Digital TV and application store, solving security problems

Vlad Buzov
Mentor Graphics Corporation
Embedded Systems Division

Embedded Linux Conference
October 16th, 2009
Grenoble, France

Participants

- CE Linux Forum

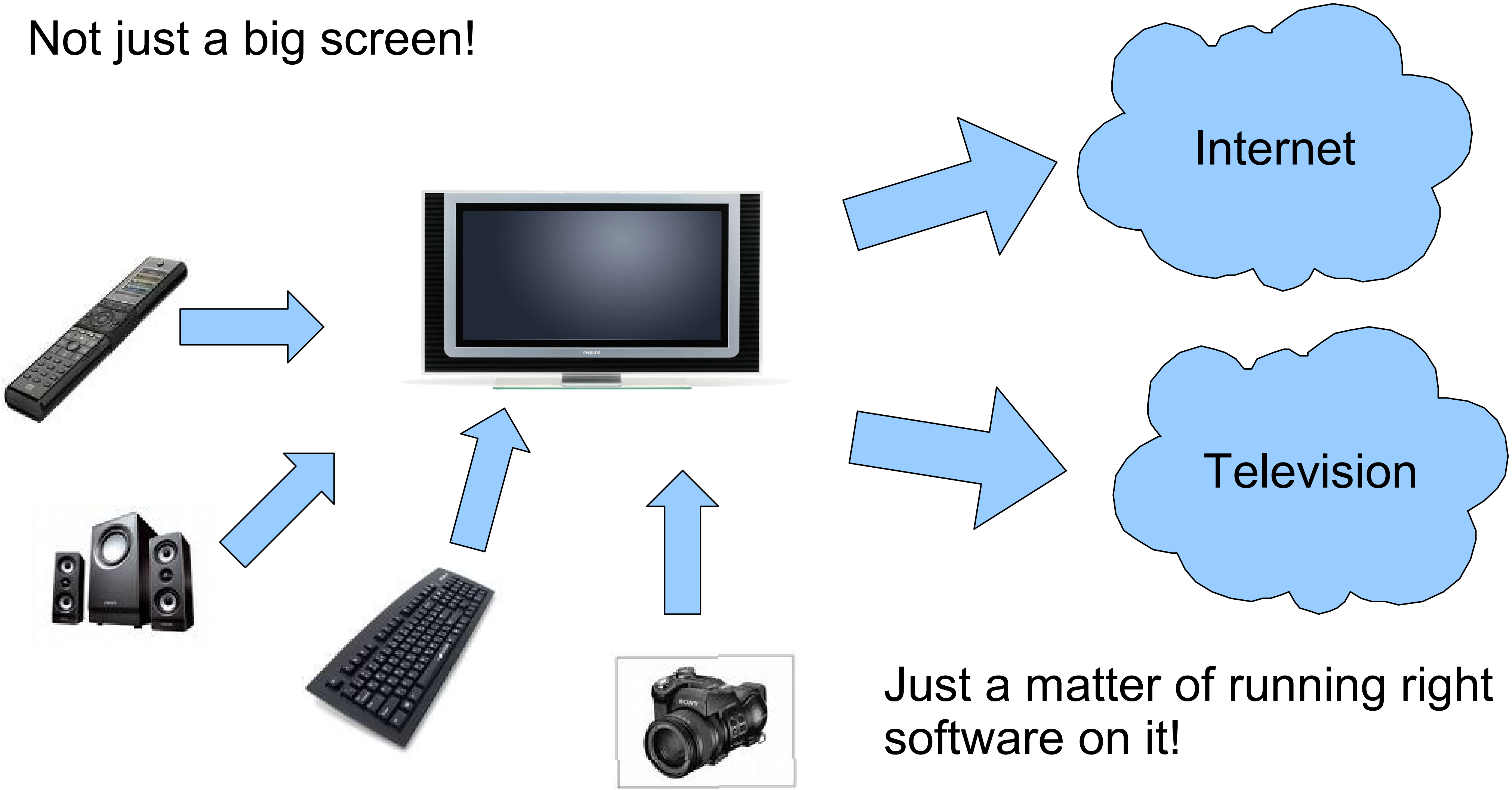
  http://celinuxforum.org

- Mentor Graphics Corporation, Embedded Systems Division

  http://www.mentor.com/products/embedded_software/

What's a Digital TV?

Not just a big screen!

