Linux Security Usability: Restricting Programs Using SELinux, AppArmor and FBAC-LSM

Z. Cliffe Schreuders

Linux Security Summit 2010

Murdoch UNIVERSITY

The Problem

- User-oriented controls typically assume processes act in the best interests of end users
- Vulnerable software
- Malware
- Rule-based application restrictions

Murdoch UNIVERSITY

Security and Usability

Murdoch UNIVERSITY

Usability Study

- SELinux
  - Studied in terms of application restrictions
- AppArmor
- FBAC-LSM
  - A new approach
  - Designed to be easy to use
  - Prototype LSM

Murdoch UNIVERSITY

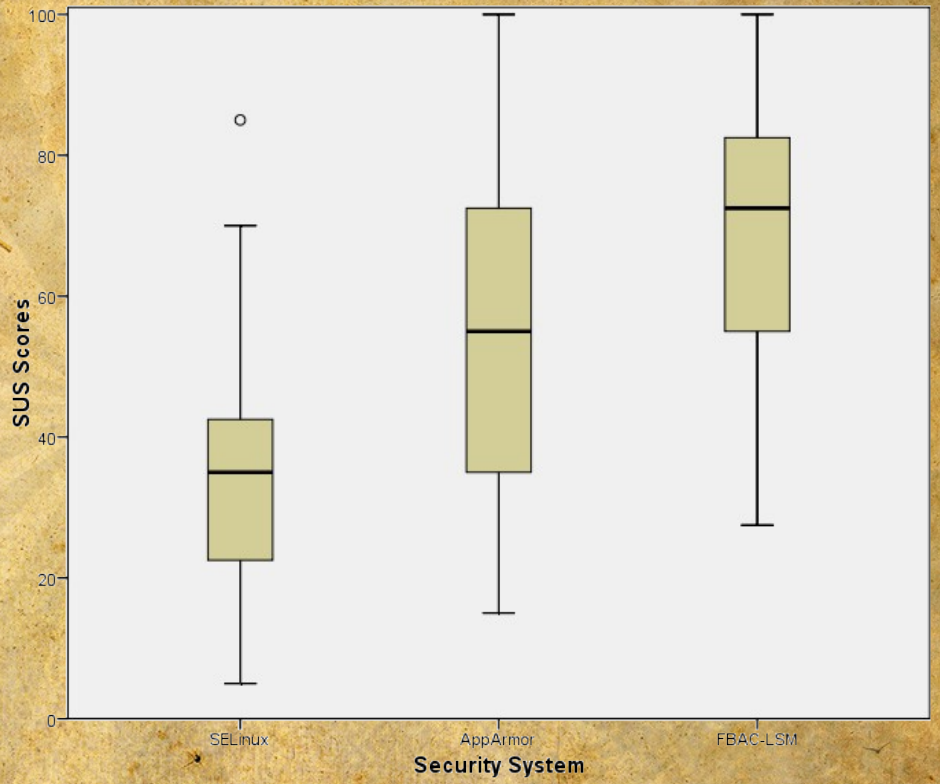
Method

- Within subjects
- 39 participants
- Created policies to confine two programs, one legitimate and one acting maliciously
- Data collected included usability feedback (the System Usability Scale, SUS) and a count of the threats which were not mitigated
- Pilot study

Murdoch UNIVERSITY

Results: Perceived Usability

- All three systems were significantly different from each other



Murdoch UNIVERSITY

Results: Creation of Policies

- All three systems were significantly different from each other

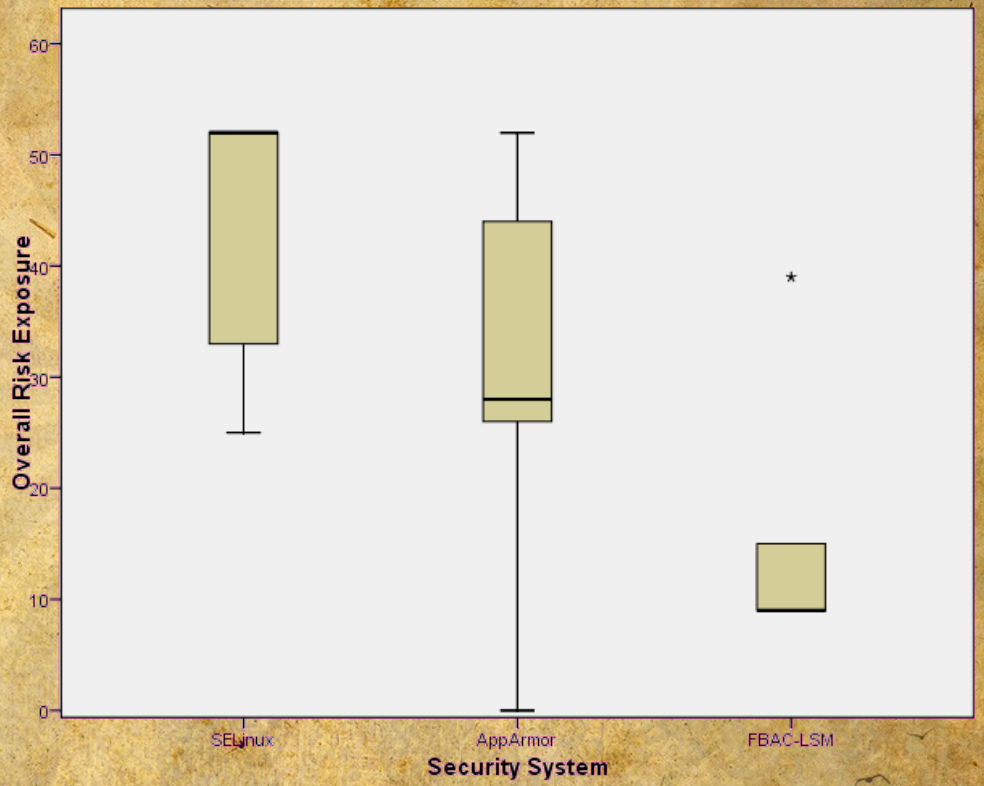
Security System	Application	No Policy (unconfined)	Unenforced Policy (unconfined)	Enforced Policy
SELinux	Opera (n=31)	21 (68%)	3 (10%)	7 (23%)
	KSirtet (n=9)	6 (67%)	1 (11%)	2 (22%)
AppArmor	Opera (n=38)	3 (8%)	10 (26%)	25 (66%)
	KSirtet (n=38)	7 (18%)	4 (11%)	27 (71%)
FBAC-LSM	Opera (n=39)	4 (10%)	0 (0%)	35 (90%)
	KSirtet (n=39)	7 (18%)	0 (0%)	32 (82%)

