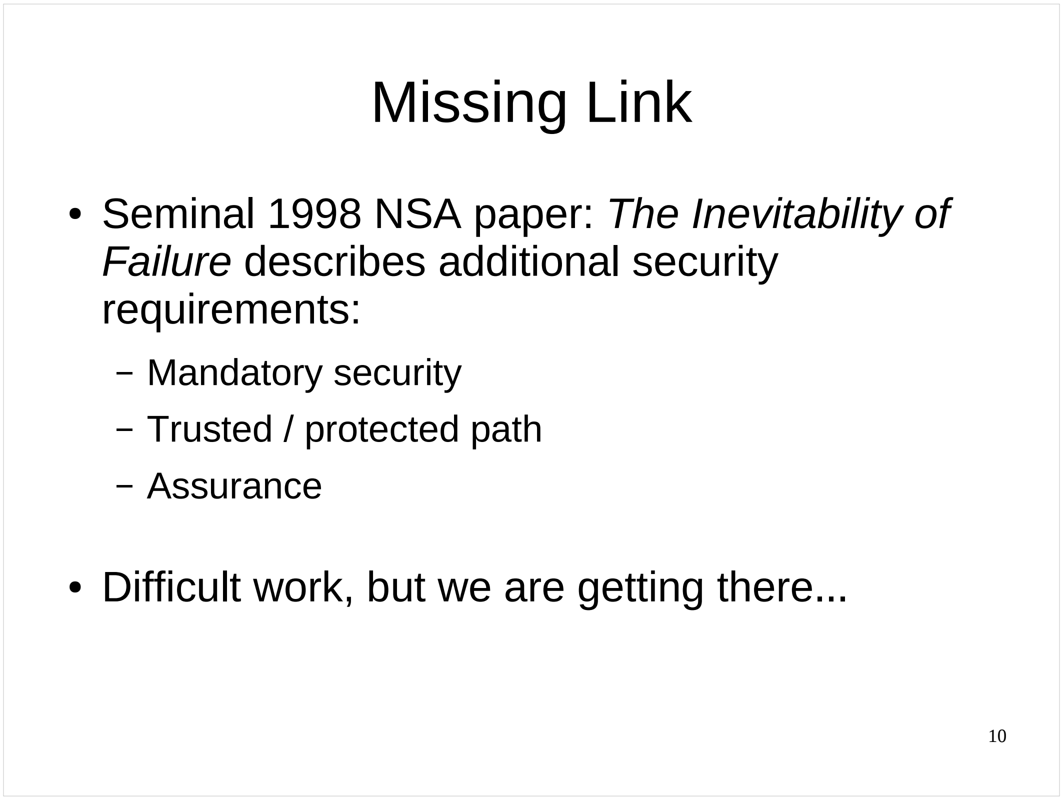
- Similar in concept to Solaris zones; also only provide isolation, which is useful, but also need to control sharing, provide protection inside container and manage entire system securely.
- bind mounts allow mount to appear in different places with different attributes, e.g. ro mount of /, private mount of $HOME & $TMP, with tmpfs.
- This can be managed manually, but is better done with PAM integration (e.g. pam_namespace)
- Was also developed/used for LSPP certification (MLDs / polyinstantantiation)
- Refer to kiosk mode anatomy slides
- Lots of ongoing work with namespaces and containers
- Demo kiosk mode?



Network Access Control

- Netfilter
  - Packet filtering and mangling framework
  - API allows kernel applications to register by protocol and packet flow point
- IPTables
  - Extensible packet filter for IPv4/IPv6
  - Connection tracking (stateful inspection)
  - NAT
  - Hundreds of contributed matches and targets

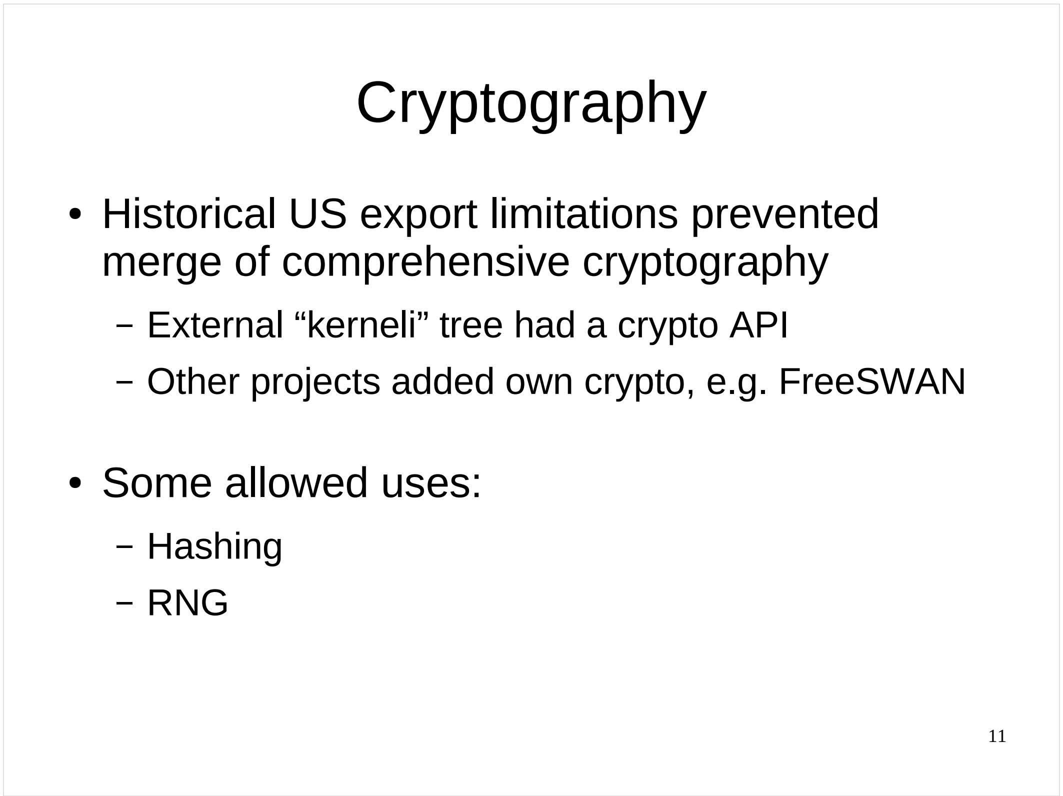
- Several generations of packet filtering prior to this: ipfw, ipchains
- Generalization, consolidation of packet flow
- Highly pluggable and extensible design
- Netfilter could support other packet filters, some efforts in this area, not mainlined
- Netfilter implemented at network layer, generic support for L3 protocols
- iptables plugins support many IP-based protocols, e.g. FTP + conntrack
- Also have bridging support with similar framework
- Firewalls are not enough alone: they’re too far from the host systems & very coarse granularity (see Inevitability paper).



Missing Link

- Seminal 1998 NSA paper: *The Inevitability of Failure* describes additional security requirements:
  - Mandatory security
  - Trusted / protected path
  - Assurance
- Difficult work, but we are getting there...

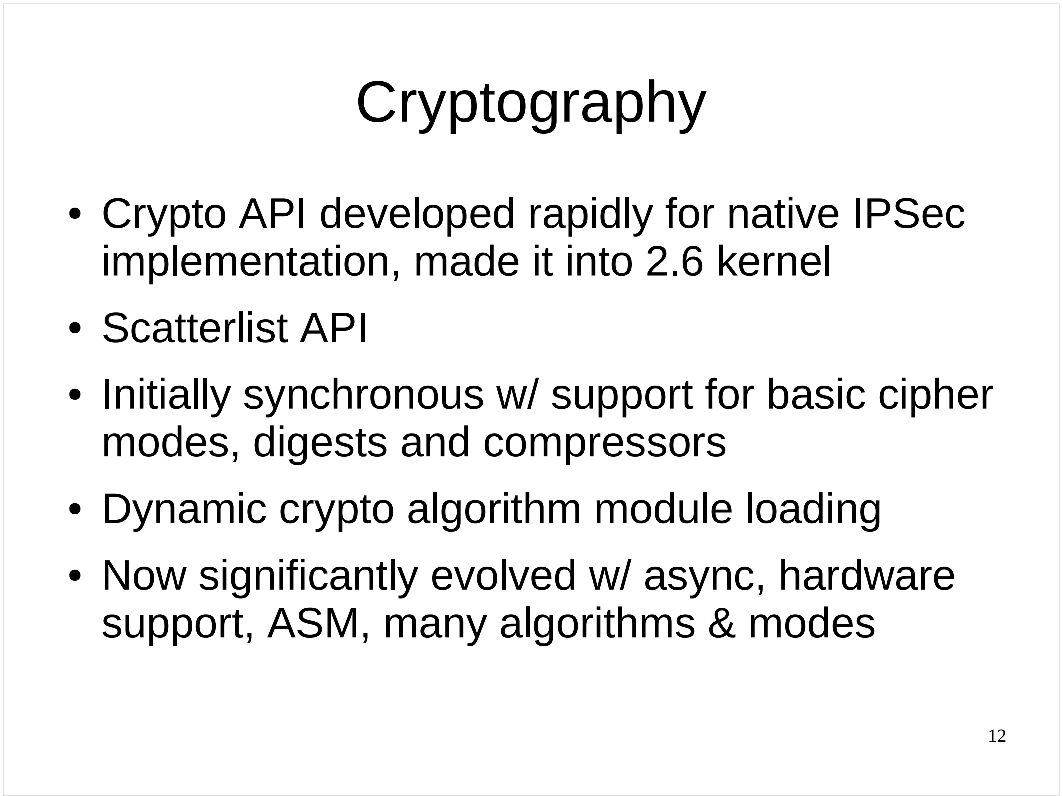
- Trusted path is a mechanism which provides confidence that: user is interacting with trusted application (trusted as in, trusted to perform the desired function, e.g. login); trusted app is interaction with actual user; also requires protection of communication channel
- Protected path is a generalization of trusted path; where all endpoints communicate via mutually authenticated channels (this can be extended to the network); e.g. prevent impersonation of cryptographic token invocation & security bypass in general
- Mandatory security -> MAC (SELinux etc), MIC, MCP (cf. Gutmann)
- Assurance: the most difficult; can include certifications, code audit etc.; FOSS improves assurance by providing source to users for verification
- Note that firewalls etc. depend on this to be



Cryptography

- Historical US export limitations prevented merge of comprehensive cryptography
  - External “kerneli” tree had a crypto API
  - Other projects added own crypto, e.g. FreeSWAN
- Some allowed uses:
  - Hashing
  - RNG

