LINUX PLUMBERS CONFERENCE

AUGUST 24-28, 2020

Improving SEPolicy Development Experience

Nagaravind Challakere, Microsoft

Shaylin Cattell, Univ. of British Columbia

Who are we?

Nagaravind Challakere

- SW Engineer at Microsoft Devices
- Surface Duo

Shaylin Cattell

- Computer Engineering at Univ. of British Columbia
- Summer Intern 2020 at Microsoft Devices

What this talk is about?

- This talk is *not about*
    - SELinux Policy Languages (Kernel, CIL)*
    - Linux Security Modules or SELinux integration with LSM in Android

* Not yet!

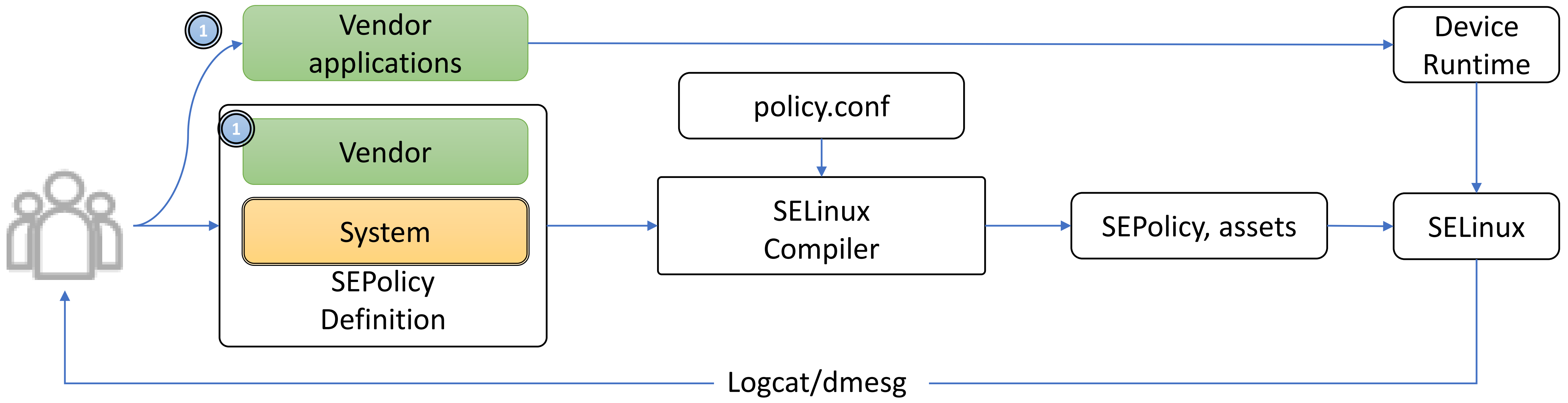
What this talk is about?

- This talk is *not about*
  - SELinux Policy Languages (Kernel, CIL)*
  - Linux Security Modules or SELinux integration with LSM in Android

- This talk *is about*
  - Channeling developer effort to “the what”, not “the how”
    - What tools would be helpful to realize it?
  - Can we rethink SEPolicy Development Workflow if different tools are made available?

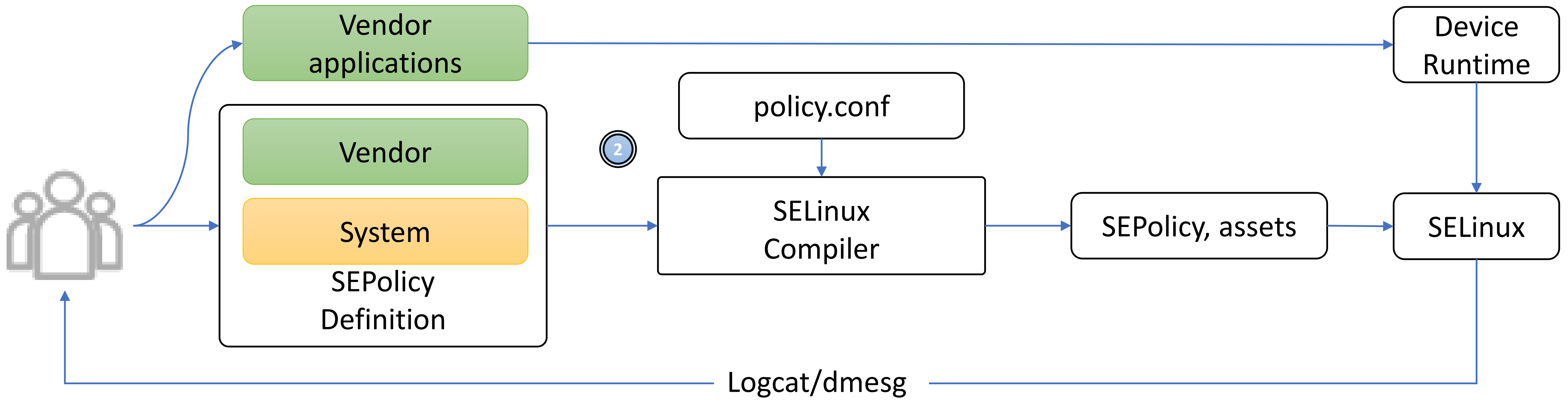
* Not yet!

