Retrospective: 26 Years of Flexible MAC

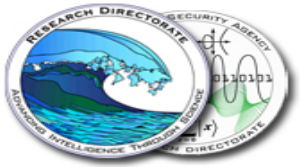
Stephen Smalley

Trust Mechanisms

Laboratory for Advanced Cybersecurity Research

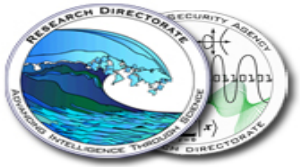
National Security Agency

August 20, 2019



Did you know…

- > 90% of Android devices today are using a security framework developed by NSA
  - and so are many Linux systems
- All iOS devices are using a security framework whose development was sponsored by NSA
  - and so are macOS and FreeBSD systems
- How did we get there?



About Us

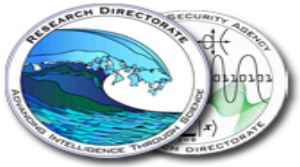
- R&D in support of NSA's mission to protect National Security Information and Information Systems.
- Celebrating our 30th anniversary next year.
- First at NSA to create and release open source software (SELinux, Dec 22 2000).
- Long history of open source contribution and collaboration.
  - Linux, Xen, FreeBSD, Darwin, OpenSolaris, Android, Zephyr, Fuchsia, ...

MAC: Background & Motivation

- Mandatory as opposed to discretionary access control (MAC vs DAC).
- Can address the “confinement problem”(Boebert & Kain 1996, Lampson 1973).
- Bound the damage that can be caused by flawed or malicious applications
- Enforce system-wide security properties.
- Facilitate gaining assurance.

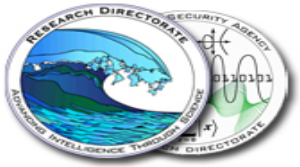
MAC: Desired Properties

- Enforces an admin*-defined security policy
  - Not subject to manipulation by the users or applications
  - (*) In some cases, not even the admin can change the policy.
- Over all subjects, objects, and operations
  - No subject, object, or operation is exempted or ignored
- Based on security labels associated with the subject and object(s) involved
  - Security equivalence classes for subjects and objects



Traditional MAC

- Fixed security policy models derived from government clearance/classification system.
  - confidentiality: Bell-LaPadula (BLP) model of Multi-Level Security (MLS)
  - integrity: Biba model (dual of BLP)
- Security labels specific to the fixed models.
- Historically limited to separate “trusted” operating systems.
  - e.g. STOP OS, Trusted Solaris, Trusted IRIX, Trusted AIX, Trusted HP-UX, …

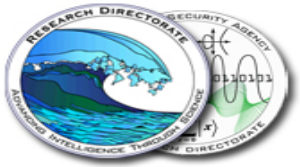


Traditional MAC: Gaps

- Required “trusted” subjects that could override the security model.
- Lacked a mechanism to effectively confine and protect the trusted subjects.
- Could not express many kinds of real world security goals (e.g. least privilege, assured pipelines, static/dynamic separation of duty).
- Did not take the executing program into account.

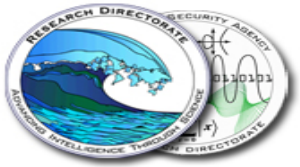
Type Enforcement (TE)

- Originally introduced to address failings of the hierarchical integrity models (Boebert and Kain, 1985).
- Particularly to support assured pipelines.
  - Ensure that one or more security-relevant subsystems are unbypassable and tamperproof.
- Capable of enforcing confidentiality, integrity, least privilege, separation of duty, …
- First implemented in the Logical Coprocessing Kernel (LOCK) system.
  - W. Boebert, The LOCK Demonstration, 11th NCSC vol 2, Oct 1988, pages 71-88.
  - O.S. Saydjari, LOCK: An Historical Perspective, 18th ACSAC, Dec 2002.



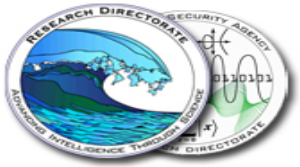
TE vs Traditional MAC

- Take program into account
  - Bind trust/permissions to programs, distinguish execute vs read access
- Separate labeling from policy
  - Label describes what subject/object is
  - Policy defines rules governing inter-label interactions - no fixed model
- Policy-driven granularity
  - Distinct permissions for each operation, grouped by policy writer as desired
- No all-powerful “trusted” subjects
  - Nothing is exempted from the security policy
  - Fine-grained granting of privilege on a per-(subject,object,operation) basis.