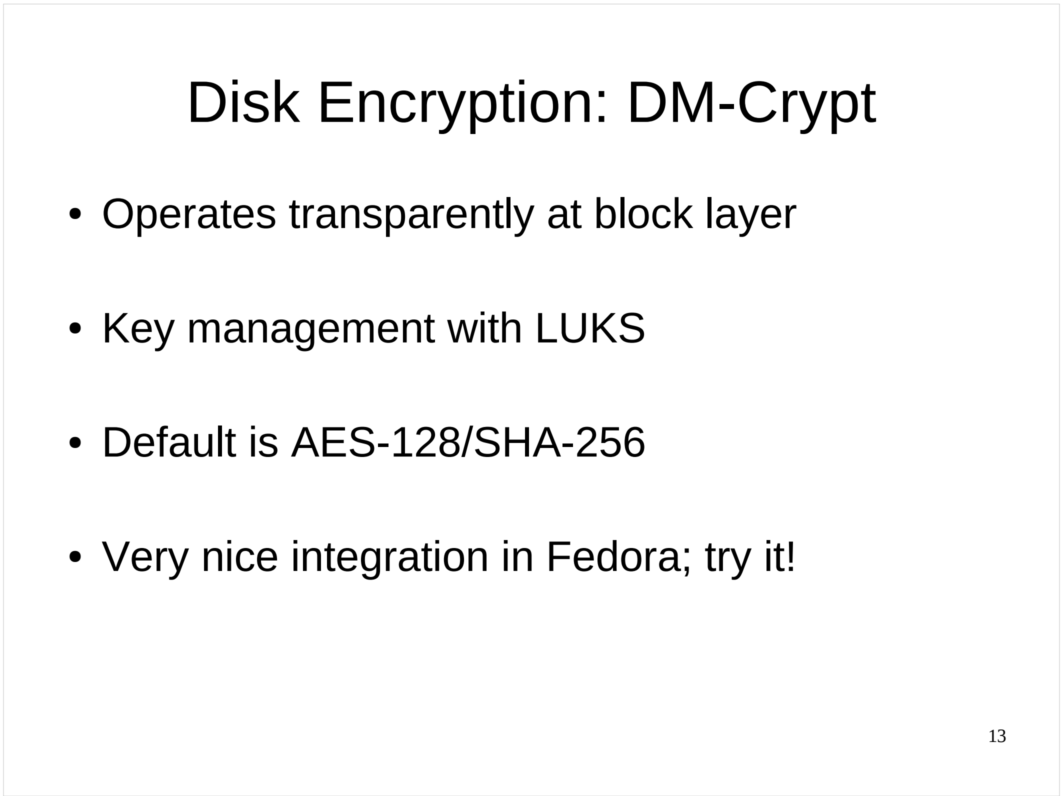
- The crypto export restrictions had the effect of preventing the merge; it would have caused enormous problems for linux distribution
- Hashing was not seen as being able to provide confidentiality, so has been present in the kernel for ages
- Changes to the laws allowed export with notification
- Note that crypto is not security (older common viewpoint); it is a component of security which requires a secure OS to function effectively



Cryptography

- Crypto API developed rapidly for native IPSec implementation, made it into 2.6 kernel
- Scatterlist API
- Initially synchronous w/ support for basic cipher modes, digests and compressors
- Dynamic crypto algorithm module loading
- Now significantly evolved w/ async, hardware support, ASM, many algorithms & modes

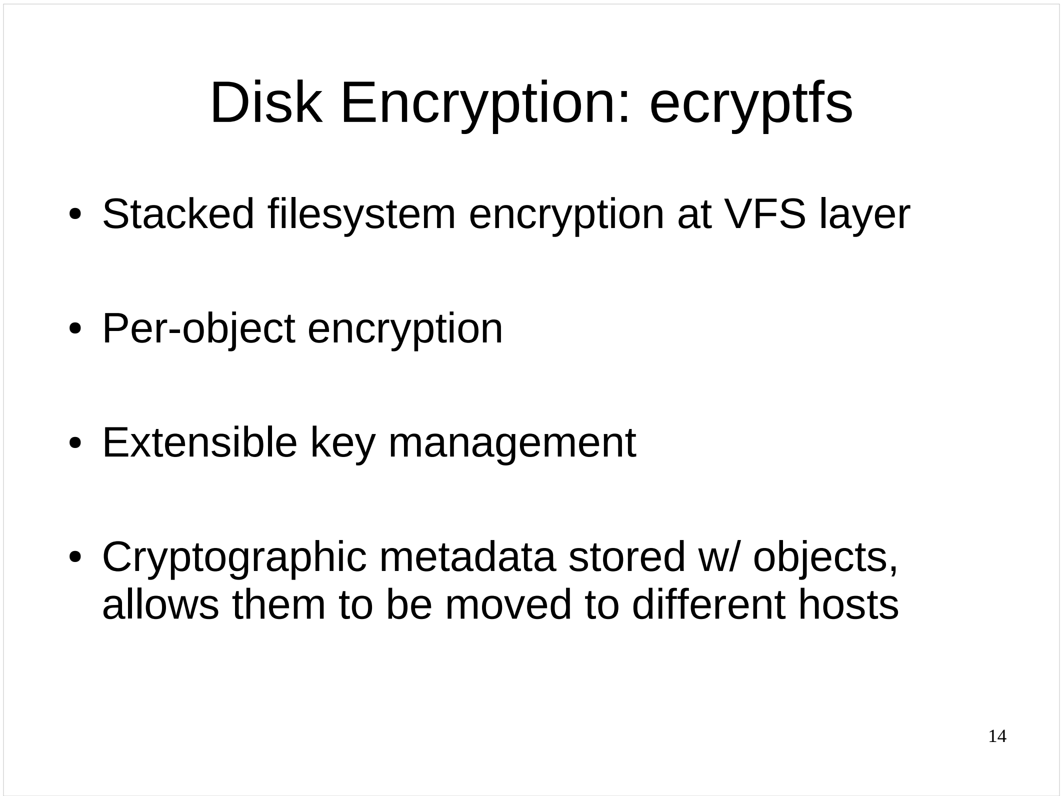
- I developed a crypto API based on several open source projects including kerneli and Nettle
- Design input from Linus and Dave Miller
- Took about 5 weeks for the initial API to be merged, had basic support for ciphers (symmetric), digests (and HMAC), and compressors; used scatterlist (vectored) API to facilitate deep kernel integration
- Was necessary for IPSec & made it in for 2.6 kernel
- Handed maintenance to Herbert Xu, who has done great work extending the modes, algorithms, scope
- Herbert will be speaking on this at LPC in PDX.
- *Are the T2 on-chip crypto specs available?*



Disk Encryption: DM-Crypt

- Operates transparently at block layer
- Key management with LUKS
- Default is AES-128/SHA-256
- Very nice integration in Fedora; try it!

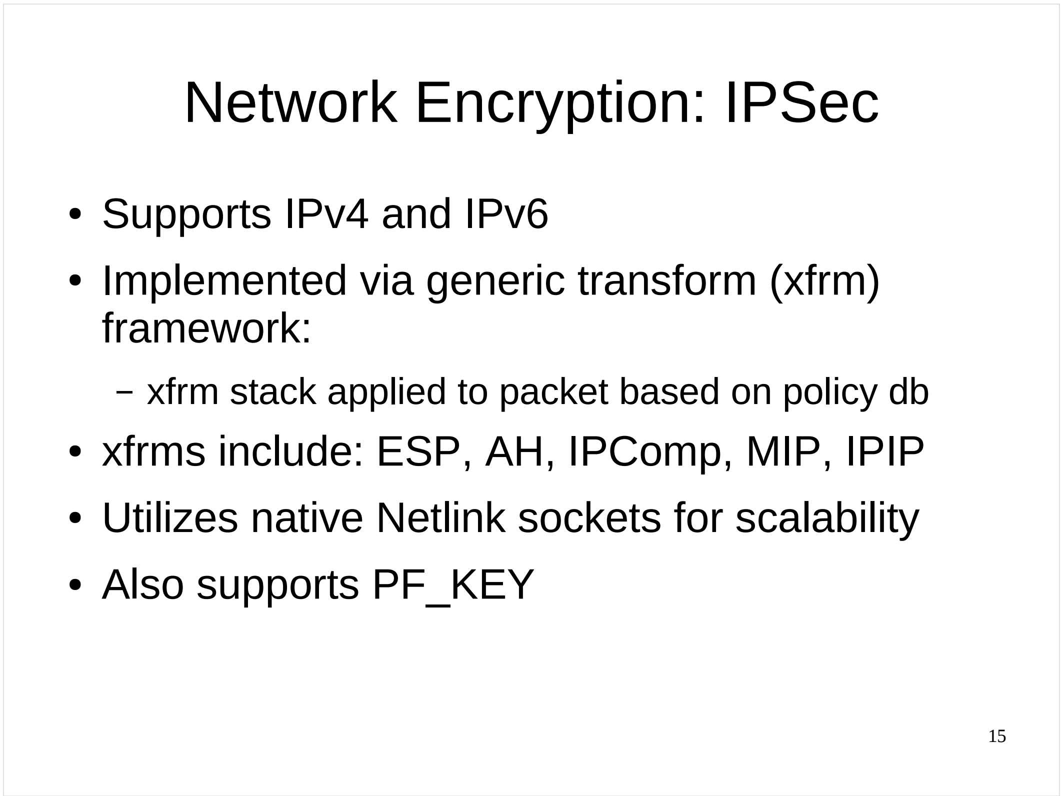
- There are many schemes available, these are some of the main ones in use.
- Linux Unified Key Setup-on-disk-format (LUKS)
- DM = device mapper, block layer plugins, allows also for things like software raid, integration with LVM
- Block layer crypto is good because it’s simple, allows encryption of RAID arrays, LVM volumes etc.
- Unmodified fs
- Swap support
- Lacks granularity



Disk Encryption: ecryptfs

- Stacked filesystem encryption at VFS layer
- Per-object encryption
- Extensible key management
- Cryptographic metadata stored w/ objects, allows them to be moved to different hosts

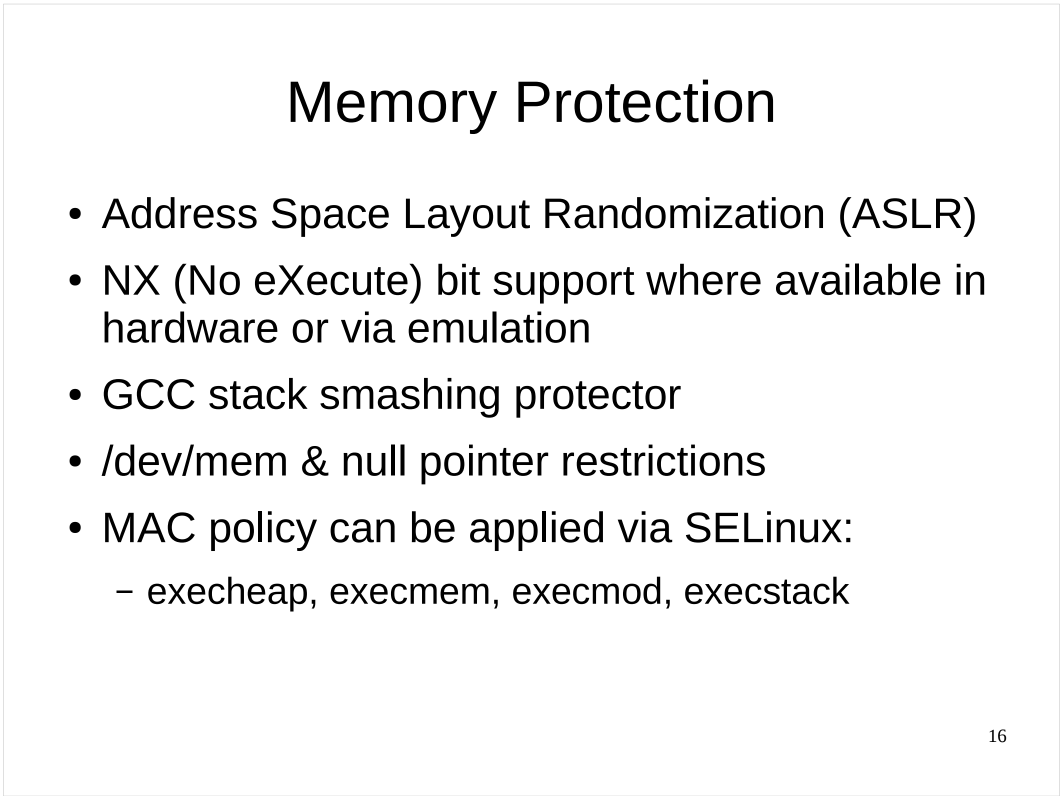
- Addresses many use cases where finer granularity is required, such as incremental backups, sharing files etc.
- Different algorithms for different objects
- Saves re-encrypting for transmission
- Selective use on fs, saves overhead
- TPM, PKCS#11 etc. for key management.
- Files appear normal in Base FS, POSIX compliance, backup etc. works as expected.