Policy creation rates for Opera and KSirtet using SELinux, AppArmor, and FBAC-LSM

Murdoch UNIVERSITY

Results: Overall Risk Exposure

- All three systems were significantly different from each other



Murdoch UNIVERSITY

Results: Risk Exposure for Opera, Trojan

- All three systems provided similar protection when the program behaved legitimately and policies were successfully created
- Significant difference in the Trojan horse risk exposure with FBAC-LSM (M=5.6, SD=4.9) having a significantly lower risk exposure than AppArmor (M=15.2, SD=9.9)

Murdoch UNIVERSITY

Constructive Criticism and Suggestions

- Based on participant feedback
  - Questionnaires
  - Debriefing sessions
  - Observations

Murdoch UNIVERSITY

Constructive Criticism: SELinux

- Interface
  - Command line tools
  - Output / policy hard to understand
- Expertise required / Complexity

Murdoch UNIVERSITY

Suggestions for Improving SELinux

- GUI tools should:
  - cover the whole process of creating, editing, compiling and enabling policies
  - facilitate the vetting of learned rules
  - provide information about the practical impact of rules
  - provide further documentation and hints

Murdoch UNIVERSITY

Suggestions for Improving SELinux

- Output from tools should be clearer
  - The policy language, AVCs, and tool feedback should be simplified
  - or other ways of viewing the information should be provided
- SETroubleshoot could take action to make the policy changes it suggests

Murdoch UNIVERSITY

SELinux Quirks and Flaws

- The polgengui tool
  - did not inform users that the created policies start in permissive mode
  - "too many values to unpack" when port numbers are specified
  - window did not fit on low resolution displays and could not be resized

Murdoch UNIVERSITY

SELinux Quirks and Flaws

- Default policies should provide actual confinement, the default policy for KSirtet did not
- AVC denial logs were sent to one of two separate log files (messages and audit.log)
- The failure to log relevant denials, as was the case with Opera, should be investigated

Murdoch UNIVERSITY

Constructive Criticism: AppArmor

- Expertise required to vet the learnt rules
- Too many decisions to make
- Interface

Murdoch UNIVERSITY

Suggestions for Improving AppArmor

- Clarify severity levels
  - Colour coding, clarify scale
- Provide more useful information about resources and executables
  - Purpose, risks, recommendations
- Help explaining on screen elements (eg “mrw”)

Murdoch UNIVERSITY

Suggestions for Improving AppArmor

- Navigate backwards
- Indication of progress
- Skip rules (to do later)
- Optionally, provide rules in list format
- Automatically suggest globing
- Enforcing state should be made more obvious
- Have predefined templates which can be used as a starting point to develop profiles

Murdoch UNIVERSITY

FBAC-LSM Usability Features

- Functionality-based
- High level policy abstractions
  - Parameterised to adapt to applications
- Automation
- A priori policy specification

Murdoch UNIVERSITY

FBAC-LSM

- Current steps to confine a program:
  - Name
  - Level of automation
  - Functionalities (suggestions automated)
  - Parameters (mostly automated)
  - Review
  - Apply

Murdoch UNIVERSITY

FBAC-LSM

- Lots of unique features
- Functional but unstable (kernel-side)
- Currently working on export to AppArmor
  - Mostly works!
- Please come and talk to me about it!

http://schreuders.org/FBAC-LSM

Murdoch UNIVERSITY

Conclusions

- There are a number of opportunities to improve the usability of SELinux and AppArmor
- A functionality-based approach can provide significant benefits to usability and security

http://schreuders.org/FBAC-LSM

Murdoch UNIVERSITY