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
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

- Querying framework
- Boolean
- Graph
- Intermediary Representation
- libsepol/policydb
- Custom parser impl.
- Parser
- AOSP
- Device bridge
- Compiled sepolicy
- Input handling

[Image with flowchart and Windows + Devices logo]
### SEPolicy Analyzer

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_domains &lt;domain&gt; [filter=&lt;fil_options&gt;]</td>
<td>Display the domains that match the input regex</td>
<td>get_domains hal_usb_.*</td>
</tr>
<tr>
<td>get_labels &lt;type&gt; [filter=&lt;fil_options&gt;]</td>
<td>Display the labels (non-domain types) that match the input regex and all attributes</td>
<td>get_types hal_.*.default_exec</td>
</tr>
<tr>
<td>get_type_heirarchy &lt;type&gt; [filter=&lt;fil_options&gt;]</td>
<td>Display the attributes that match the input regex, as well as the types/domains associated with them.</td>
<td>get_type_heirarchy halserverdomain</td>
</tr>
<tr>
<td>get_permissions &lt;domain&gt; [filter=&lt;fil_options&gt;]</td>
<td>Display the rules (allow and neverallow) that contain the input type/domain/attribute as a source. Input must be an exact name (not regex).</td>
<td>get_permissions vendor_init</td>
</tr>
<tr>
<td>get_accessors &lt;label&gt; [filter=&lt;fil_options&gt;]</td>
<td>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).</td>
<td>get_accessors wifi_prop</td>
</tr>
</tbody>
</table>
## SEPolicy Analyzer

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

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

\[
\text{filter} = \{ \text{filterType1: "filterValue1", filterType2: ["filterValue2", "filterValue3"],}\n\text{disjointTypes: "filterType2"}\}
\]

• 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:

\[
\text{filter} = \{ \text{permission: [read, write]} \}
\]
\[
\text{filter} = \{ \text{permission: [read, write], disjointTypes: permission} \}
\]
\[
\text{filter} = \{ \text{source: [vendor_init, halserverdomain], permission: [read, write], disjointTypes: [source, permission]} \}
\]
Discussion

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

• [https://source.android.com/security/selinux](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
Thank You
Backup
Use Cases

Debug a compile error from conflicting 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 sock_file { append; }

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

Debug a permission denial

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

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

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

```plaintext
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

### Size of Sample Policy

<table>
<thead>
<tr>
<th>Element</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Types</td>
<td>1007</td>
</tr>
<tr>
<td>Domains</td>
<td>213</td>
</tr>
<tr>
<td>Attributes</td>
<td>138</td>
</tr>
<tr>
<td>Allow rules</td>
<td>8973</td>
</tr>
<tr>
<td>Neverallow rules</td>
<td>949</td>
</tr>
<tr>
<td>File contexts</td>
<td>668</td>
</tr>
<tr>
<td>Property contexts</td>
<td>527</td>
</tr>
</tbody>
</table>

### Response Time of Analyzer

<table>
<thead>
<tr>
<th>Function</th>
<th>Time for Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetDomains</td>
<td>1 – 20 ms</td>
</tr>
<tr>
<td>GetChildDomains</td>
<td>&lt; 1 ms</td>
</tr>
<tr>
<td>GetPermissionsForDomain</td>
<td>1 – 20 ms</td>
</tr>
<tr>
<td>GetAccessorsForLabel</td>
<td>1 – 20 ms</td>
</tr>
<tr>
<td>GetLabelsForPath</td>
<td>50 – 300 ms</td>
</tr>
<tr>
<td>GetLabelsForProperty</td>
<td>&lt; 1 ms</td>
</tr>
<tr>
<td>GetOrphans</td>
<td>10 – 200 ms</td>
</tr>
</tbody>
</table>