Network Encryption: IPSec

- Supports IPv4 and IPv6
- Implemented via generic transform (xfrm) framework:
  - xfrm stack applied to packet based on policy db
- xfrms include: ESP, AH, IPComp, MIP, IPIP
- Utilizes native Netlink sockets for scalability
- Also supports PF_KEY

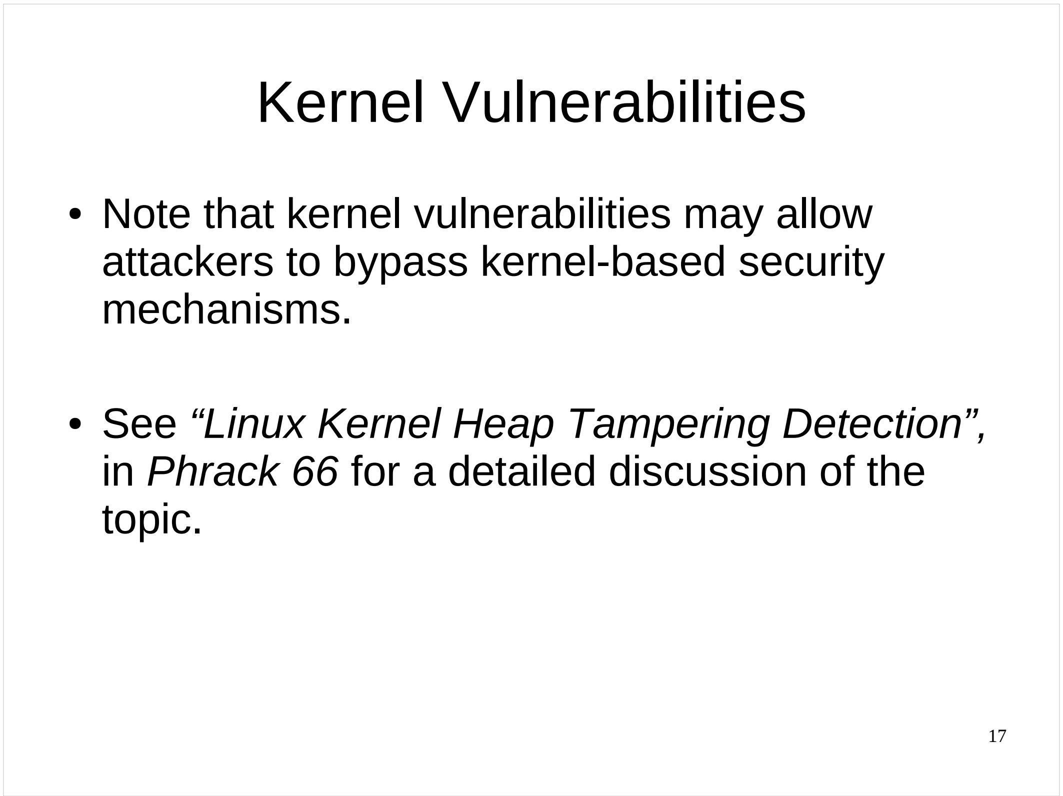
- Native IPsec stack made possible by crypto policy changes, designed and implemented by DaveM and Alexey; unorthodox design aimed at max. performance and utility
- By the time this became available, many people were using other non-kernel crypto, e.g. SSH, SSL, userland VPNs.
- This stack is used in commercial appliances, so you may be using it anyway...



Memory Protection

- Address Space Layout Randomization (ASLR)
- NX (No eXecute) bit support where available in hardware or via emulation
- GCC stack smashing protector
- /dev/mem & null pointer restrictions
- MAC policy can be applied via SELinux:
  - execheap, execmem, execmod, execstack

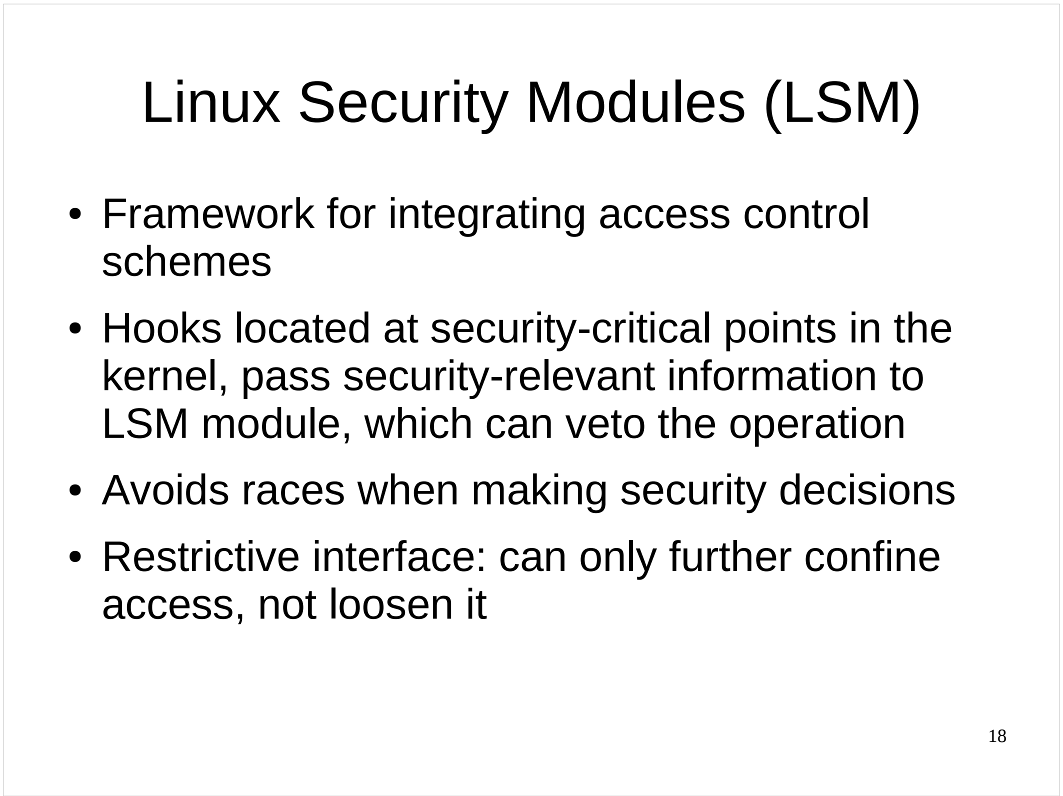
- Several schemes for resisting memory-based attacks, depending on which distro and hardware you use
- ASLR: randomizes various aspects of application address space: libraries, heap, stack, text; has been broken
- See ExecShield (may be dropped soon b/c not upstreamable & hw does it better), Mark Cox and Drepper’s docs
- Much of this work comes from grsecurity / pax / openwall
- Some of this is done in conjunction w/ userspace, e.g. glibc and elf hardening
- The usability/security tradeoff of the linux protections has come under criticism, some of it warranted
- Several external projects feed patches and help in, not always successfully
- nx emulation uses segment limits
- FORTIFY_SOURCE



Kernel Vulnerabilities

- Note that kernel vulnerabilities may allow attackers to bypass kernel-based security mechanisms.
- See *“Linux Kernel Heap Tampering Detection”,* in *Phrack 66* for a detailed discussion of the topic.

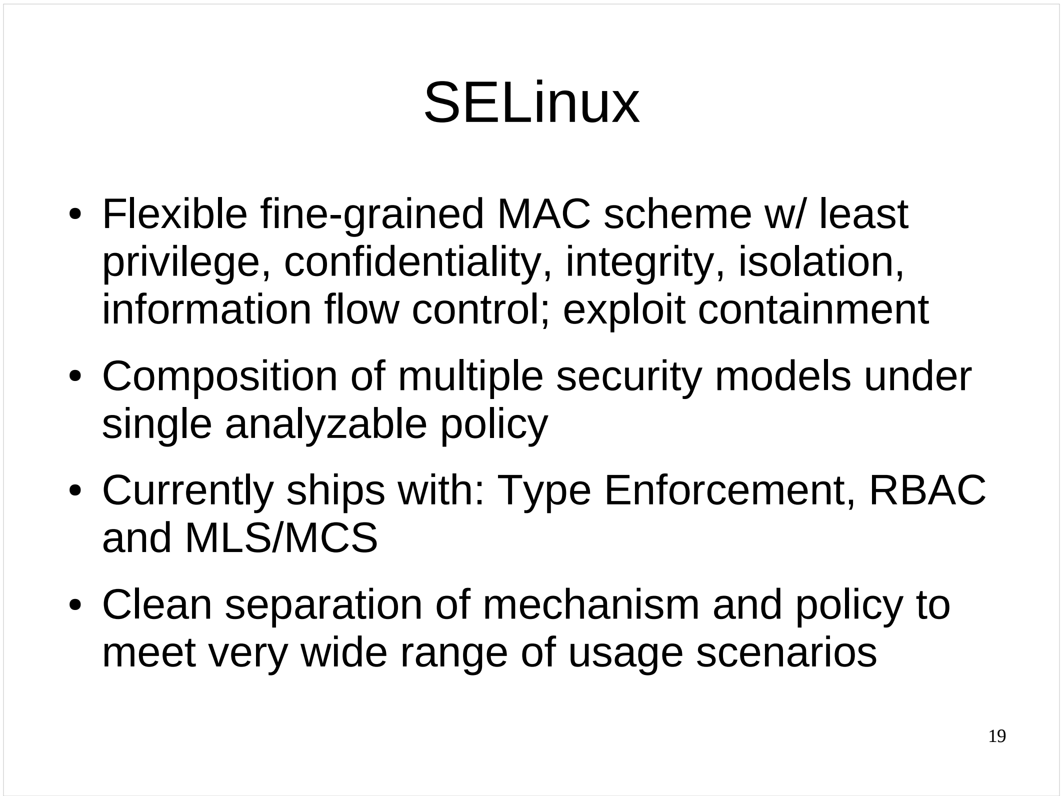
- grsecurity folk have been working in this area; some of it is likely not upstreamable
- LKIM addresses this; see referenced ACM paper; code not currently available as open source
- A kernel vulnerability can arise from almost any kernel bug – it may not be recognizable as a security bug to even the most experienced kernel developer.
- One mechanism used by developers who suspect their bug is security related is to notify vendor-sec, which includes the security response folk from all of the major vendors, for analysis & coordination.
- Linus’ policy is to simply fix all bugs (“security is not special”) without fanfare; this is controversial but does have one clear benefit: the bug is fixed.



