Security-Enhanced Linux (SELinux) Case Study

John Bush and Heather Romero

INF-527

University of Southern California

October, 2015

What is SELinux?

- Started as research project in the 1990's by the NSA and Secure Computing Corporation (now owned by McAfee)
  - Result of prior research in the 1980s in high-assurance operating systems.
  - Problem: Realization software is inherently flawed (The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments")
  - Goal: To provide a more secure underlying operating system be implementing Mandatory Access Controls.
  - "Flask Architecture" renamed to SELinux and released to the public under the GNU GPL in 2000.
- SELinux has been added to various Linux distributions (Fedora, OpenSuse, Ubuntu) to encourage open-source development and adoption.
- Its architecture strives to separate enforcement of security decisions from the security policy itself and streamlines the volume of software charged with security policy enforcement.

SELinux Overview

- Replaces user-based model with a policy-based model
  - All actions reading and writing data are controlled by a security policy
- Separates the applications and processes executing on the system (applications are provided own view of resources through namespaces)
  - Isolates attack
  - Limits the damage of compromised software
- Original NSA policy was known as strict policy
  - Followed whitelist concept: default was to deny applications access unless specifically allowed
    - Requires maintenance to keep list updated
    - Works well in strict regulated environments, but does not work well on regular desktops
- To improve on strict policy, targeted policy was introduced (Fedora Core 3)
  - List of Deny statements
  - Allowed all actions given by a user except the targeted list
    - Protected critical applications, network processes

Design Objectives

- Intended to demonstrate the ability to add MAC to Linux
- Three types of Security Models:
  - Type Enforcement (TE)
    - SELinux uses type enforcement to constrain individual processes (subjects) to defined rules, rather than run at the permissions of of the standard Linux user level who called them.
    - Ex: a root user calls a text editor, that editor now runs at root privileges, the same as the user.
  - Role-Based Access Control (RBAC)
    - Each user gets a set of roles
    - Each role is assigned a set of TE domains
  - Traditional Multi-Level Security (MLS)
    - the Bell-LaPadula model (clearances, classifications, and categories)

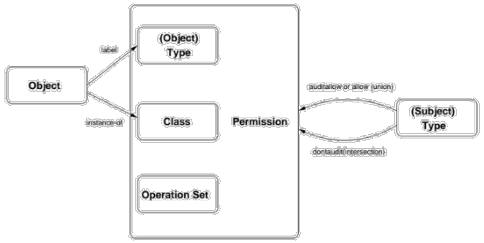
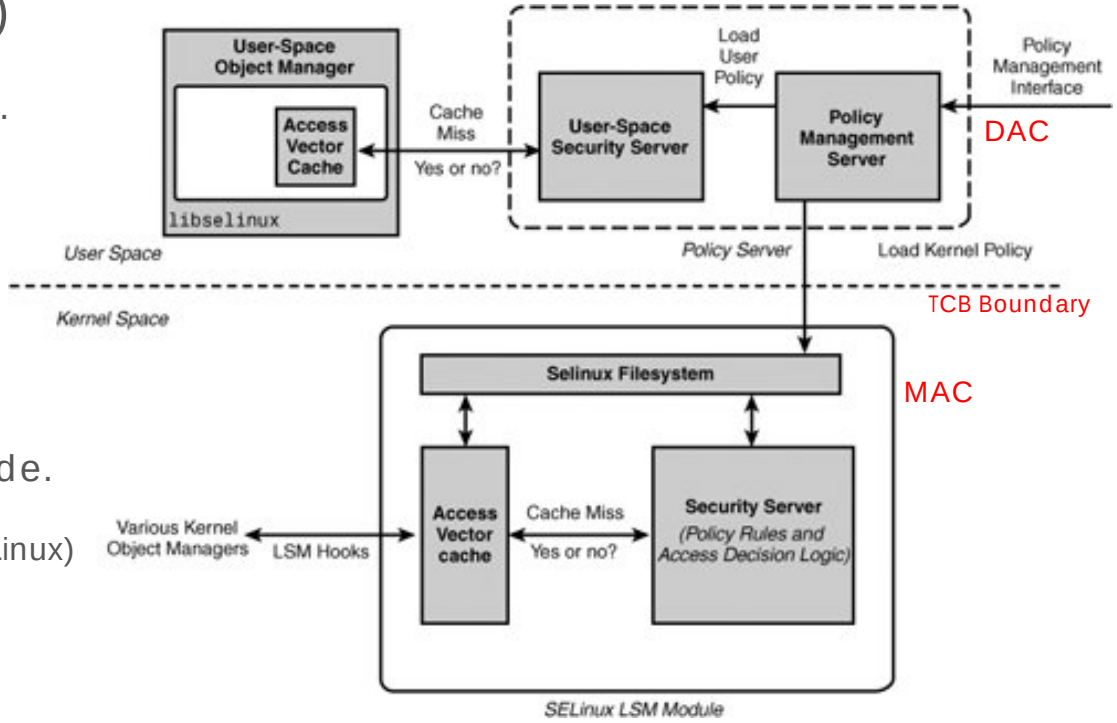