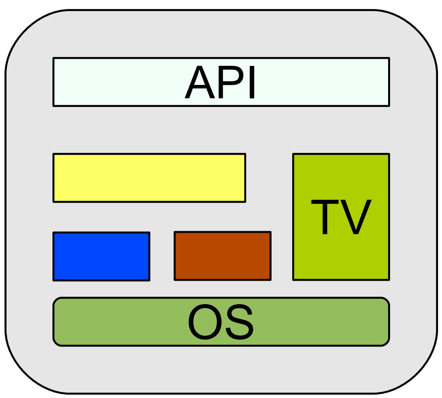


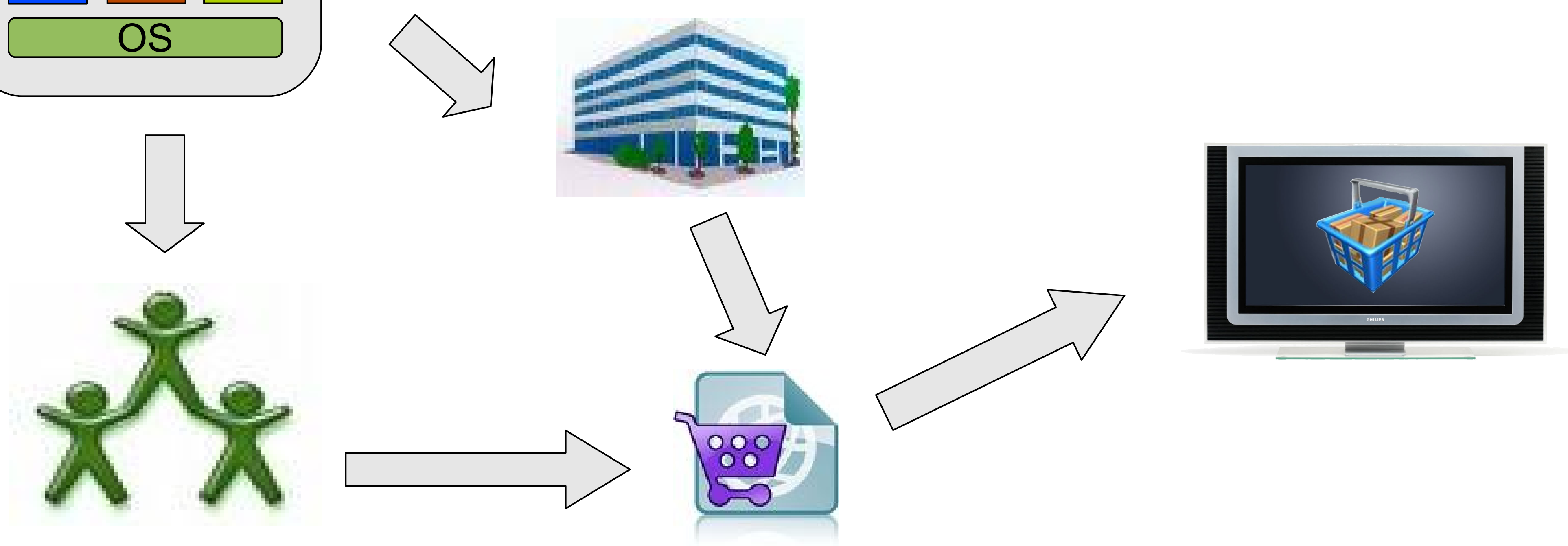
Just a matter of running right software on it!

Mentor Graphics EMBEDDED SYSTEMS

Let others do that!

- Create a development platform
  - Give it to other companies and people
    - Build application market
      - Let customers download the applications