Linux Security Modules (LSM)

- Framework for integrating access control schemes
- Hooks located at security-critical points in the kernel, pass security-relevant information to LSM module, which can veto the operation
- Avoids races when making security decisions
- Restrictive interface: can only further confine access, not loosen it

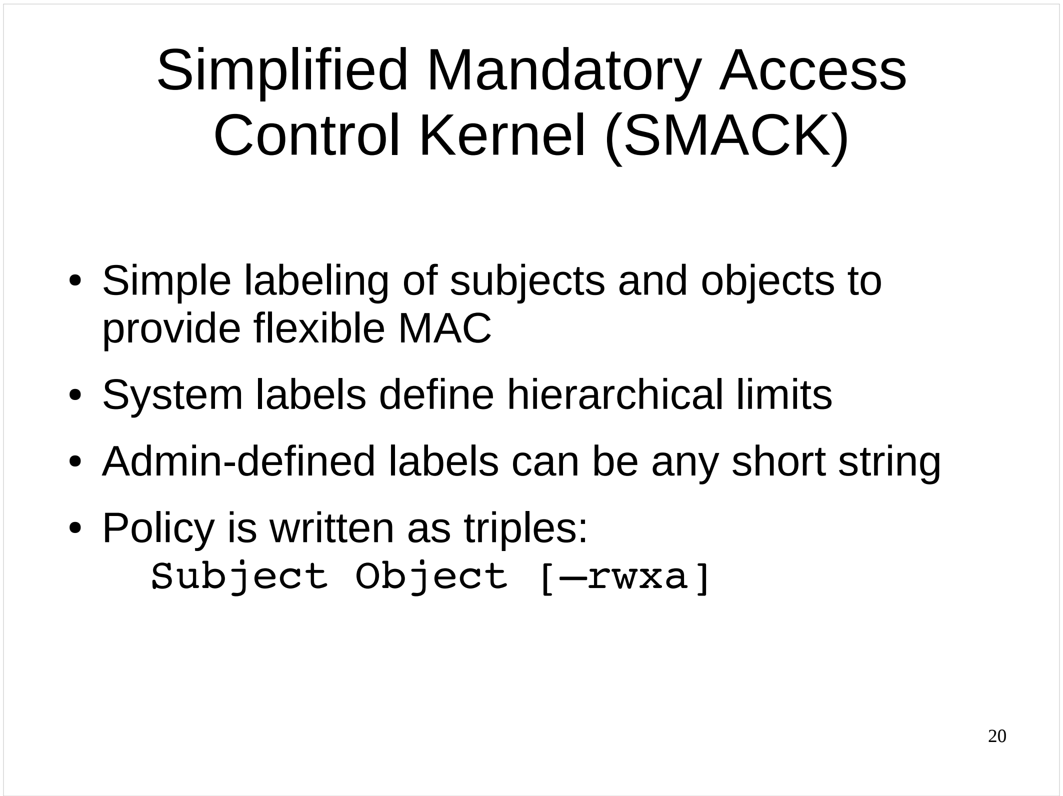
- Developed in response to Linus’ initial reaction to SELinux, where he did not want to decide on a security model for the kernel, so make it pluggable
- AppArmor, SELinux, SGI etc. developers worked on it, then these were ported to LSM, some new LSMs developed
- Lots of controversy subsequently as SELinux remained the only significant user; Linus reiterated his position that as there was no consensus on security model, LSM remains; use Arjan protocol for reviewing new modules to avoid flamewars
- Drawbacks include weak semantics, lack of consistent security model for ISVs / admins etc.
- One benefit is diversity of ideas (D. Wagner)
- Related work: BSD MAC framework, XSM, XACE



SELinux

- Flexible fine-grained MAC scheme w/ least privilege, confidentiality, integrity, isolation, information flow control; exploit containment
- Composition of multiple security models under single analyzable policy
- Currently ships with: Type Enforcement, RBAC and MLS/MCS
- Clean separation of mechanism and policy to meet very wide range of usage scenarios

- Rationale:
  - “trusted” systems not viable / generally useful
  - Need whole-system approach (i.e. extend to network, database, virt, desktop...)
  - Mainstream MAC
- Targeted policy: limited confinement to network facing services and base OS: made it possible to enable by default
- Proven effectiveness, limits exploitation of vulns
- Usability addressed with high level abstractions, e.g. kiosk mode, svirt
- Related work: SEBSD, FMAC; interop desired
  - Kylin 3 (KACF), apparently “B2” class
- Certified LSPP/EAL4+, also shipping enabled by default in Fedora
- Low-level policy is complex; relies on high level abstractions for usability, like a spreadsheet on a PC.
- Customization is still a challenge; work to be done..
- Strong developer community

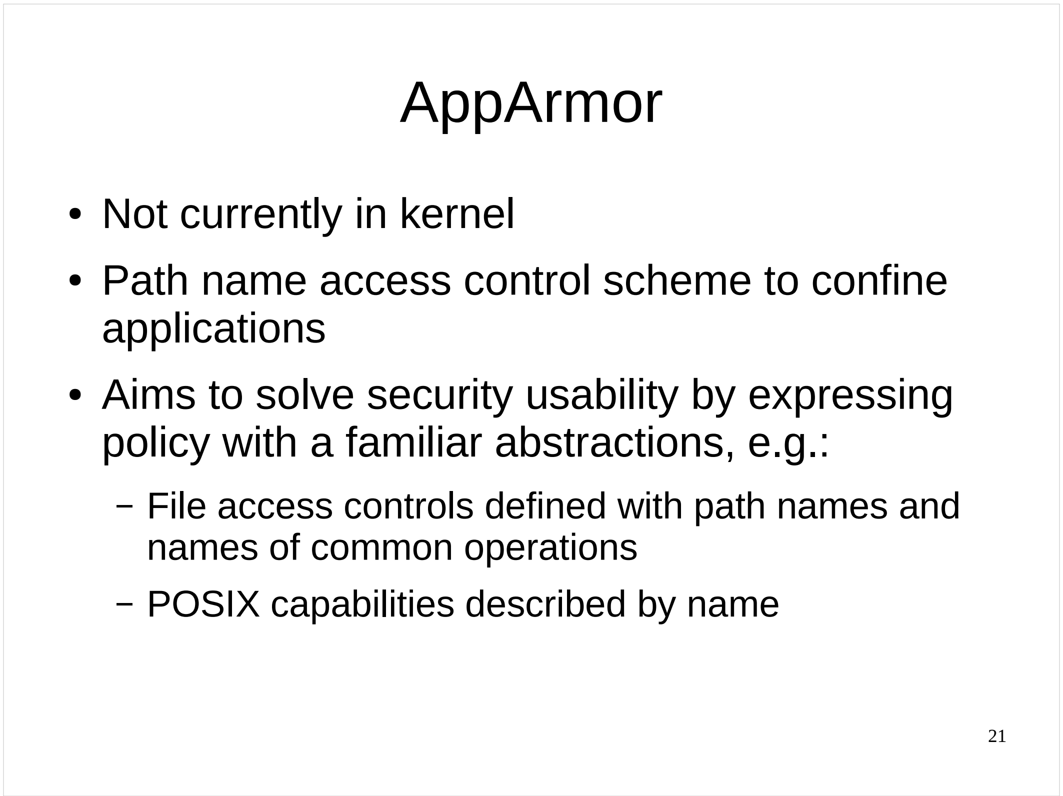


Simplified Mandatory Access Control Kernel (SMACK)

- Simple labeling of subjects and objects to provide flexible MAC
- System labels define hierarchical limits
- Admin-defined labels can be any short string
- Policy is written as triples:

```
Subject Object [—rwxa]
```

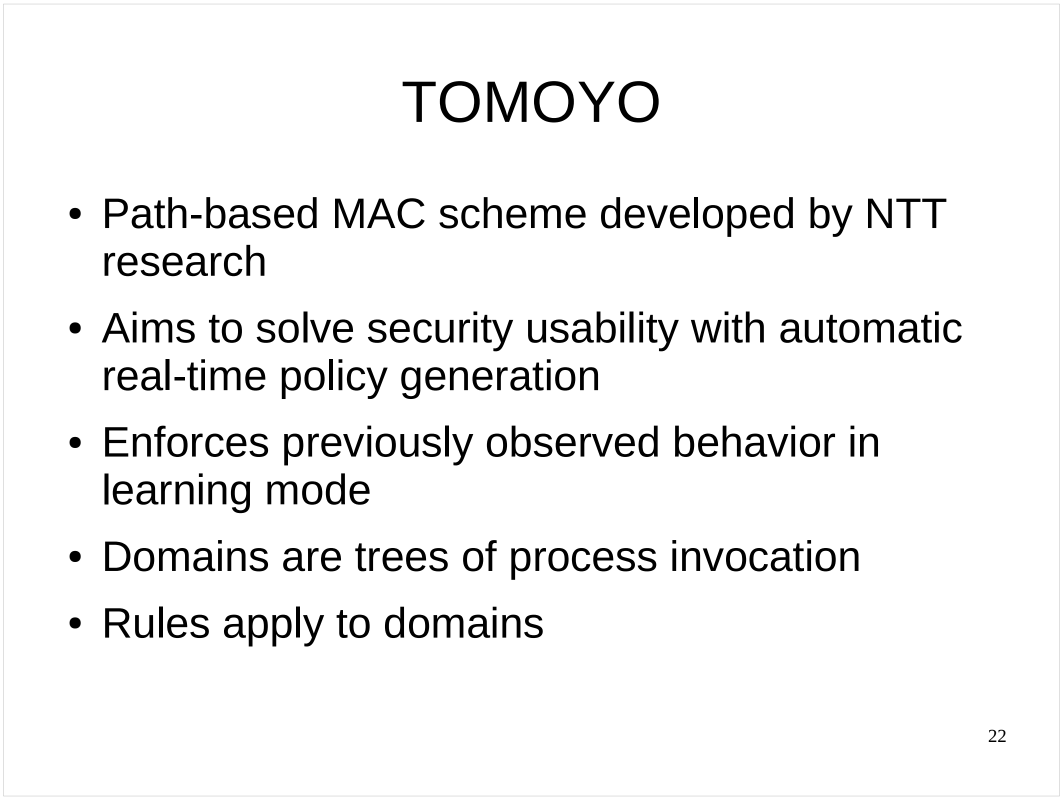
- Developed by Casey, who has a long history with Trusted OSs and is aware of their drawbacks
- System labels: hat / floor are like system high/low
- Some of the simplification appears genuinely useful (e.g. for sockets), although overall it leads to coarser and thus less expressive policy foundation (perhaps something like CISC vs. RISC)
- Need more analysis of efficacy and practical demonstration to be able to evaluate, but should be able to achieve useful security goals
- Not aware of fielded systems using SMACK as yet
- Is it too simple to be generally useful ?