SEPolicy Developer Workflow



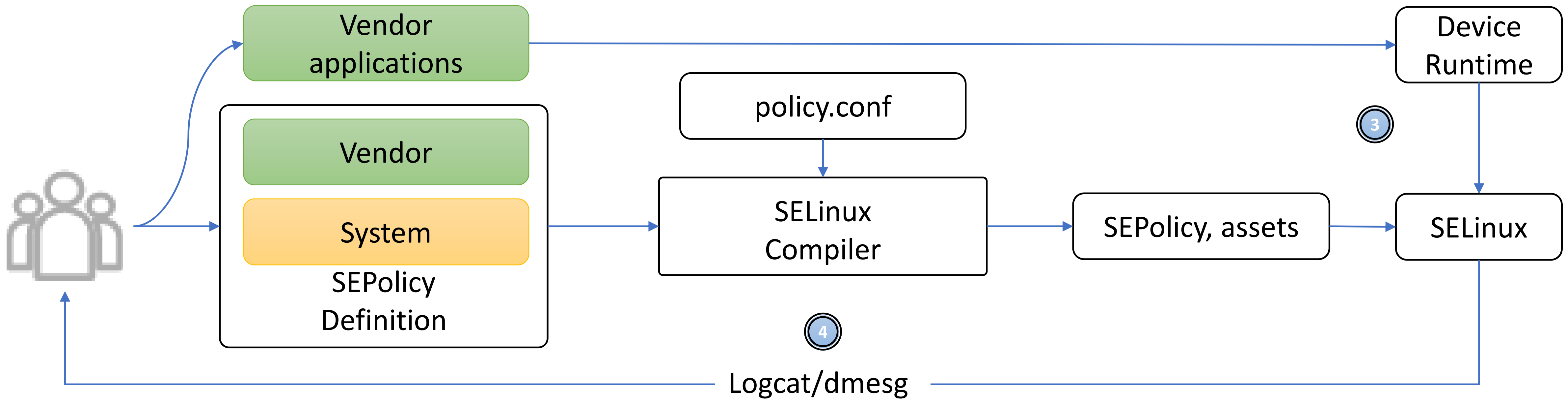
- Developer makes incremental changes to functionality
- Modifies vendor policy definitions

SEPolicy Developer Workflow



- Policy is compiled
- policy.conf – avoid negative patterns
  - Partial Definition of security model

SEPolicy Developer Workflow



- SEPolicy is loaded
- Result of changes in this iteration are available via logs
  - Denials prompt revisions

Pitfalls of existing workflow

- Granting excessive permissions
- Overloading domains with responsibilities
- Accrued permissions not revoked

Anti-patterns

Pitfalls of existing workflow

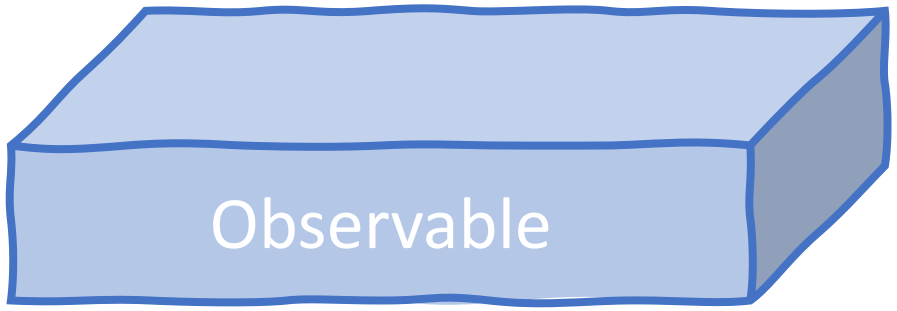
- Granting excessive permissions
- Overloading domains with responsibilities
- Accrued permissions not revoked
- Hard to predict if a change is effective
- No way to measure impact of an incremental change
- No invariants defined for vendor policy
- Equivalence classes