From TE to Flexible MAC

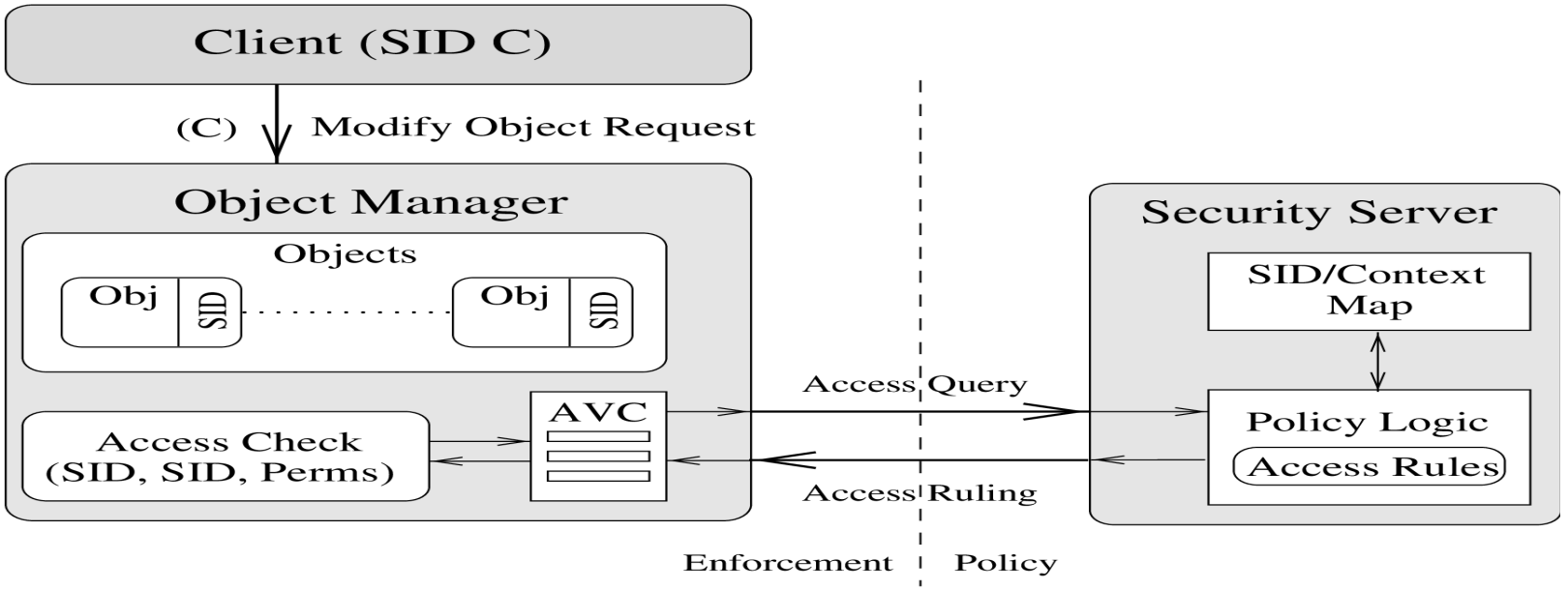
- Even TE has its limitations.
  - Need to support additional security models and future evolution.
- Generalized TE to create a flexible MAC architecture.
- Grew out of earlier work in LOCK and Trusted Mach (TMach).
- Prototyped in Mach 3 via the Distributed Trusted Mach (DTMach) and Distributed Trusted Operating System (DTOS) projects.
- Part of a broader NSA research program ("Synergy") to develop a distributed, microkernel-based OS security architecture.
- Motivated by potential security and assurance benefits of microkernel-based architectures and interest in Mach by industry.