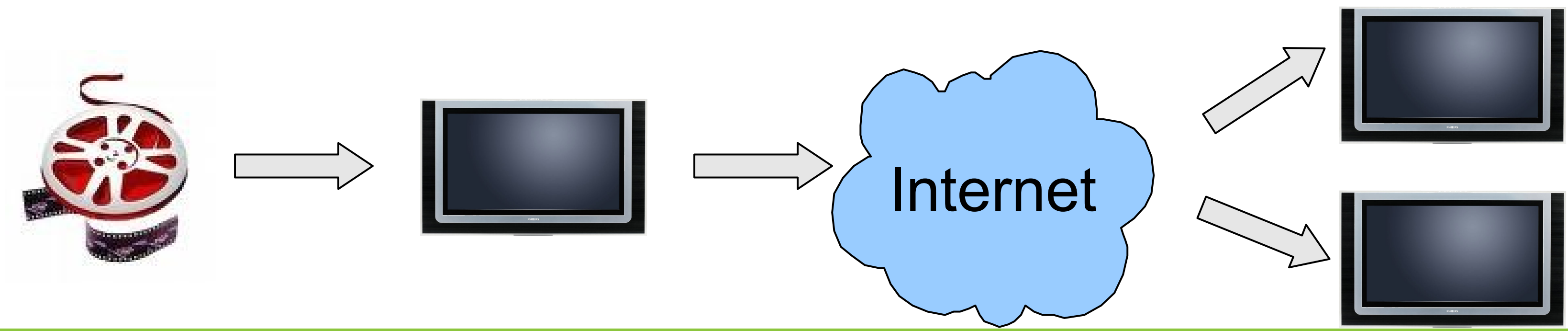




Mentor Graphics EMBEDDED SYSTEMS

Any issues?

- A lot!
- We are going to focus on one – security
- TV contains sensitive data and deals with IP
- Third party applications can not be considered as trusted
- No way to control particular applications
- How to protect TV from badly-behaving apps?



Mentor Graphics EMBEDDED SYSTEMS

General approach

- Define what third-party applications are allowed to do in the system and their resource constraints
- Create a Sandbox – restrict third-party applications access to system resources
- How – depends on a platform

Mentor Graphics EMBEDDED SYSTEMS

Subject

- SPACE - **SP**lit **A**pplication archite**C**tur**E**
  - Digital TV Software platform developed by Philips
  - Based on GNU/Linux
- Sandboxing – one of the open questions
- Use Linux Security Module to restrict third-party applications:
  - SELinux
  - **SMACK**
  - TOMOYO

Goal

Evaluate:

- How to apply SMACK LSM to implement third-party application sandboxing on SPACE platform
- What we have to pay for it – CPU, memory, sanity..

Agenda

- SPACE
- SMACK
- Third-party application access control requirements
- How to address the requirements using SMACK
- Proposed solution
- Impact to system:
  - Memory consumption
  - Performance impact