Anti-patterns

Lack of tools

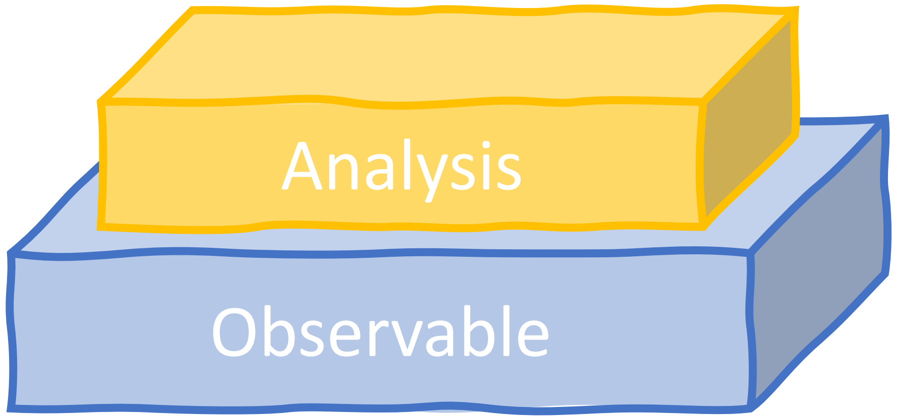
Improving policy development

- Enumerate
  - types
  - type hierarchy
  - permissions
  - transitions
- Ability to construct complex queries
- Tresys setools v4
  - sesearch, sedta, seflowinfo



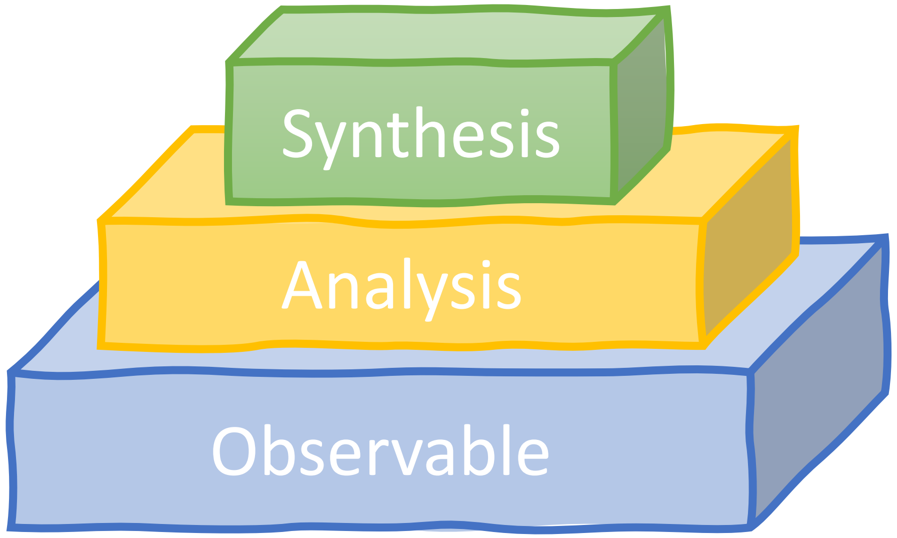
Improving policy development

- Semantic analysis
  - Get orphaned types
  - Identify gaps in domain transition
  - Identify overlapping policy definitions
- Error identification
  - Mapping denials to policy definition