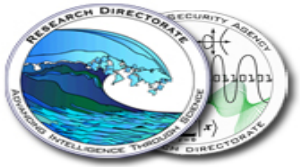
Flexible MAC meets Mach

- Deficiencies of Mach's capability model
  - Lack of control over capability propagation
  - Lack of granular control over capability usage
  - Lack of support for revocation
- DTMach/DTOS enhancements
  - Provide MAC policy control over propagation
  - Support fine-grained MAC policy control over capability use and service invocation
  - Reports/papers at http://www.cs.utah.edu/flux/dtos

Flexible MAC Architecture

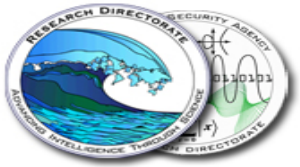


R. Spencer et al, The Flask Security Architecture: System Support for Diverse Security Policies, 8th USENIX Security, Aug 1999



DTOS Approach to MAC: Benefits

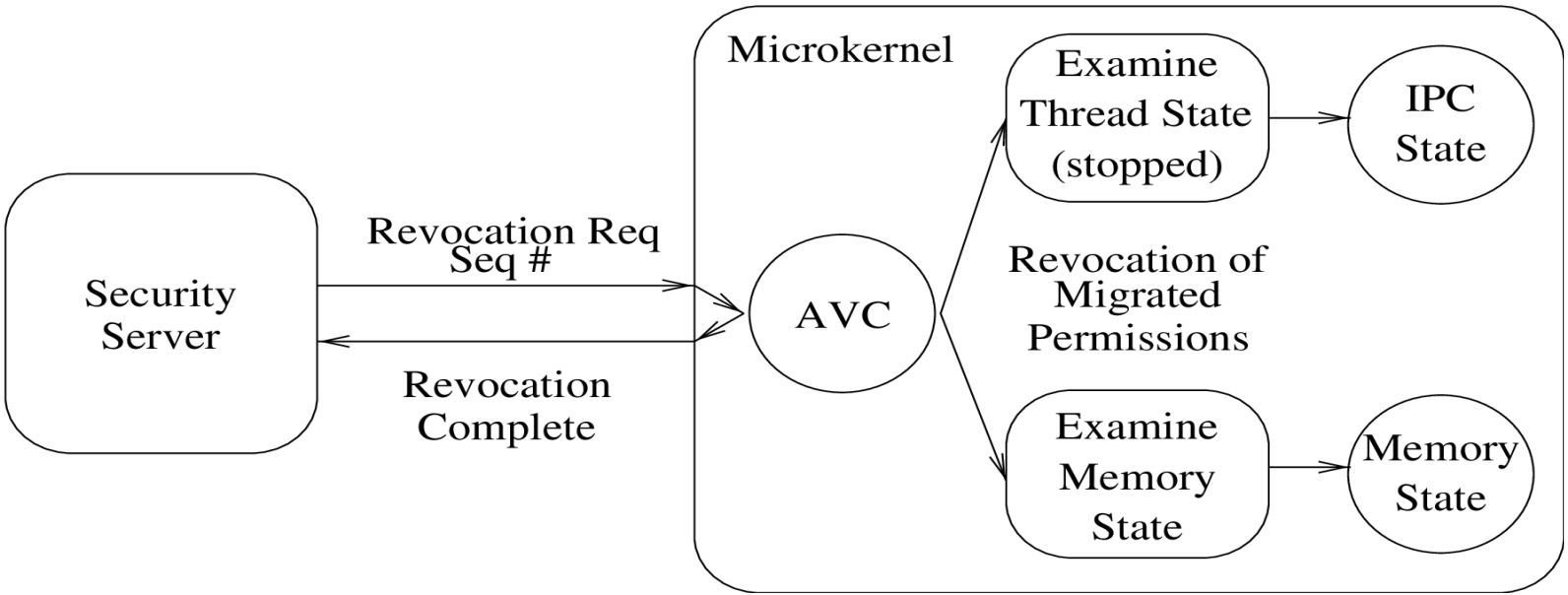
- Assurable implementation
  - Direct support for labeling and access control in microkernel
  - Capability leak by userspace component can be mitigated by microkernel checks
  - Reduced assurance burden on userspace components
  - Disaggregated TCB - userspace object managers, limited trust in each
- Centralized security policy
  - Amenable to analysis, audit, management
- Flexible support for security policies