Figure 1: SELinux extended Type Enforcement (TE) policy model basics.

SELinux Architecture

- Security-Enhanced Linux (SELinux) is a Linux kernel security module (not its own operating system) that provides a mechanism for supporting access control security policies, specifically mandatory access controls (MAC).

- Linux Security Modules (LSM)
  - LSMs are additional frameworks added to the Linux security kernel.
    - Four official LSM's exist:
      - SELinux
      - AppArmor
      - Smack
      - TOMOYO
    - Provide "hooks" which are system interrupts that occur after an access request is made.
      - Directs the access request to the configured module (SELinux)
      - Occurs after DAC



Access-Request Architecture

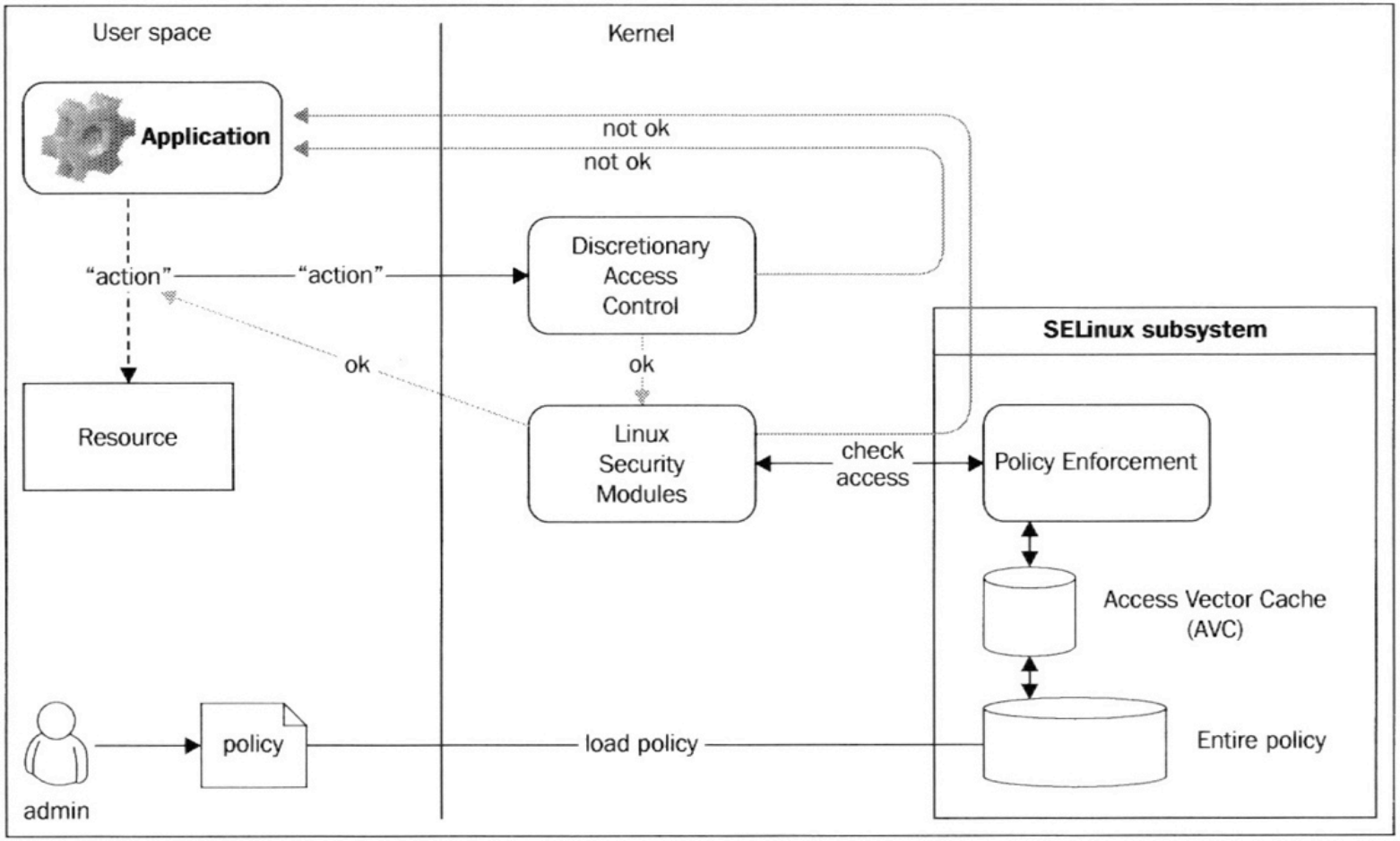


Image from "SELinux Cookbook" by Sven Vermeulen

Policy

- Policy is the set of rules for accessing data/processes
  - Types are defined for data objects
  - Domains are defined for processes
- The policy uses roles to limit the domains that can be entered and user identities to specify the roles that can be attained
- Policy Changes
  - Adding users
  - Adding permissions
  - Adding programs to an existing domain
  - Creating a new domain
  - Creating a new type
  - Creating a new role

Policy Module Example

- Source-code for policy file is: WiresharkCapture.te

```
policy_module(WiresharkCapture, 1.0.0)                    <header and policy name>

########################################
#
# Declarations
#

Type wireshark_t;                                         <“types” are declared>
type network_eth0_config_t;

########################################
#
# Local policy
#

allow wireshark_t network_eth0_config_t:file { read write getattr };