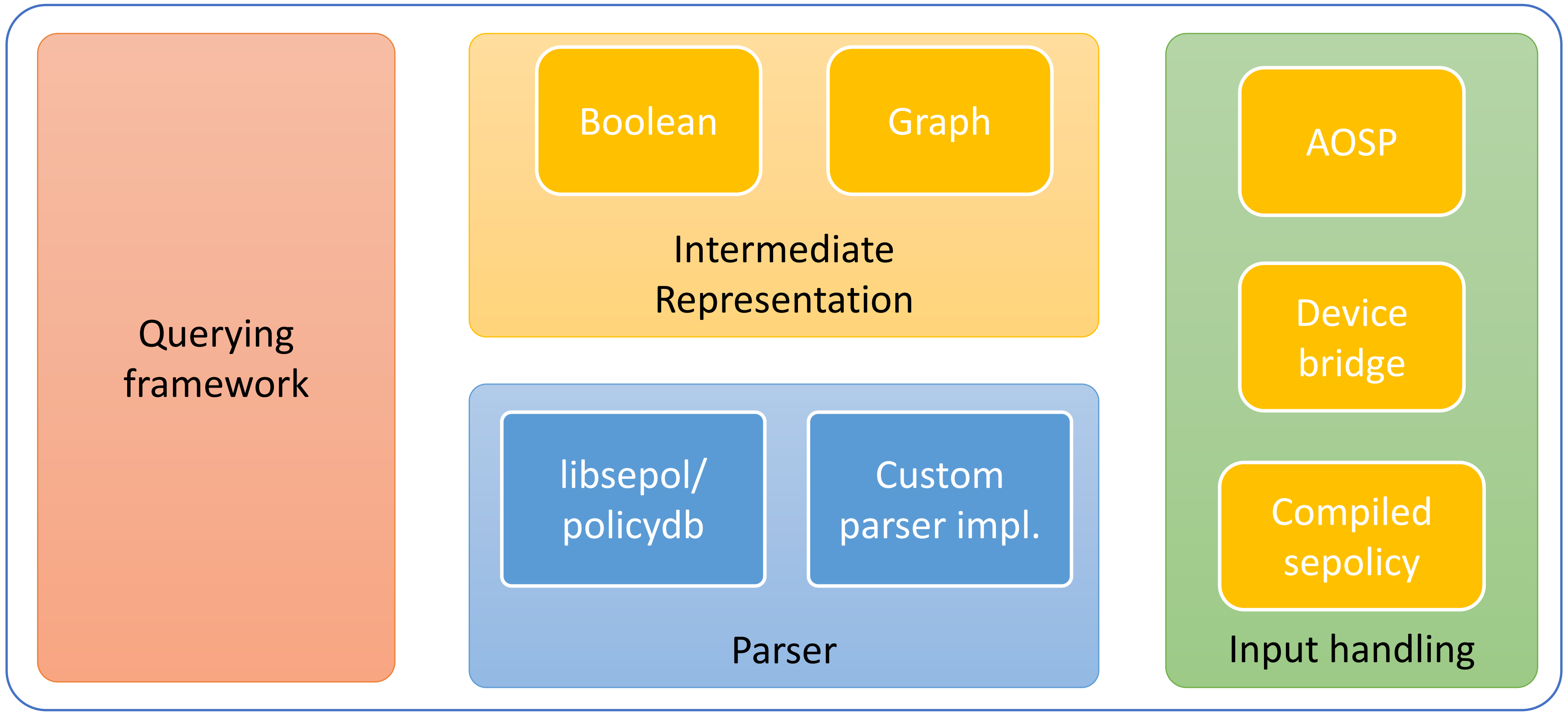


Improving policy development

- Template based checks
  - Reinforce best practices
- Error identification
  - Mapping errors to policy definition
  - Mapping denials to policy definition



SEPolicy Analyzer



SEPolicy Analyzer

Command	Description	Example
get_domains <domain> [filter=<fil_options>]	Display the domains that match the input regex	get_domains hal_usb_.*
get_labels <type> [filter=<fil_options>]	Display the labels (non-domain types) that match the input regex and all attributes	get_types hal_.*_default_exec
get_type_heirarchy <type> [filter=<fil_options>]	Display the attributes that match the input regex, as well as the types/domains associated with them.	get_type_heirarchy halserverdomain
get_permissions <domain> [filter=<fil_options>]	Display the rules (allow and neverallow) that contain the input type/domain/attribute as a source. Input must be an exact name (not regex).	get_permissions vendor_init
get_accessors <label> [filter=<fil_options>]	Display the rules (allow and neverallow) that contain the input type/domain/attribute as a target type. Input must be an exact name (not regex).	get_accessors wifi_prop

SEPolicy Analyzer

Command	Description	Example
get_paths <path> [filter=<fil_options>]	Display the file contexts that overlap with the input regex path, as well as their labels and constraints (e.g. directory only, file only).	get_paths /dev/.*/ram[0-9]
get_properties <property> [filter=<fil_options>]	Display the property contexts that match the input regex, as well as their labels and constraints (e.g. string, boolean, int).	get_properties ro.*
get_orphans <label> [filter=<fil_options>]	Display all labels that match the regex input and are not the target type of any allow rules, either directly or through attributes.	get_orphans .*
get_missing_dt	Identify potential bugs in domain transition definition	get_missing_dt
get_dt [filter=<fil_options>]	Display domain transitions from the policy	get_dt