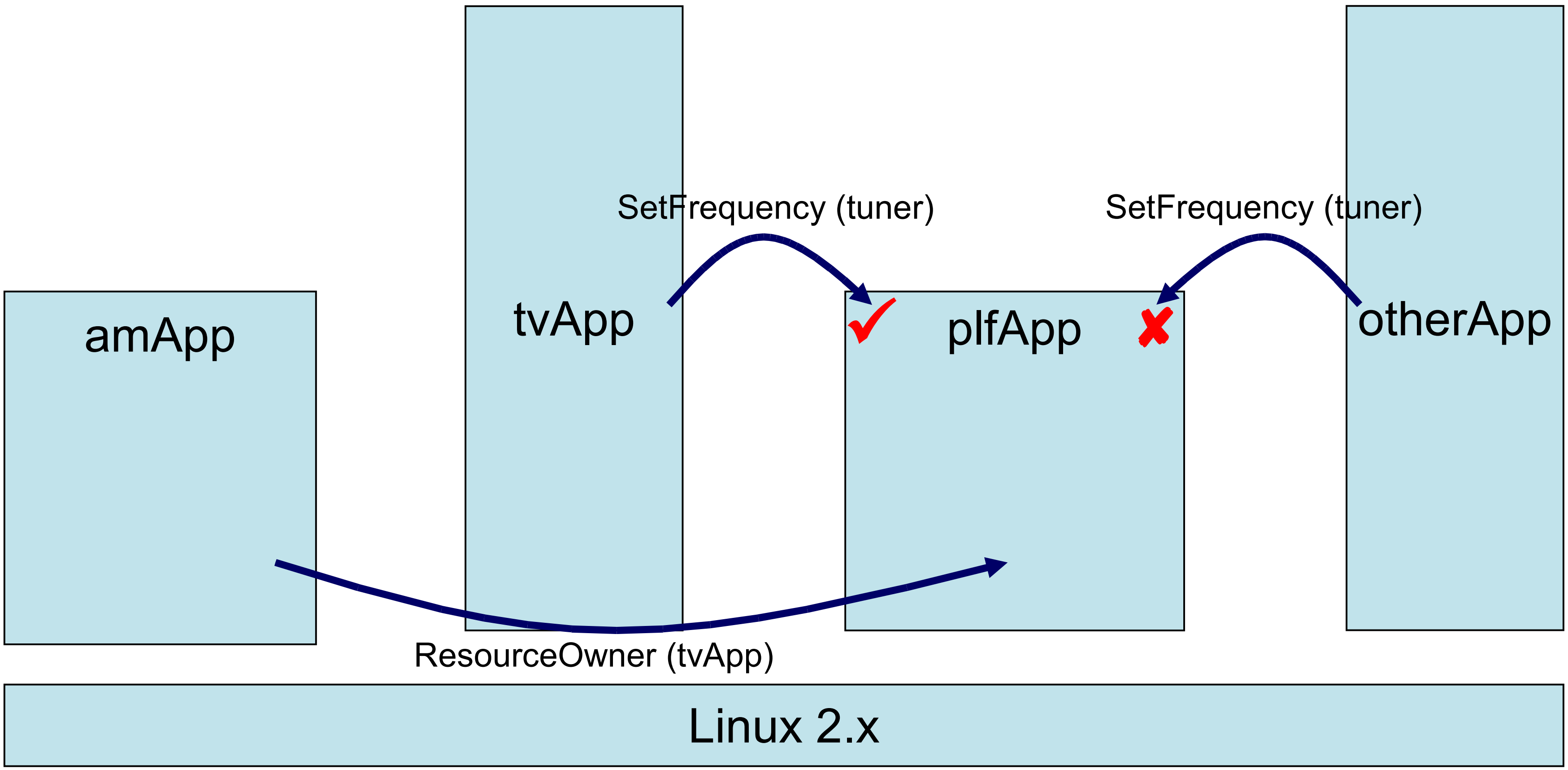
Mentor Graphics EMBEDDED SYSTEMS

SPACE - overview

Based on the open source components:

- Linux kernel 2.6.x
- DirectFB
  - All applications in SPACE are DirectFB applications that create DirectFB window and draw into it
- SaWMan – Shared application and window manager
  - Custom DirectFB window manager module
  - Application manager process is hooked to SaWMan and control life cycle of other applications and their appearance on a screen
- FusionDale
  - DirectFB Fusion library application
  - Implements high-level IPC mechanisms based on shared memory

SPACE - architecture



SMACK - overview

- Simplified Mandatory Access Control Kernel
- Linux Security Module – hooked to various Linux kernel subsystems (file system, network stack)
- On every operation access check is performed according to a system-wide policy (rule set)

SMACK - terms

- Subject
  - Subjects are tasks running in the system
- Object
  - Files, IPC objects, tasks
- Access
  - Any attempt by a subject to put or get any information from a subject
- SMACK Label
  - Security attributes of subjects and objects
  - Stored in extended FS attributes, configuration files or inherited from object owner

Mentor Graphics EMBEDDED SYSTEMS

SMACK - rules

- Default rules

  1. Any access requested by a task labeled "*" is denied.
  2. A read or execute access requested by a task labeled "^" is permitted.
  3. A read or execute access requested on an object labeled "_" is permitted.
  4. Any access requested on an object labeled "*" is permitted.
  5. Any access requested by a task on an object with the same label is permitted.
  6. Any access requested that is explicitly defined in the loaded rule set is permitted.
  7. Any other access is denied.

- Explicit rules

subject-label	*object-label*	*access (rwxa)*
author	book	rw
reader	book	r

Third-party applications access control requirements
Types of applications

- Content viewers
  - Access to pluggable media – USB sticks and external hard drives, SD/MMC cards
  - Access to data partition shared among all the applications in the system
- Entertainment applications
  - Access to hardware acceleration resources
  - Access to multiple input devices that may not be directly handled by vendor software
- Internet services
  - Network access

Third-party applications access control requirements
Requirements list

- No access to certain device nodes
- No access to certain mounted data partitions
- No ability to mount file system
- Limited network access
- Limited access to platform API
- Limited memory consumption
- Limited CPU consumption

How to apply SMACK
Other access control mechanisms

- Other mechanisms
  - Users and groups
  - POSIX capabilities
  - C-groups
- May interfere with LSM/SMACK, so their impact should minimized (e.g. root gets all capabilities, overrides SMACK)
- LSM/SMACK should be a central mechanism for access control Otherwise, it's a mess – difficult to administrate, easier to break in
- Some requirements can not be directly addressed by SMACK
- Create a hybrid solution with SMACK playing a major role