                                                          <allow rule between types>
                                                          <defines modes of access>
```

- Compiled policy module is: WiresharkCapture.pp

SELinux Modes

- **Enforcing**
  - SELinux policy is enforced. SELinux denies access based on SELinux policy rules.
- **Permissive**
  - SELinux policy is not enforced. SELinux does not deny access, but denials are logged for actions that would have been denied if running in enforcing mode.
- **Disabled**
  - SELinux is disabled. Only Linux system DAC rules are used.

Access Control in Linux vs SELinux

- **Standard Linux**
  - Is DAC only
  - Subjects contain a user/group ID.
    - ex: user “Schell” is a member of the “Faculty” group
  - Objects contain a similar user/group ID.
    - Ex: file “grades.txt” contains:
      - Owner = Schell
      - Group = Faculty
      - Others = specifics “everyone else”
    - Schell owns the file, but anyone in the faculty Group has some form of defined access.
  - Access is a combination of:
    - Read, Write, and/or Execute

- **SELinux**
  - DAC is always checked first.
    - If DAC access is disallowed SELinux is not referenced.
  - Access decided through Type Enforcement
    - All subjects and objects have an associated “security context”.
      - Is essentially a label:
      - user:role:type:level
      - Note: generally only “type” is compared.
    - The subject’s security context is compared against the object’s security contact.
      - “Is this subject type allowed to access this object type?”
  - Access granted only if both DAC is allowed and appropriate Type Enforcement exists.