SEPolicy Analyzer – Filters and Collapse

- Every querying command can contain an optional filter clause E.g.

```
filter={filterType1:"filterValue1", filterType2:["filterValue2", "filterValue3"],
disjointTypes:"filterType2"}
```

- Each filterType in the filter is a conjoint expression
- It is possible to query for disjoint values for multiple types
- collapse=True can be used to combine multiple different rules to get effective set of permissions
    - Useful in building abstractions e.g. checking policy equivalence

E.g:

```
filter={permission:[read, write]}
```

```
filter={permission:[read, write], disjointTypes:permission}
```

```
filter={source:[vendor_init, halserverdomain], permission:[read, write],
disjointTypes:[source, permission]}
```

Discussion

Code: https://aka.ms/sepolicy-analyzer

References

- https://source.android.com/security/selinux
- SELinux project wiki
- SELinux Notebook
- Model based analysis of large datacenter networks
- Batfish: Open source network configuration analysis tool – Commercially available

LINUX PLUMBERS CONFERENCE

AUGUST 24-28, 2020

Thank You

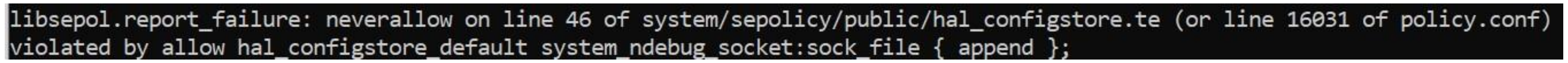
Windows + Devices

Backup

Use Cases

Debug a compile error from conflicting rules

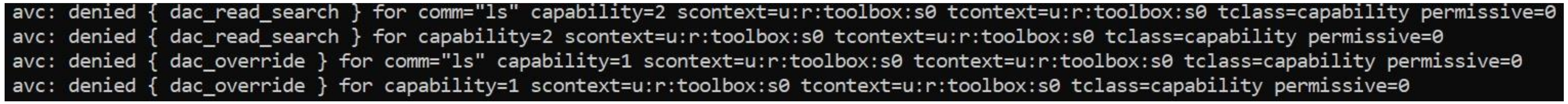
- Search through rules by source, target, and permissions to easily view rules



```
libsepol.report_failure: neverallow on line 46 of system/sepolicy/public/hal_configstore.te (or line 16031 of policy.conf)
violated by allow hal_configstore_default system_ndebug_socket:sock_file { append };
```

Debug a permission denial

- Check whether a target is inaccessible
- Check whether a rule allowing permission exists
- View labels assigned to files and properties



```
avc: denied { dac_read_search } for comm="ls" capability=2 scontext=u:r:toolbox:s0 tcontext=u:r:toolbox:s0 tclass=capability permissive=0
avc: denied { dac_read_search } for capability=2 scontext=u:r:toolbox:s0 tcontext=u:r:toolbox:s0 tclass=capability permissive=0
avc: denied { dac_override } for comm="ls" capability=1 scontext=u:r:toolbox:s0 tcontext=u:r:toolbox:s0 tclass=capability permissive=0
avc: denied { dac_override } for capability=1 scontext=u:r:toolbox:s0 tcontext=u:r:toolbox:s0 tclass=capability permissive=0
```

Query for other policy content

- Query for types, domains, attributes, etc.
- Filter results by any relevant field

Example

- Identify the permissions of domains in setting a property

```
get_types vendor_custom_prop
get_accessors vendor_custom_prop
get_type_hierarchy property_type
get_accessors vendor_related_prop
get_permissions domain filter={targetClass=“property_service”}
get_permissions su filter={targetClass=“property_service”}
```

Performance

Element	Number
Types	1007
Domains	213
Attributes	138
Allow rules	8973
Neverallow rules	949
File contexts	668
Property contexts	527

Size of Sample Policy

Function	Time for Response
GetDomains	1 – 20 ms
GetChildDomains	< 1 ms
GetPermissionsForDomain	1 – 20 ms
GetAccessorsForLabel	1 – 20 ms
GetLabelsForPath	50 – 300 ms
GetLabelsForProperty	< 1 ms
GetOrphans	10 – 200 ms

Response Time of Analyzer