AppArmor

- Not currently in kernel
- Path name access control scheme to confine applications
- Aims to solve security usability by expressing policy with a familiar abstractions, e.g.:
  - File access controls defined with path names and names of common operations
  - POSIX capabilities described by name

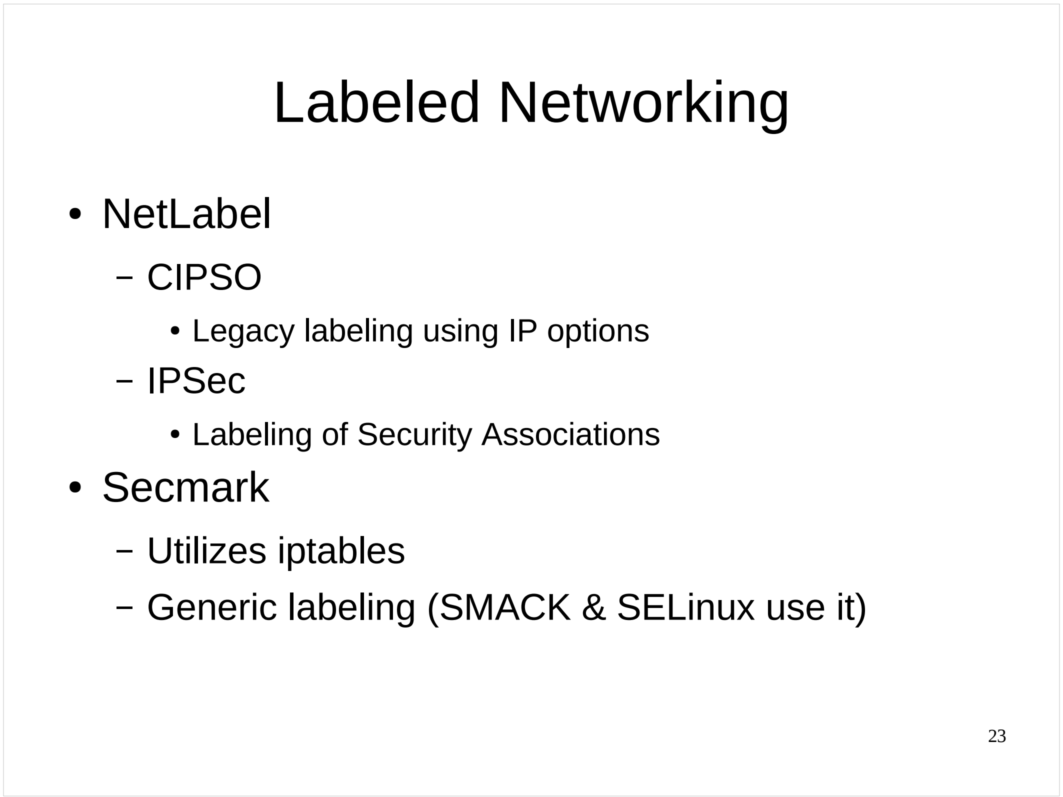
- The pathname aspect has been contentious; critics are concerned with object aliasing (aka forgeable references), incomplete mediation, that the model does not generalize, and will not ultimately be as simple as expected
- Similar concept to “No Fly List”: assess the name of the object instead of the object itself;
- Changes have been made to the scheme which address some of the aliasing issues (e.g. more control over linking), and advocates are ok with the usability/security trade-off
- The flamewar aspect is overblown: it is normal and expected for security engineers to robustly analyze each others work, also part of Internet culture.
- Linus has taken the “I don’t like your security model” argument off the table



TOMOYO

- Path-based MAC scheme developed by NTT research
- Aims to solve security usability with automatic real-time policy generation
- Enforces previously observed behavior in learning mode
- Domains are trees of process invocation
- Rules apply to domains

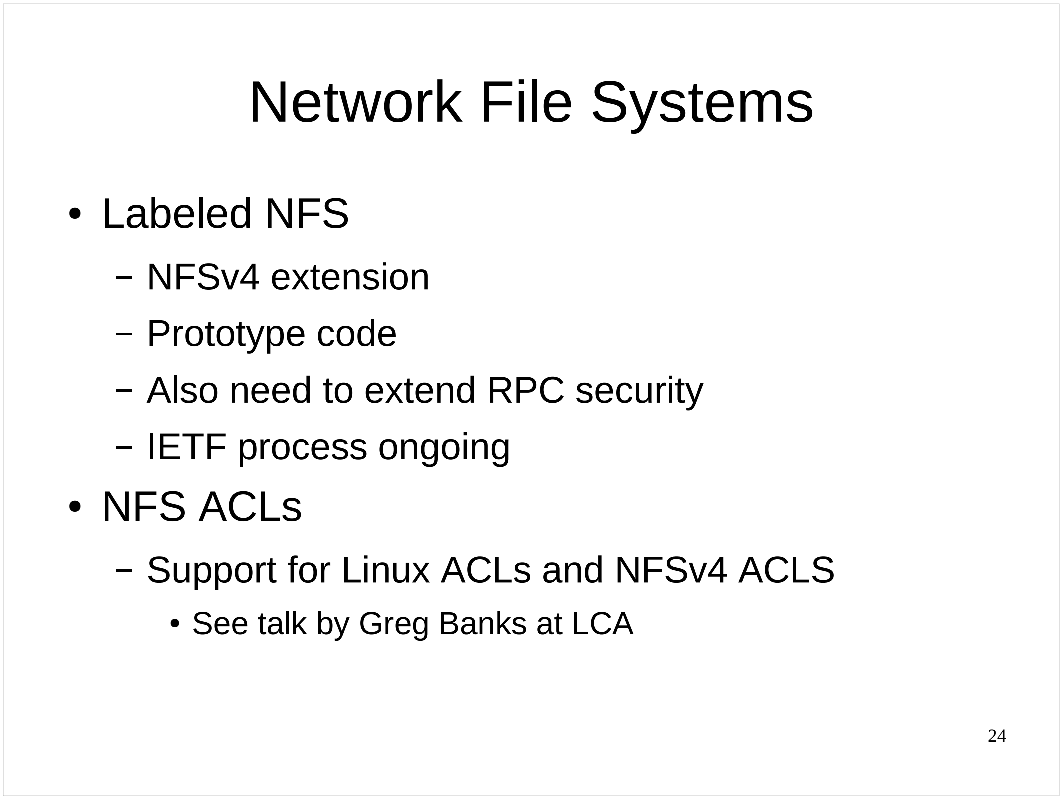
- R&D project from NTT; bosses told developers to make something new
- Attempts to solve usability with automated policy generation
- Pathnames are labels
- No MLS, no RBAC,
- “Task Oriented Management Obviates Your Onus on Linux”
- Aimed at average users and admins, not security professionals (according to their docs)
- Not clear how “status quo encapsulation” or “unobserved but valid code path execution” is addressed.



Labeled Networking

- NetLabel
  - CIPSO
    - Legacy labeling using IP options
  - IPSec
    - Labeling of Security Associations
- Secmark
  - Utilizes iptables
  - Generic labeling (SMACK & SELinux use it)

- This is quite a complicated area overall, and while the code is essentially complete and likely useful for advanced users, it will take time for suitable general purpose abstractions and applications to evolve
- Secmark is “local” labeling and does not require protocol support anywhere; NetLabel is “remote” labeling and requires protocol support at each end.
- Labeled networking ultimately is required to extend the trusted path concept over the network, and is as such an essential component of securing the networked systems of the future.

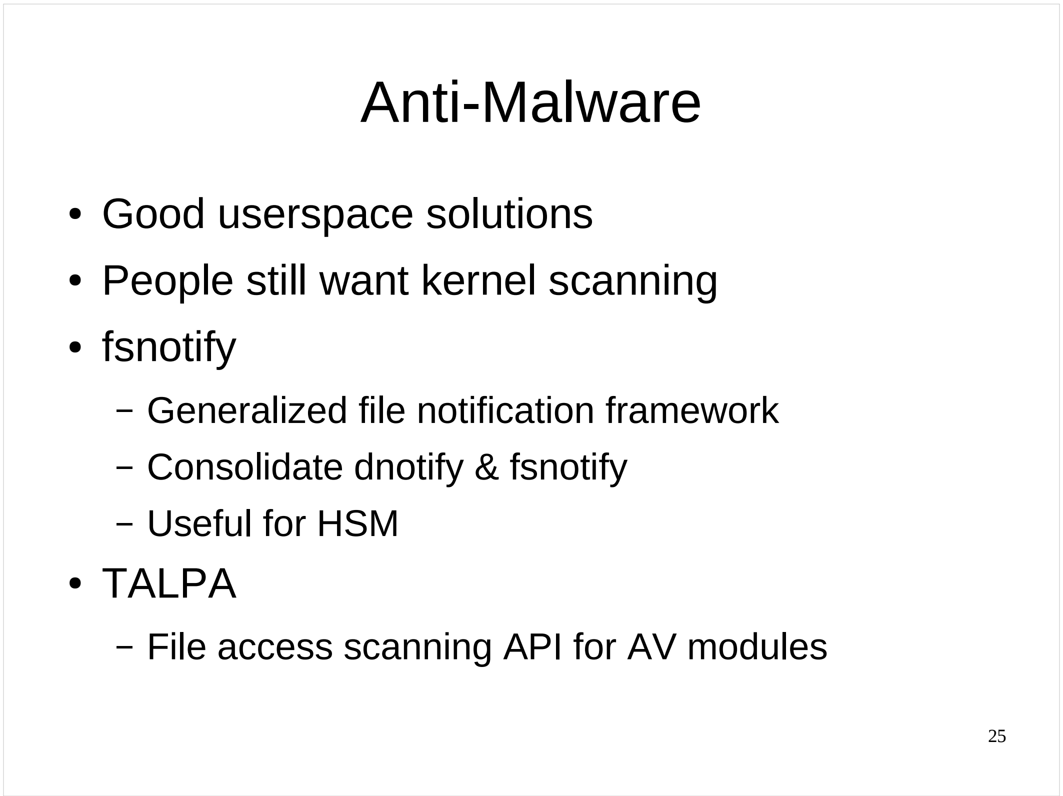


Network File Systems

- Labeled NFS
  - NFSv4 extension
  - Prototype code
  - Also need to extend RPC security
  - IETF process ongoing
- NFS ACLs
  - Support for Linux ACLs and NFSv4 ACLS
    - See talk by Greg Banks at LCA

Labeling:
- Several closed / proprietary implementations
- No standard
- Best if open and standard, generalized