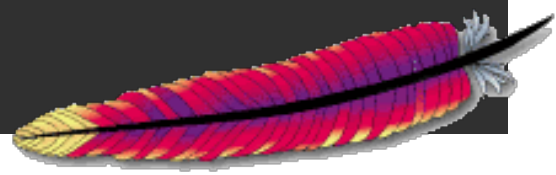
Label Comparison: TE vs BLP

- SELinux Security Context:
  - Users
    - Is a collection of Roles
    - Ex: faculty_r, staff_r, professor_r
  - Roles
    - Similar to a Unix group ID
  - Types
    - Similar to domains
    - Subjects/Objects with the same type are in the same domain.
    - Rules in policy files allow cross-domain (type) access
  - Levels
    - Consists of a sensitively level and category (access class)
    - Ex: level = s0:c0.c2, c4
      - s0 = sensitively level
        - Is hierarchical
      - c0.c2 = categories 0,1,2
      - c4 = and category 4
    - Alias names can be assigned to categories.
      - Are non-hierarchical

- Bell-LaPadula:
  - Clearances
    - Subject = unclassified, secret, topsecret
  - Classifications
    - Object = unclassified, secret, top-secret
  - Categories
    - Further constrains access by a group.
    - Ex: NUCLEAR, EUROPE, MISSLE
  - Access
    - *-Property: No writing down
    - SSC: No reads up

Note: again, in SELinux only Type(s) are generally compared unless in MLS enforcement.

Type Enforcement Example



- Apache Web-server:
  - Linux commands for viewing security context:
    - ls –Z filename — displays security context of a file
    - ps –Z processname — displays security context of a process
    - id –Z username — displays security context of a user
  - Apache Webserver
    - The Apache server process is of type: httpd_t
    - A configuration file for Apache is of the type: httpd_config_t
    - Question: Should security context allow access between process/file?
  - /etc/shadow file
    - The local shadow file (storing hashed passwords) is of type: shadow_t
    - Question: Should security context allow access between Apache/shadow?
  - Discussion:
    - Type enforcement = comparison of a subject type (domain) to an object type (domain).
    - Is this really Mandatory Access, or additional Discretionary Access?

SELinux Security

- Administratively defined and not set at user discretion
- Clean separation of policy from enforcement (well-defined policy interfaces)
  - Policy rules define how processes interact with data and other processes
- Access control across all types of users and groups
- Lower vulnerability to privilege escalation attacks
  - attacker can only gain access to data and processes allowed by the normal policy
- Can be used to enforce data confidentiality and integrity
  - Type enforcement = integrity policy
  - MLS = Confidentiality policy