How to apply SMACK
Addressing the requirements

- No access to certain device nodes
  - Mark protected device nodes with special SMACK label
- No access to certain mounted data partitions
  - Mark with special SMACK label
- No ability to mount file system
  - SMACK allows controlling this privilege basing on a label associated with a process trying to mount file system

How to apply SMACK
Addressing the requirements

- Limited network access
  - SMACK allows assigning labels to external host and networks
  - SMACK supports mapping between SMACK labels and Netlabel/CIPSO
  - All unlabeled incoming packets are labeled with default “ambient” label
  - All together it allows to control traffic between 3rd party applications and external networks
- No ability to create device node
  - Not directly supported by SMACK (general LSM provides a hook for that)
  - Remove CAP_MKNOD POSIX capability to achieve that

- Limited access to platform API
  - SPACE uses DirectFB Fusion IPC library
  - Fusion is based on native Linux IPC, namely shared memory and special device driver helper – so theoretically SMACK can be applied since it works with native Linux IPC mechanisms
  - Fusion “world” is represented by a device node and shared memory mapped file
  - Different API groups should be either moved to different Fusion “world” (memory overhead) or split to a number of memory mapped files per functionality (needs Fusion modification)
  - Out of scope of this work

How to apply SMACK
Addressing the requirements

- Limited memory consumption
  - There is no such an object as Memory quantum
  - Can apply C-groups to control 3rd party applications memory consumption
- Limited CPU consumption
  - This type of control is not directly supported neither by SMACK nor other Linux mechanisms
  - To lower CPU consumption in case of intensive CPU load the application manager can adjust priorities of external applications processes

How to apply SMACK
Running third-party applications

- All third-party applications are running as super user (UID 0)
  - Prevent any correlation with users/groups mechanism
  - Common way for embedded Linux applications
- All third-party applications have all POSIX capabilities disabled
  - UID 0 doesn't give any privilege
- All third-party applications are assigned a special SMACK label
- All third-party applications are put into a special memory C-group
- All third-party applications run with lower priority than vendor applications

How to apply SMACK
Proposed solution – SMACK rule set

- Defines explicit relationships between 3rd party applications and resource groups
- Each resource group is associated with a SMACK label
- Platform applications are trusted and simply override SMACK by having a full set of POSIX capabilities enabled (CAP_MAC_OVERRIDE)
- Accompanied with necessary SMACK configuration files
  - Network ambient label, mapping between hosts/nets and SMACK labels

How to apply SMACK
Proposed solution – SPACE changes

- Initialization sctipts

  Init scripts to label system resources and enforce SMACK ruleset

- Platform API changes

  Expose different API groups as separate native Linux objects (e.g. device node, memory mapped file)

- External Application installer
  - Use special key to sign 3rd party applications during build process
  - Download an application from application store or USB stick
  - Check against the key
- External Application launcher

Mentor Graphics EMBEDDED SYSTEMS

Applied SMACK

- It's been all theory before
  - Input data: Linux/SMACK sources, SPACE documents and open source components, DigitalTV platform to explore SPACE
  - Result: Proposed solution
- Need to create a reference implementation to verify that the theory is going to work
- Implemented SMACK rule-set, simple application launcher and simulate third-party application
- Verified that the rule-set works
- Measured SMACK overhead

Applied SMACK
Test environment

- Hardware platform – performance analysis
  - NXP TV543 DigitalTV board, MIPS 300Mhz
  - Linux kernel 2.6.27.9 – SMACK doesn't support host/net labeling
- Software platform – verification and memory footprint
  - QEMU 0.9.1 for MIPS Malta Core LV
  - Linux kernel 2.6.30-rc7
- Root file system
  - Glibc-2.9
  - Busybox 1.7.2 with a SMACK patch from smack-util-1.0 package applied
  - attr-2.4.43 package to manipulate extended file attributes
  - libcap-2.16 library to manipulate POSIX capabilities of a process
  - SMACK rule set stored in /etc/smack/load
  - SMACK test applications

Applied SMACK Labeling resources

- *third_party*

Assigned to all third-party applications running in the system

- *tv*

Assigned to Digital TV specific data

- *ext_media*

Assigned to mount point and files located on external media (e.g. MMC/SD card)