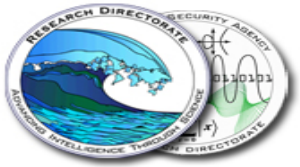
After Mach: Flux, Fluke and Flask

- Migrated to University of Utah's Flux project and Fluke microkernel.
- Called for renewed work in OS security.
  - P. Loscocco et al, The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments, 21st NISSC, Oct 1998.
- Produced the Flask security architecture.
  - R. Spencer et al, The Flask Security Architecture: System Support for Diverse Security Policies, 8th USENIX Security Symp, Aug 1999.
  - http://www.cs.utah.edu/flux/flask

Flask & Dynamic Security Policies



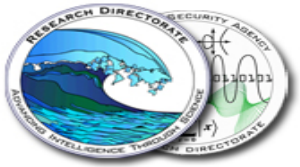
R. Spencer et al, The Flask Security Architecture: System Support for Diverse Security Policies, 8th USENIX Security, Aug 1999



Going Mainstream: SELinux

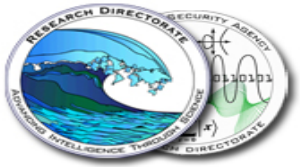
- Motivation:
  - Demonstrate viability of Flask architecture in a real (i.e. non-research) OS.
  - Provide an open source reference implementation.
  - Provide a long-term research platform for our own R&D.
- Linux chosen as the foundation based on:
  - large developer community, growing adoption, open development model
- First SELinux public release, December 2000

  P. Loscocco and S. Smalley, Integrating Flexible Support for Security Policies into the Linux Operating System, FREENIX '01, June 2001.



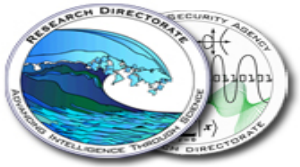
SELinux: Flask in a Monolithic Kernel

- Adapted first for a monolithic kernel architecture
  - Security server moved from userspace to kernel
- Then adapted again to gain adoption into mainline Linux
  - Worked with Linux community to create Linux Security Modules (LSM) framework, adapted SELinux to LSM
  - C. Wright et al, Linux Security Modules: General Security Support for the Linux Kernel, 11th USENIX Security, Aug 2002
  - SELinux merged into Linux 2.6.0-test3, August 2003



SELinux: Flask in Linux Userspace

- Security server API, userspace AVC
- D-BUS, Xorg, PostgreSQL, NSCD, …
- Supports consistent access control for the entire system
  - E. Walsh, Application of the Flask Architecture to the X Window System Server , 3rd SELinux Symposium, March 2007.
  - K. Kohei, LAPP/SELinux: A secure web-application stack using SE-PostgreSQL, PGCon 2010, May 2010.

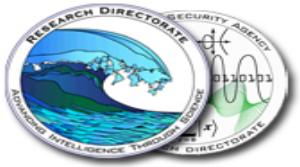


Branching Out: SE BSD and SE Darwin

- Motivation:
  - Demonstrate applicability to other OSes
  - Foster wider adoption
- Sponsored MAC for FreeBSD and Darwin.
- MAC framework adopted into FreeBSD, macOS, and iOS.
  - C. Vance et al, Security-Enhanced Darwin: Porting SELinux to Mac OS X, 3rd SELinux Symposium, 2007
  - R. Watson, New approaches to operating system security extensibility, Technical Report UCAM-CL-TR-818, April 2012
  - R. Watson, A decade of OS access-control extensibility, Queue 11, 1, January 2013.

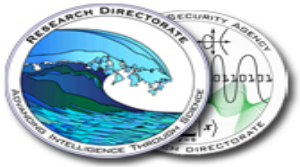
Going Mobile: SE for Android

- Motivation:
  - Increasing demand to use mobile devices in government
  - Growing interest in Android as an open platform with broad market adoption
  - Need for improved security in mobile OSes
- Explored MAC at two layers:
  - Kernel: Adapting SELinux to unique needs of mobile OSes
  - Middleware: Research into novel forms of MAC for Android middleware
- S. Smalley and R. Craig,
  Security Enhanced (SE) Android: Bringing Flexible MAC to Android, NDSS '13, Feb '13.
- http://selinuxproject.org/page/SEAndroid