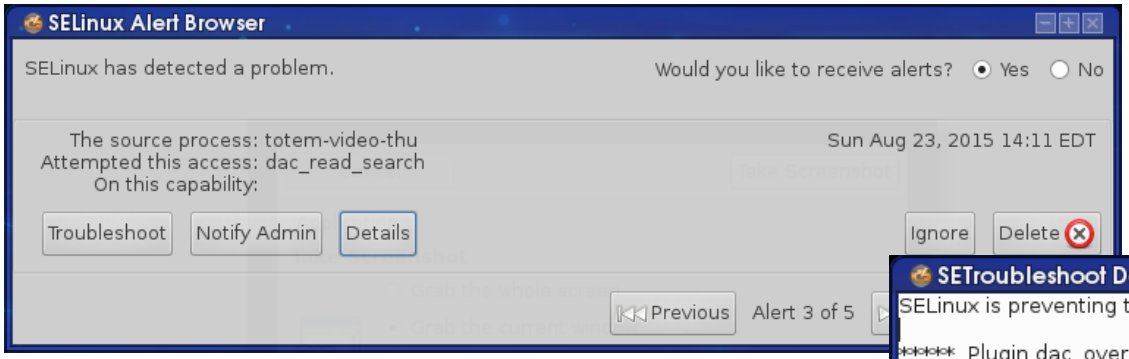
Properties of Secure Systems

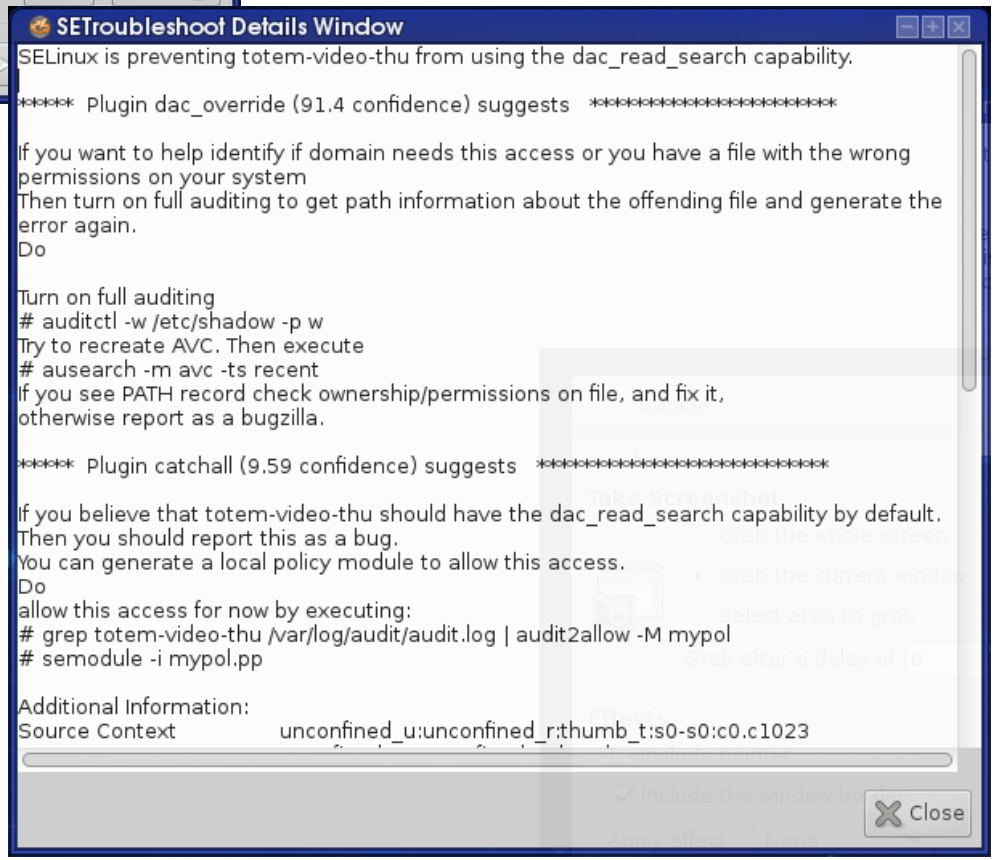
- Reference Monitor
  - Always invoked
    - the policy of the SELinux is always invoked to access data/processes
  - Tamper-proof
    - separation of applications/processes, MAC
  - Verifiable
    - Too complex
- TCB (code design)
  - Layering
  - Information hiding
  - Minimalization

Usability of SELinux

- Most well known MAC system in Linux operating systems
  - Red Hat and Fedora come with comprehensive policies
- Still not widely used in desktop environments
  - Typical configuration contains ~100,000 rules
    - Subject Type to Object Type access rights are “1 to 1” for each relationship (rule).
    - Creates a large number of access rights!!!
  - Requires a lot of maintenance – policy changes/fine-tuning
  - Creating a policy from scratch is very complex - requires expert knowledge of the OS and security policies

SELinux Alert Example





Alerts can be confusing and vague.