- *prot_device*

Device node that should be protected from third-party applications

- *open_device*

Device node open for third-party applications

- *trusted_net*

Network resources open for third-party applications

- *untrusted_net*

Network resources closed for third-party applications

Applied SMACK
The ruleset

1.	third_party	open_device	rw
2.	third_party	ext_media	rw
3.	third_party	trusted_net	w
4.	trusted_net	third_party	w
5.	_	trusted_net	w
6.	trusted_net	_	w
7.	untrusted_net	_	w
8.	_	untrusted_net	w

Applied SMACK
Third-party application loader

- Starts as a normal application as root
- Forks a new process
- Sets SMACK label
- Disable all POSIX capabilities
- Executes application binary

Example:

run_app -n -l third_party wheather_applet -s weather.net

Applied SMACK Verification

- Preparation script

  Creates device nodes, files and directories simulating TV resources. Label them with corresponding SMACK labels.

- Client-Server application

  Simple server (listen on TCP port) and client. Two instances of the server run on two QEMU VMs (one is “trusted host”, another is “untrusted”)

- SMACK test script
  - Simulates 3rd party applications.
  - Tries to access various resources created by the preparation script
  - Connects to servers running on the trusted and untrusted VMs
  - Reports errors if: prohibited operation is allowed and vice versa
- NO errors were reported by the script

SMACK system impact
Memory consumption analysis

Includes:

- Source code examination
- SMACK module object file analysis
- Run-time memory allocation tracing

Tools:

- GNU binutils for MIPS
- Built-in Linux capabilities
  - /proc/meminfo
  - SLAB allocator tracer (kmemtrace)
- GNU debugger for MIPS

SMACK system impact
Static memory consumption

SMACK module built-in part of Linux 2.6.30-rc7

- Total 24KBytes
- Code: 20722 Bytes
- Static data: 1304Bytes

SMACK system impact
Dynamic memory consumption

- Code analysis shown just a few places where SLAB is called
- For every IPC object (e.g. file, socket) SMACK implements an associated structure:
  - 28 Bytes per fs inode object in memory
  - 24 Bytes per mounted file system (super block)
  - 32 Bytes per socket
- SLAB allocator overhead – 128 Bytes per each SMACK structure

Mentor Graphics EMBEDDED SYSTEMS

SMACK system impact
Dynamic memory consumption

- 644 KBytes difference in dynamic memory consumption
- Includes 587 KBytes allocated for file system super blocks, inodes and socket objects
- Running the SMACK test script does not affect dynamic memory consumption

SMACK system impact
Performance analysis

- File system performance tests only – due to older kernel version 2.6.27.9
- Bonnie++ to create files of different size
  - SMACK impact on file manipulation operations
  - Measures the number of files created per second
  - 10240 files of 0, 1 and 10240 Byte sizes
- Bonnie++ to write large amounts of data
  - SMACK impact on read/write operations
  - Measures the number of bytes read/written per second
- Copying files located in RAM based file system (tmpfs)
  - 'cp' and 'dd' with 1, 8, 64, 1024 KByte block sizes
  - 10 MByte files
  - Measures the number of bytes read/written per second

SMACK system impact
Performance analysis results

- Relative to results retrieved non-SMACK kernel
- File creation (sequential and random order)
  - 0 Byte: 5% degradation
  - 1 Byte: 6% degradation
  - 10 Kbyte: 12% degradation
  - Write buffers disabled: 2-4% degradation (SMACK is compensated by I/O overhead)
- File deletion
  - Random order: 7-10% degradation
  - Sequential order: Up to 30% degradation
  - Write buffers disabled: 1-3% degradation in both cases (SMACK is compensated by I/O overhead

SMACK system impact
Performance analysis results

- Read operation
  - Neither 'bonnie++' nor 'cp' tests shown any degradation
- Write operation
  - 'Bonnie++', 'cp' and 'dd' shown 0-5% degradation depending on block size
  - Worst result of 5% degradation is for byte-to-byte writing to a file of 10 MByte size
  - Smaller block size results in higher overhead

Conclusion

- Formulated access control requirements for third-party applications running is SPACE
- Created a solution to address the requirements
- Proven that the solution is working and suitable for embedded TV platforms

Highlights

- SMACK itself is not enough to create a comprehensive solution for third-party application sandboxing
- Using high-level IPC mechanisms may complicate the solution depending on how high-level IPC is mapped to native Linux IPC
- It's a proof of concept

That's it! Time for Q&A

- SMACK home page

  http://schaufler-ca.com/

- SMACK for DigitalTV whitepaper

  http://elinux.org/Security#Papers

- SPACE

  http://jointspace.sourceforge.net/