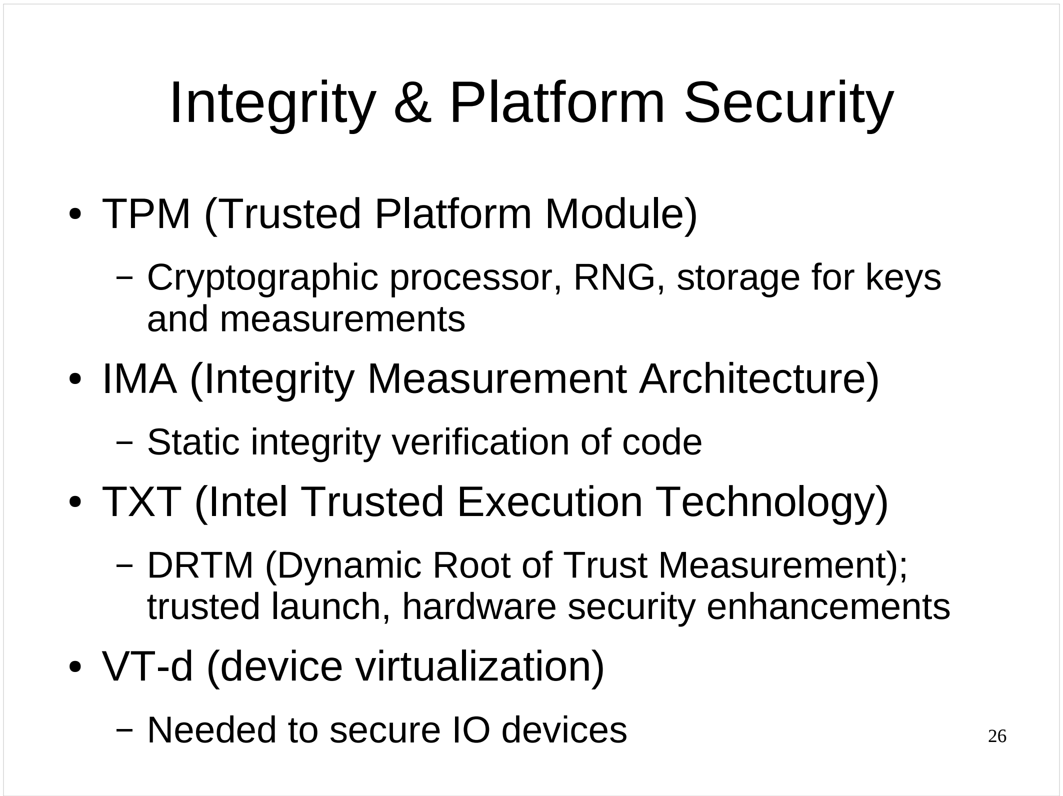
ACLs:
- Lots of variations, need interop
- NFSv3 has support for native ACLs
  - uses xattr APIs
- NFSv4 ACLs are implementation of Windows model, which is very different (see Greg's slides)
- NFSv4 code partially implemented
- ZFS, GPFS has it
- Needed for NAS interop, windows clients etc.



Anti-Malware

- Good userspace solutions
- People still want kernel scanning
- fsnotify
  - Generalized file notification framework
  - Consolidate dnotify & fsnotify
  - Useful for HSM
- TALPA
  - File access scanning API for AV modules

- Problematic area; kernel devs not keen on in-kernel scanning, should mostly be done in userspace; but may have valid use-cases for network file servers
- AV companies not community oriented, often have proprietary kernel modules
- Eric Paris started working as intermediary, came up with fsnotify and TALPA
- TALPA; file access scanning API
- fsnotify merged, remaining status unclear

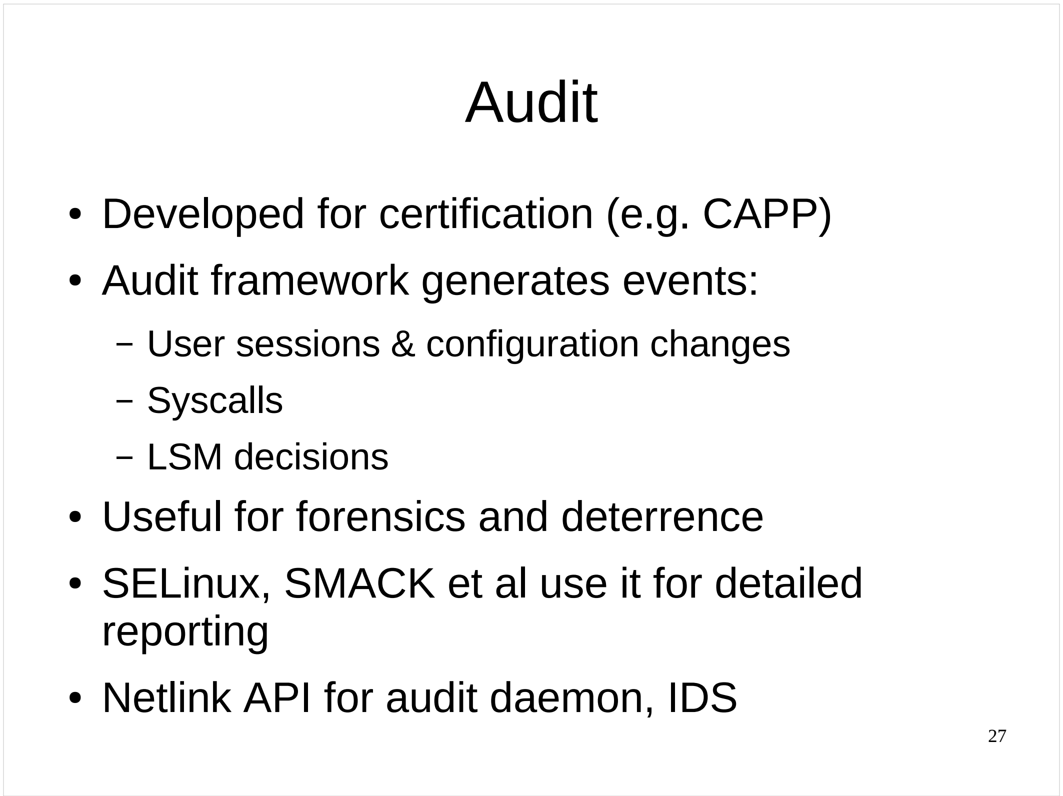


Integrity & Platform Security

- TPM (Trusted Platform Module)
  - Cryptographic processor, RNG, storage for keys and measurements
- IMA (Integrity Measurement Architecture)
  - Static integrity verification of code
- TXT (Intel Trusted Execution Technology)
  - DRTM (Dynamic Root of Trust Measurement); trusted launch, hardware security enhancements
- VT-d (device virtualization)
  - Needed to secure IO devices

- This is part of the missing link: need to protect kernel!

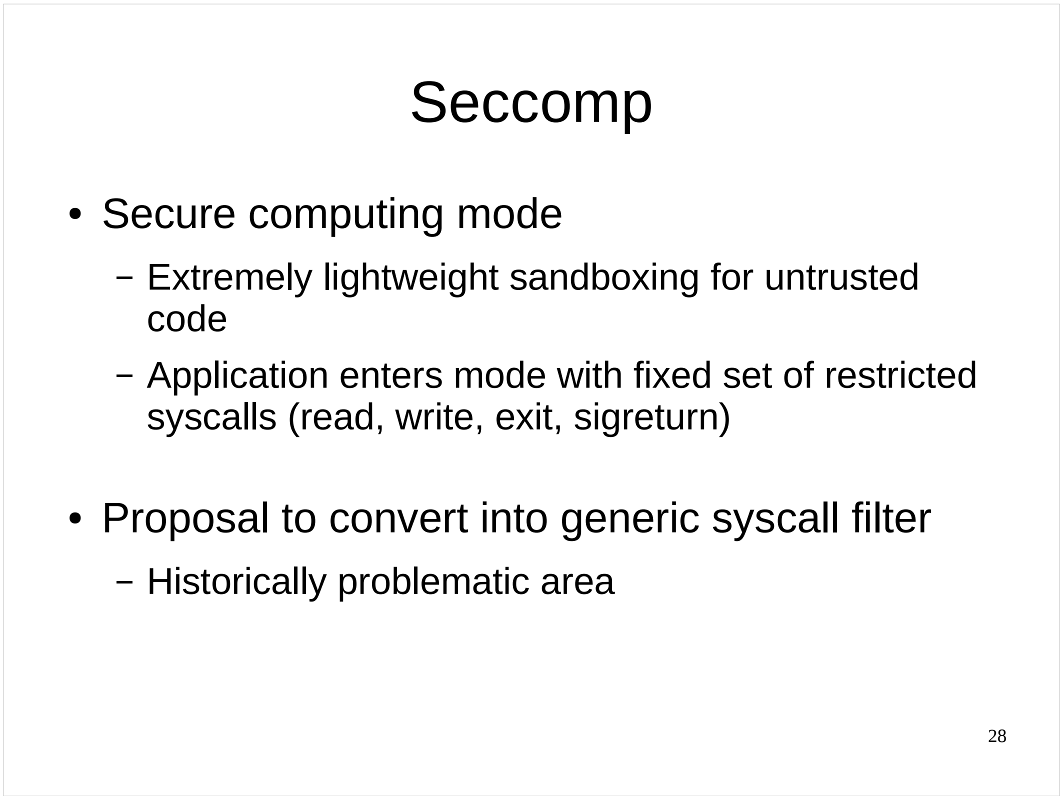
- Linux is @ leading edge of this & continues to advance
- DRM controversial (see TCPA rebuttal: TCPA can implement DRM, but is not itself DRM).
- TPM can be very useful:
  - BitLocker
  - Sealing credentials for PGP, SSL, SSH etc.
  - Bring trusted environment up on untrusted system
- Remote attestation
- Static root of trust too difficult to work with
- Dynamic root of trust more promising
    - TXT; do not have to trust everything!
- Invisiblethings Lab blog very useful
- Integrity Measurement Architecture (IMA) from IBM; LKIM / contextual inspection next step (runtime...)
- VT-d necessary to e.g. properly virtualize DMA



Audit

- Developed for certification (e.g. CAPP)
- Audit framework generates events:
  - User sessions & configuration changes
  - Syscalls
  - LSM decisions
- Useful for forensics and deterrence
- SELinux, SMACK et al use it for detailed reporting
- Netlink API for audit daemon, IDS

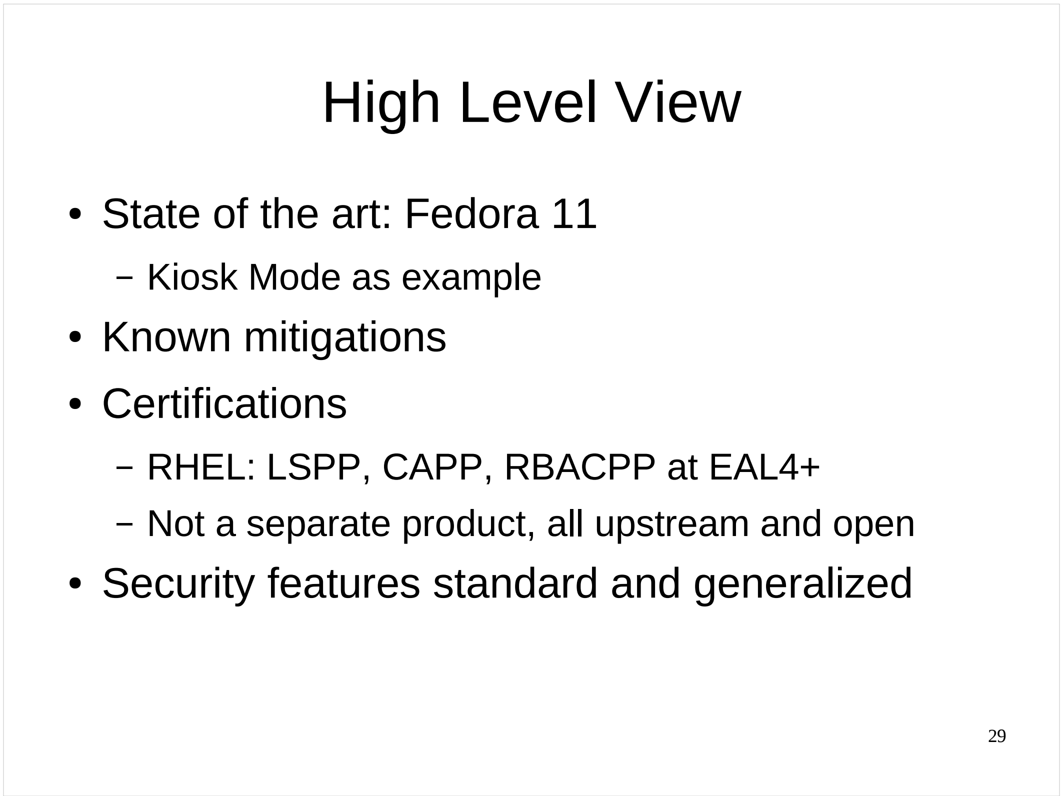
- Arguably not a security feature
- Standard feature of C2 / CAPP
- Developed for CAPP certification
- syscall auditing has performance issue which gets blamed on SELinux
- Promising general use: IDS (in development)
- Also has filtering in kernel