SELinux Alerts and Errors

- What is an Alert?
  - An action has occurred which violates your policy.
- RedHat Summit 2012: SELinux for Mere Mortals (Video)
  - Four types of common errors:
    - 1.) Incorrect labeling
      - Admin error of incorrectly labeling, or not labeling, types.
      - Use SELinux logs/alert messages to determine the access attemp.
      - As root re-label subjects/objects
    - 2.) Policy out-of-date:
      - Admin has made a system change and now policy is needs update.
      - Ex: installed new application which wasn't available at original policy creation.
    - 3.) Bug in the policy
      - The Linux distribution has an error in its default policy.
      - Fixed with a ticket to developer for a patch.
      - Not good!
    - 4.) System is under attack

Subversion

- Discussion: How would an adversary attack the system?
- Changes against the Policy:
  - Change/copy the policy file (/etc/selinux/…)
    - Or at least view it: Fedora 22: DAC policy allows global read access
    - Adversary could look for flaws in policy.
  - All access denied by default, but someone must have access to change policy
    - Loss of administrator (root) credentials (theft, brute-force, guess) results in system compromise.
    - Insider threat
  - Change the label (security context) of a subject/object
    - Or move an object off of the current SELinux environment!
    - Will no longer be protected.
- Audit2allow command
  - Designed by administrators to quickly and automatically set “allow” rules based on denial logs.
  - Human factor: annoying alerts are easily turned off, but is the allow fully understood?
- Risk of backdoors/malware
  - Each Linux distribution includes its own SELinux policy set (who verifies?)
  - Cannot verify all of the Linux source code
- Trusted distribution
  - Attack integrity of installation/source files
    - Was the distribution acquired through bit-torrent? Checksum verified?
  - Attack integrity of file/system updates
    - Open source development, inserting of malicious code?

System Comparison

- Conceptual difference between SELinux and GEMSOS
  - SELinux separates the policy from the enforcement mechanism
  - The mechanism consults the policy in order to enforce
  - Policy can be completely removed and replaced without changing the architecture.
  - Provides greater flexibility, but greater risk for subversion
- GEMSOS is a system designed around the policy
  - Focus is on designing a system and its mechanism as one.
  - Less flexibility, but less risk for subversion.
- Comparison against other Linux Security Modules (LSM).
  - AppArmor (http://apparmor.net)
  - Smack (http://schaufler-ca.com/)
  - TOMOYO (http://tomoyo.osdn.jp)

← Big differences vs SELinux:
-level of industry backing/development.
-types of labeling (object labels vs file-paths).
-definition of objects.
-support for MLS.

SELinux in a High Assurance Environment

TCSEC M-Component A1 Requirements

Requirement	SELinux	GEMSOS	Requirement	SELinux	GEMSOS
1 – OBJECT REUSE	No. Memory wiping not mentioned.	YES	14 – COVERT CHANNEL ANALYSIS	No.	YES
2 - LABLES	YES	YES	15 – TRUSTED FACILITY MANAGEMENT	No. Subversion threat!	N/A
3 – LABEL INTEGRITY	No. No mention of cryptographic bindings of labels to objects.	YES	16 – TRUSTED RECOVERY	No.	YES
4 – EXPORT. LABEL INFO.	Unknown.. Does TE information export? Does second system support same policy enforcement?	N/A. Single Level Only	17 – SECURITY TESTING	No. No documentation online as to SELinux testing results.. Assurance problem.	YES
5 – EXPORT. TO MULT-DEVICES	No. Not defined as of requirement #4.	N/A. Single Level Only	18 – DESIGN SPEC. VERIFICATION	No. No FTLS exists (not possible to map SELinux back to a FTLS as it runs on an untrusted Unix environment).	YES
6 – EXPORT TO SINGLE-DEVICES	No. Not defined as of requirement #4.	YES	19 – CONFIGURATION MANAGEMENT	No. There is no configuration management plan or guidance as to how the system is/should-be configured, only trust the distributer "got it right".	YES
7 – LABEL HUMAN READABLE	No. Labels are not included on any printed output.	N/A	20 – TRUSTED DISTRIBUTION	No. Subversion threat!	YES
8 – SUBJECT SENSIVITY LEVLES	No. Subject sensitively labels changed by an administrator. No notification practices are defined to administrator/user.	N/A	21 – SECURITY FEATURES USER GUIDE	Yes. - Fedora 22 SELinux User's and Administrator's Guide	YES
9 – DEVICE LABELS	No. Labels	YES	22 – TRUSTED FACILITY MANUAL	Kind of. Documentation on maintenance and configuration available online and in printed publications. Are all sources trusted?	YES
10 - MAC	YES	YES	23 – TEST DOCUMENTATION	No. No test documentation found online.. Assurance issue.	YES
11 – TRUSTED PATH	No.	YES	24 – DESIGN DOCUMENTATION	YES. Reasonable documentation on Flask and SELinux architecture.	YES
12 – SYSTEM ARCHITECTURE	No. The underlying TCB (hardware, firmware, Unix code) is not verified.	YES	25 - RAMP	Depends on the long-term support of the distributor.	YES
13 – SYSTEM INTEGRITY	No. Not possible to verify hardware or firmware in system designed to be multi-platform.	YES			