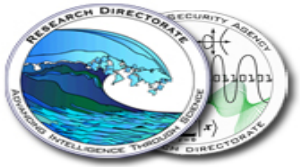
Android: MAC applied

- An exemplar of how to apply MAC
  - 100% confined+enforcing, well-defined security goals, automated validation & testing
  - TCB protection, attack surface reduction, decomposition / separation, app isolation
- The largest install base of any MAC system ever
  - Currently ~90% of > 2 billion active Android devices (5.0 or later)
- Now also coming to ChromeOS
  - Initially just the Android container, but WIP for all of ChromeOS



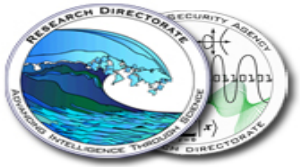
MAC meets Virtualization

- NetTop
  - VMWare/SELinux hybrid to support multiple security level connectivity from a single desktop; early pre-release user of SELinux
  - Similar approach later commercially realized in Red Hat's secure virtualization and container model (sVirt, OpenShift)
- Secure Virtual Platform (SVP)
  - Explored virtualization and trusted computing to address residual risks
  - Applied these technologies to construct a secure system architecture
  - Influenced High Assurance Platform (HAP), SecureView, OpenXT, Secure Mobility



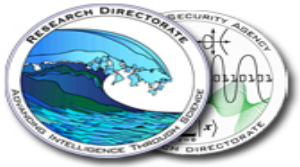
Hypervisors: Microkernels Revisited

- Opportunity to revisit microkernel-like architecture for security
  - Isolate untrusted and security-critical components
  - Enforce assured pipelines, e.g. inline VPN or DAR
- Xen chosen as research platform
  - viable open source hypervisor, community, adoption
- Making Xen suitable for security
  - Hypervisor Mandatory Access Control (Xen Security Modules / Flask)
  - Secure inter-VM communication (IVC / libvchan / v4v / Argo)
  - Decomposing domain-0
  - Support for lightweight domains/VMs



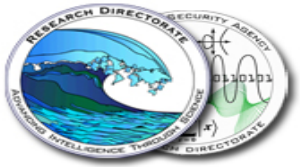
SVP: MAC in a system architecture

- Discrete components enforce MAC focused at proper abstraction level
  - Components leverage MAC from others
  - Union embodies system MAC policy and helps guarantee important security properties
- Hypervisor, OS, and userspace object managers collectively enforce system MAC policy
  - B. Hicks et al, An architecture for enforcing end-to-end access control over web applications, SACMAT '10, June 2010.
  - P. Alexander et al, Model Checking Distributed Mandatory Access Control Policies, ACM TISEC 18, 2, Dec 2015.



MAC: Then vs Now

- Mainstream OSes with MAC in 2000 (before SELinux):
  - None
- Mainstream OSes with MAC today:
  - Flexible MAC: Linux, Xen, FreeBSD, Android, ChromeOS, macOS, iOS
  - Traditional MAC: Solaris (TX), Windows (MIC)

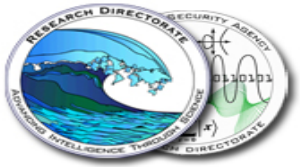


MAC for Emerging OSes

- Zephyr, https://www.zephyrproject.org/
  - Adapting MAC for highly resource-constrained devices
  - Preserving real-time guarantees under MAC
- Fuchsia, https://fuchsia.dev/
  - Revisiting MAC & capability-based microkernels
  - Blending with Secure Virtual Platform
    - hypervisor functionality, running Linux in VM

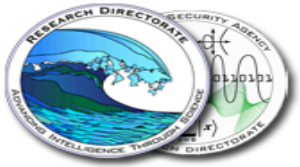
MAC Futures

- Advancing usability without sacrificing security
- Composing MAC models
- Distributed MAC enforcement and management
- Multi-party MAC



MAC: Call to Action

- MAC is now well established for OS hardening & app/container/VM isolation, but…
- It isn't being effectively used by higher level services or emerging technologies.
- Absent a change in this regard, failure remains inevitable.
- Leverage it!



Questions?

- Contact me: sds@tycho.nsa.gov
- SELinux Project:
  - https://www.nsa.gov/what-we-do/research/selinux/
  - GitHub: https://github.com/SELinuxProject
  - Developers' mailing list: selinux@vger.kernel.org
    - Subscribe by sending 'subscribe selinux' to majordomo@vger.kernel.org