Seccomp

- Secure computing mode
  - Extremely lightweight sandboxing for untrusted code
  - Application enters mode with fixed set of restricted syscalls (read, write, exit, sigreturn)
- Proposal to convert into generic syscall filter
  - Historically problematic area

- Andrea Archangeli originally developed this for a grid computing business
- syscall wrapping considered harmful: LSM is the right way to go; see Robert Watson paper "Exploiting Concurrency Vulnerabilities in System Call Wrappers"
  - problems with races mainly
- Google investigating for Chrome:
  - proves point that lack of consistent security API is a problem
- LSM is right solution for hook points:
  - still inconsistent between distros



High Level View

- State of the art: Fedora 11
  - Kiosk Mode as example
- Known mitigations
- Certifications
  - RHEL: LSPP, CAPP, RBACPP at EAL4+
  - Not a separate product, all upstream and open
- Security features standard and generalized

- Tie back to inevitability goals

- Meets extremely wide range of needs, from end user desktop to military, stock exchanges etc.

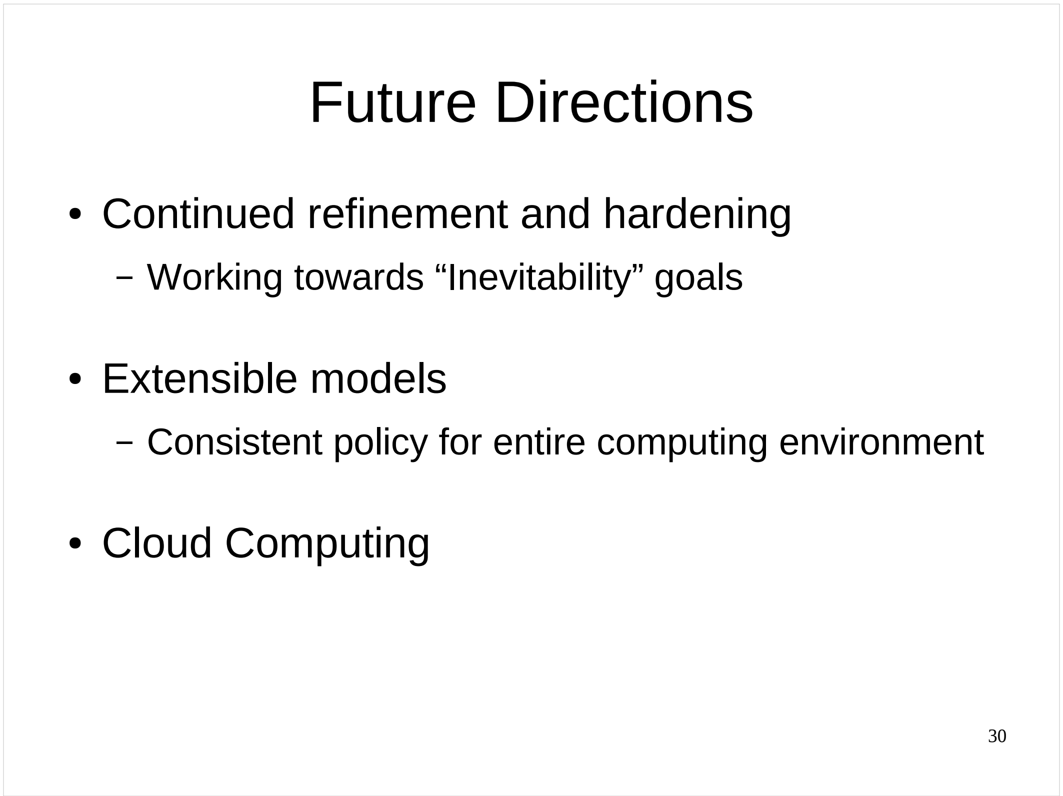
- Combined together, current code provides layered security (defense in depth; of course there is then attack in depth...), incremental improvements & retrofitting means we can deliver better security to large audience

- Has also sparked improvements in other OSs security

- Move to security as standard feature of OS is a major step in itself

- Several known exploits blocked, covered in LinuxWorld article in 2008

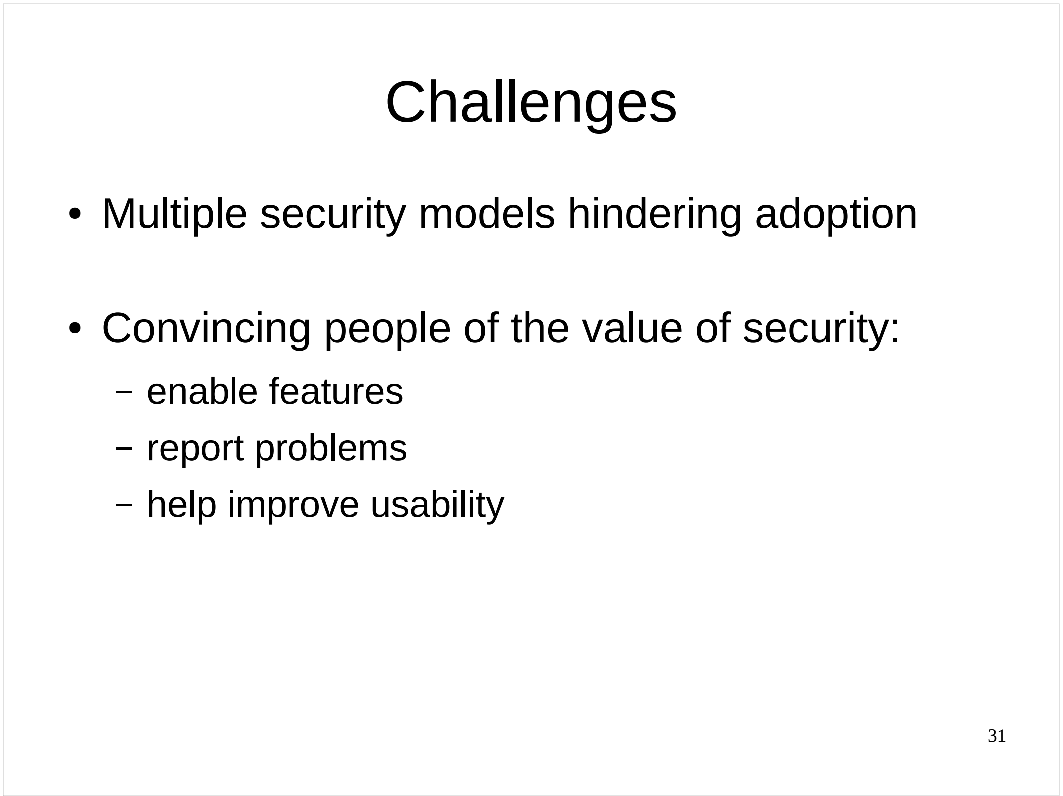
- CtF contests seem to be not using Fedora anymore...



Future Directions

- Continued refinement and hardening
  - Working towards “Inevitability” goals
- Extensible models
  - Consistent policy for entire computing environment
- Cloud Computing

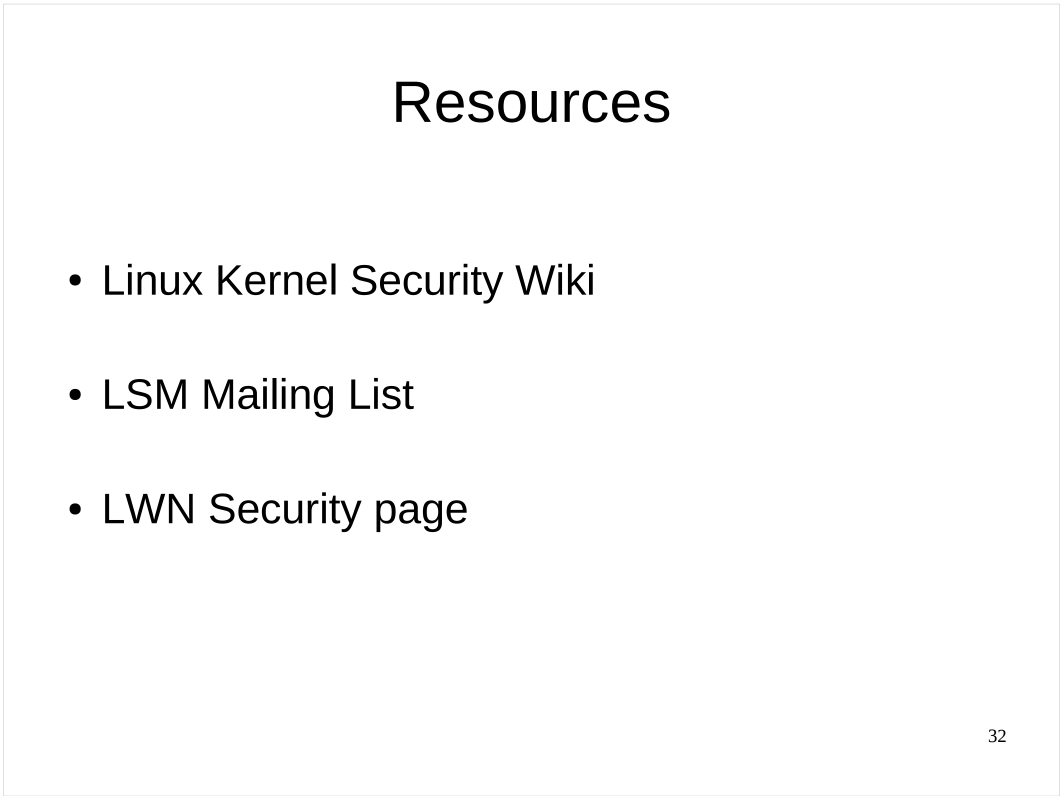
- This process will not end, refinement expected to be ongoing
- Computing environment evolving: virtualization, cloud etc. are just what we know about now... security needs to generalize in terms of technologies and use-cases
- As we can’t redesign the OS from the ground up w/ security in mind, retrofit & refinement is the only option; so we *need* to work with that



Challenges

- Multiple security models hindering adoption
- Convincing people of the value of security:
  - enable features
  - report problems
  - help improve usability

- ISV & user adoption: need to support multiple security models; no standard API (google chrome folk compare Apple dev vs. Linux...); not impossible, but extremely difficult (need to design flexibility into each layer, e.g. svirt, xace; then develop abstracted API....)
- Security is useless if nobody enables it...
- If people have problems, reporting the problems allows us to solve them!
- Core issue is getting people to understand the need for security (cf. Seat belts, bike helmets) and to then participate in the development cycle
- Some people still disable DAC by doing everything as root; we probably don’t have much hope of convincing them that MAC is worthwhile, but for the general user base, we can do a great deal, and we can also do a lot for specialist security users at the same time with the same codebase.