"*This work is not intended as a complete security solution. It is not an attempt to correct any flaws that may currently exist in an operating system. Instead, it is simply an example of how mandatory access controls that can confine the actions of any process…*" –NSA.gov https://www.nsa.gov/research/selinux/

Assurance

- SELinux Assurance Questions
  - How do you verify SELinux is configured and functioning properly?
  - How do you verify the trust of SELinux source code for correct implementation?
- System flexibility comes at an assurance cost.
  - How do you know you have CORRECTLY created a rule for every type of scenario?

Creating the rules is not difficult, "*..the challenge is determining the many thousands of accesses one must create to permit the system to work...*" -Book: "SELinux by Example"

Discussion

- SELinux is obviously not GEMSOS
  - Problems:
    - Does not meet the TESEC requirements for A-1
    - Is not verifiable. Low Assurance / High Subversion factor.
    - Requires expert technical knowledge for policy creation.
  - Question:
    - 1.) Should it be used? If so, by who?
      - Can a home internet user rely on SELinux for protection?
        - Fedora 22: User and applications run at the unconfined_u/r
      - Can a system admin protect his responsible network with SELInux?
        - Can your system admin build your corporate policy?
        - Trust on RedHat/Fedoras?
    - 2.) Does SELinux provide an enhancement to modern computer security?
      - Would Target/Sony/HomeDepot/OPM/…. breaches been prevented?
      - If your system is infected with a virus… what will it have access to?

Practical Example

- Virtual Machine Exercise
- Fedora 22 with SELinux installation
- Two objectives:
  - Setup type enforcement to prevent an application running at administrative privileges (root) from opening a file.
  - Create a MLS rule to provide confidentiality between three users and three files at different clearance/classification levels.
- Accomplished by:
  - Writing a basic level SELinux policy module for type enforcement.
  - Labeling objects.
  - Assigning clearances to subjects.
- File is located on Google Drive (4.5Gb)

Further Reading

- Dan Walsh of RedHat
  - Main developer for writing SELinux policy
  - Blog: http://danwalsh.livejournal.com
    - Contains examples of SELinux polices
    - Addresses SELinux vs multiple modern-day threats and exploits
- Russell Coker of RedHat
  - SELinux developer, author and presenter
  - Website: http://www.coker.com.au/selinux/
    - Contains past lecture material, SELinux resources, and a virtual machine test environment.
- RedHat - SELinux for Mere Mortals (video)
  - RedHat 2012/2013 Summits
  - Introduction to SELinux series by Thomas Cameron (Chief Architect SELinux Canada and USA)
  - 2013 Summit: https://www.youtube.com/watch?v=bQqX3RWn0Yw
  - 2012 summit: https://www.youtube.com/watch?v=MxjenQ31b70
- Eli Billauer
  - Freelance electrical engineer
  - Website: http://www.billauer.co.il/selinux-policy-module-howto.html
  - Best example of writing a simple policy module