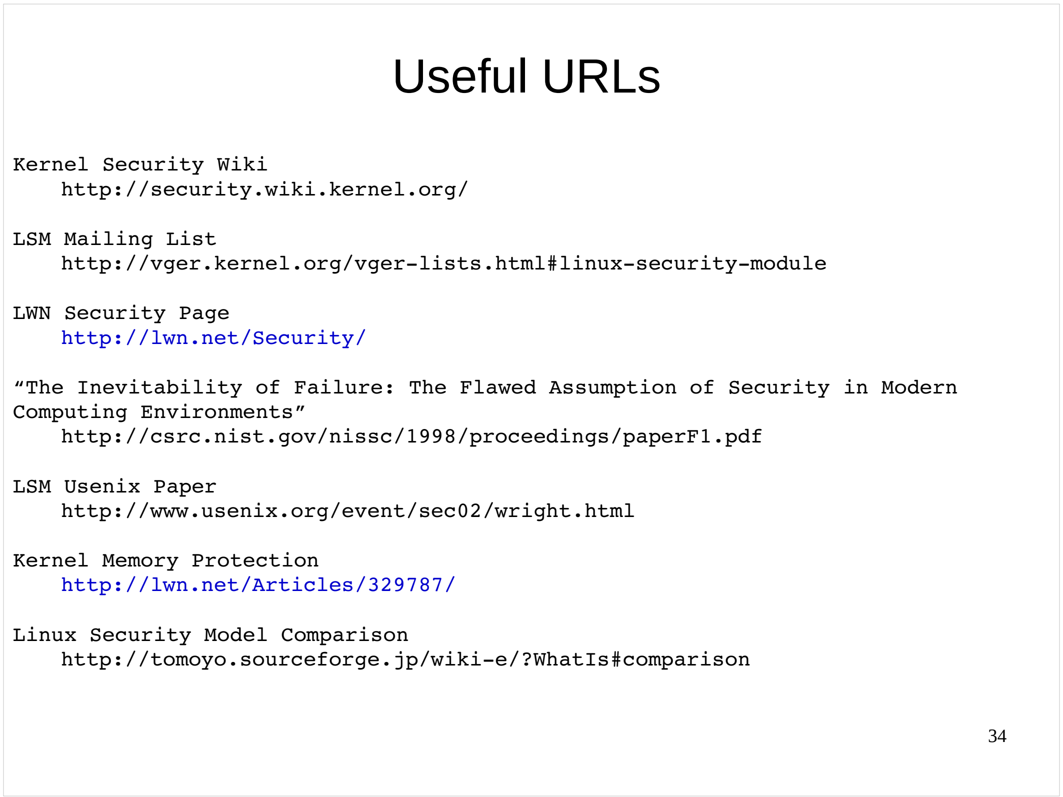
Resources

- Linux Kernel Security Wiki
- LSM Mailing List
- LWN Security page

- These are the most important links; everything else can be found from these
- LSM list is for general kernel security development discussion
- Also, a list of URLs is given at the end of the slides

Questions?

- Thanks to Stephen Smalley, Paul Moore and Dan Walsh for feedback on these slides



Useful URLs

Kernel Security Wiki
    http://security.wiki.kernel.org/

LSM Mailing List
    http://vger.kernel.org/vger-lists.html#linux-security-module

LWN Security Page
    http://lwn.net/Security/

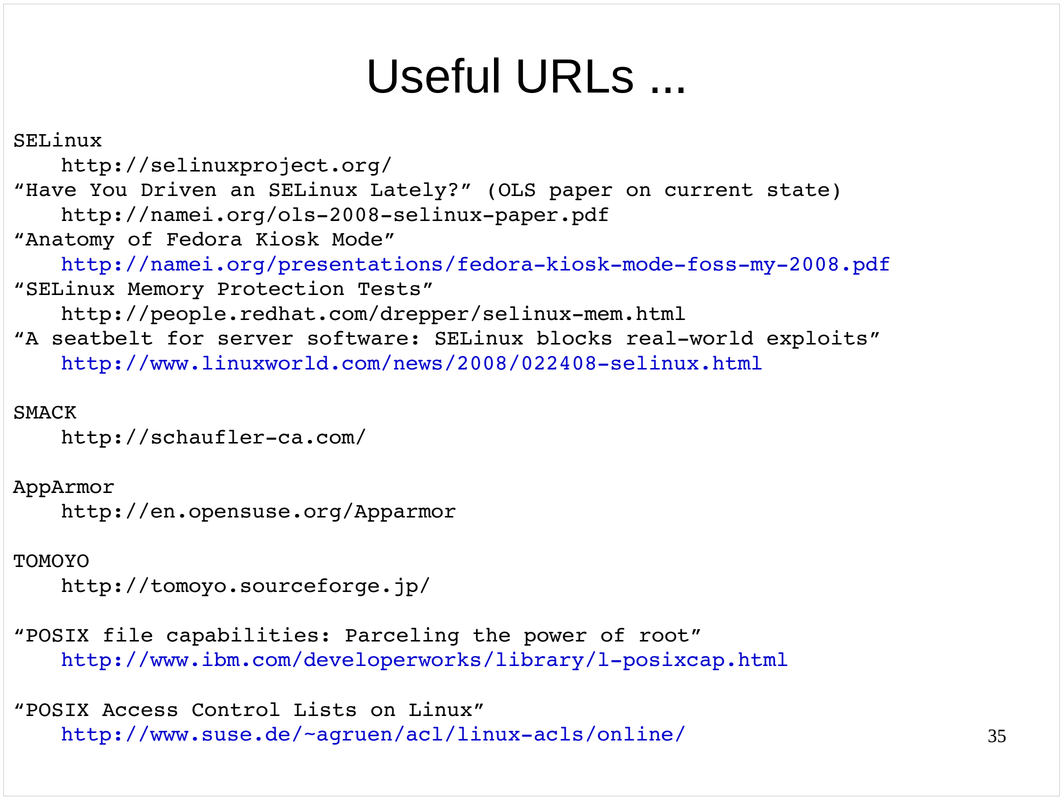
“The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments”
    http://csrc.nist.gov/nissc/1998/proceedings/paperF1.pdf

LSM Usenix Paper
    http://www.usenix.org/event/sec02/wright.html

Kernel Memory Protection
    http://lwn.net/Articles/329787/

Linux Security Model Comparison
    http://tomoyo.sourceforge.jp/wiki-e/?WhatIs#comparison

Note: this is not for display purposes really, but for people to use locally when looking for links.



Useful URLs ...

SELinux
    http://selinuxproject.org/
“Have You Driven an SELinux Lately?” (OLS paper on current state)
    http://namei.org/ols-2008-selinux-paper.pdf
“Anatomy of Fedora Kiosk Mode”
    http://namei.org/presentations/fedora-kiosk-mode-foss-my-2008.pdf
“SELinux Memory Protection Tests”
    http://people.redhat.com/drepper/selinux-mem.html
“A seatbelt for server software: SELinux blocks real-world exploits”
    http://www.linuxworld.com/news/2008/022408-selinux.html

SMACK
    http://schaufler-ca.com/

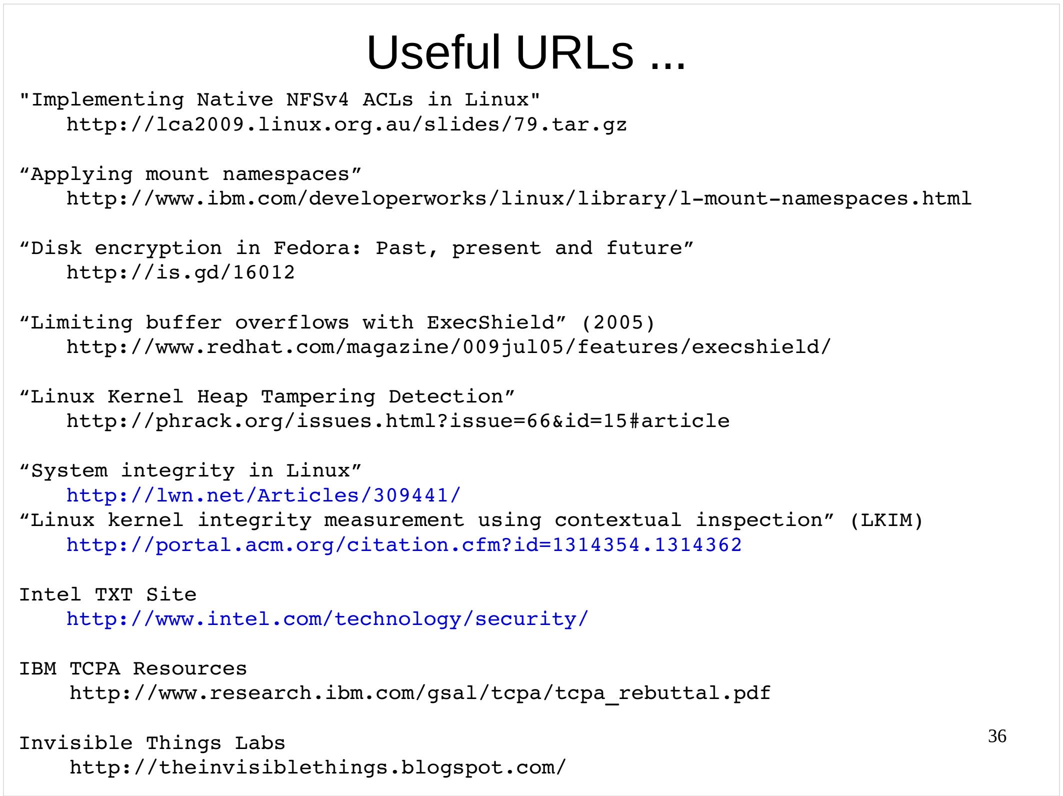
AppArmor
    http://en.opensuse.org/Apparmor

TOMOYO
    http://tomoyo.sourceforge.jp/

“POSIX file capabilities: Parceling the power of root”
    http://www.ibm.com/developerworks/library/l-posixcap.html

“POSIX Access Control Lists on Linux”
    http://www.suse.de/~agruen/acl/linux-acls/online/

Note: this is not for display purposes really, but for people to use locally when looking for links.



Useful URLs ...

"Implementing Native NFSv4 ACLs in Linux"
http://lca2009.linux.org.au/slides/79.tar.gz

“Applying mount namespaces”
http://www.ibm.com/developerworks/linux/library/l-mount-namespaces.html

“Disk encryption in Fedora: Past, present and future”
http://is.gd/16012

“Limiting buffer overflows with ExecShield” (2005)
http://www.redhat.com/magazine/009jul05/features/execshield/

“Linux Kernel Heap Tampering Detection”
http://phrack.org/issues.html?issue=66&id=15#article

“System integrity in Linux”
http://lwn.net/Articles/309441/
“Linux kernel integrity measurement using contextual inspection” (LKIM)
http://portal.acm.org/citation.cfm?id=1314354.1314362

Intel TXT Site
http://www.intel.com/technology/security/

IBM TCPA Resources
http://www.research.ibm.com/gsal/tcpa/tcpa_rebuttal.pdf

Invisible Things Labs
http://theinvisiblethings.blogspot.com/

Note: this is not for display purposes really, but for people to use locally when looking for links.