References

- **Beijing Jiaotong University**: Zhai, Wu, “Automatic Analysis Method for SELinux Security Policy”. April 2012. http://www.tdeig.ch/SELinux/Publications/Analysis_Method.pdf
- **Beijing University of Technology**: Xu, Xiao, Chuangbai Xiao, Chaoqin Gao, and Guozhong Tian. "A Study on Confidentiality and Integrity Protection of SELinux." 2010 International Conference on Networking and Information Technology (2010): n. pag. Web. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5508513.
- **FedoraProject**: "SELinux User's and Administrator's Guide." Fedora 22. FedoraProject.org, 2014. Web. <https://docs.fedoraproject.org/en-US/Fedora/22/html/SELinux_Users_and_Administrators_Guide/index.html>.
- **IBM DeveloperWorks**: Ivashko, “Secure Linux: Part 1. SELinux – history of its development, architecture and operating principles.” May 2012. http://www.ibm.com/developerworks/library/l-secure-linux-ru/l-secure-linux-ru-pdf.pdf
- **NUCES**: Khan, Kashif, Muhammad Admin, Abbas Afridi, and Waqas Shehzad. "SELinux IN and OUT." IEEE Xplore. National University of Computer & Emerging Sciences (NUCES), May 2011. Web. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6014064.
- **NSA**: "Loscocco, Smalley, Muckelbauer, Taylor, Turner, and Farrell. "The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments." 21st National Information Systems Security Conference. The University of Utah, Oct. 1998. Web. <http://www.cs.utah.edu/flux/fluke/html/inevitability.htm>.
- **NSA**: Smalley, Vance, and Salamon. "Implementing SELinux as a Linux Security Module." Original NSA Publication (n.d.): n. pag. University Bologna. May 2002. Web. <http://www.cs.unibo.it/~sacerdot/doc/so/slm/selinux-module.pdf>
- **PennState**: Hanson, Chad. "SELinux and MLS: Putting the Pieces Together." PennState, n.d. Web. <http://www.cse.psu.edu/~trj1/cse543-f07/papers/SELinux-MLS.pdf>.

References

- **RedHat**: Ančincová, Barbora. "Security-Enhanced Linux User Guide." Red Hat Enterprise Linux 6 Security-Enhanced Linux. RedHat, n.d. Web. <https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Security-Enhanced_Linux/index.html>.
- **RedHat**: Cameron, Thomas. "2012 Red Hat Summit: SELinux For Mere Mortals." 2012 Red Hat Summit: SELinux For Mere Mortals. RedHat, n.d. Web. 2012. https://www.youtube.com/watch?v=MxjenQ31b70.
- **UNC Charlotte and Samsung**: Ahn, Gail-Joon, Wenjuan Xu, and Xinwen Zhang. "Systematic Policy Analysis for High-assurance Services in SELinux." 2008 IEEE Workshop on Policies for Distributed Systems and Networks. Sefcom, 2008. Web. http://sefcom.asu.edu/publications/systematic-policy-analysis-policy2008.pdf.
- **usenix.org**: Write, Chris, Crispin Cowan, James Morris, Stephen Smalley, and Greg Kroah-Hartman. "Linux Security Modules: General Security Support for the Linux Kernel." Usenix.org, 2002. Web. <https://www.usenix.org/legacy/events/sec02/full_papers/wright/wright.pdf>
- **SELinux Cookbook**: Vermeulen, Sven. SELinux Cookbook. Vol. 1. Birmingham, Mumbai: PACKT, 2014. Print.
- **SELinux By Example**: Mayer, Frank, Karl MacMillan, and David Caplan. SELinux by Example: Using Security Enhanced Linux. Upper Saddle River, NJ: Prentice Hall, 2007. Print.
- **The University of Utah**: Spencer, Ray, Stephen Smalley, Peter Losocco, Mike Hibler, David Andersen, and Jay Lepreau. "The Flask Security Architecture: System Support for Diverse Security Policies : Flux Research Group." The Flask Security Architecture: System Support for Diverse Security Policies. The University of Utah, 1999. Web. https://www.flux.utah.edu/paper/spencer-security99.
- **Wikipedia**: "Security-Enhanced Linux." Wikipedia. Wikimedia Foundation, n.d. Web. 26 Sept. 2015. Web. https://en.wikipedia.org/wiki/Security-Enhanced_Linux.