Lopol:
Logical Policy Analysis in SELinux

Aleks Kissinger
Center for Information Security
University of Tulsa

This research was supported by MPO Contract MDA 904-02-R-0039.

Policy Analysis

- Many directives => one "big picture"
- APOL
  - policy browsing
  - simple dataflow and re-labeling analysis
- SLAT
  - constructs flow graphs from policy
  - transforms graphs and security goals into model checker input to verify
- 2 ends of the spectrum
  - interactive, easy, shallow analysis
  - non-interactive, very flexible but more difficult

Lopol Goals

- Lightweight
- Interactive
- Experimental/iterative
- Backload parsing and representation of the text policy
- Hide "details", focus on rules and queries in logical programming language

PROLOG

0100101010010101010101001100101010101000010101010101010101000010100100

- First order logic
- Good for "gluing" relationships together with shared characteristics
- Atoms, predicates, rules
  - Atom: “thing”, single identifier
    - root
  - Predicate (relation): relationship + named predicate
    - hasrole(root, sysadm_r)
  - Rule (Horn clause): (p1 and p2 and p3) implies Q
    - HasTypeAccess(u, t) :- hasrole(u, r), hastype(r, t)
- Queries against rules try to find values for which the rule is true

Datalog

- Subset of PROLOG, with a couple of tweaks in syntax/terminology
  - atoms are string literals, rather than lowercase words
- Designed to deal with large amounts of data (100K-line SELinux policy, for example)
- Restrictions so queries return in finite space and time
  - Primary restriction is stratification

Representation

01001010100101010101010011001010101010000101010101010101010100010100100

- Works directly from policy.conf
- Policy directives ⇒ relations
  - type_trasition type1 type2:class newtype ⇒
    type_transition(type1, type2, class, newtype)
- Sets and complements flattened, like checkpolicy does before generating the binary AVTab
  - type_trasition {type1a type1b} type2:class newtype ⇒
    type_transition(type1a, type2, class, newtype)
    type_transition(type1b, type2, class, newtype)

Core Relations

01001010100101010101010011001010101010000101010101010101010100010100100

- 9 relations
  - userrole
  - roletype
  - roledominance
  - typealias
  - typetransition
  - avallow
  - avauditallow
  - avauditdeny
  - avdontaudit*
- Identity Relations for types, classes, SID's, users, attributes, and permissions
  - ex. typeis("sysadm_t", "sysadm_t")

Implementation

- Three parts:
- checkpolicy -L
  - versions 1.26 and 1.28 modified to support "-L" flag
  - outputs a directory of .tuples and .map files, read in by…
- bddshell
  - interactive front-end to John Whaley's bddbddb (b5b)
  - fast data encoding + Datalog evaluation using Binary Decision Diagrams
- lopol.datalog
  - low-level rules and relations used to construct analyses

Center for Information Security
www.cis.utulsa.edu The University of Tulsa®

Methodology

- Have: checkpolicy -L output + lopol.datalog
- Need: policy fallacies + methods of correction
- Analysis => rules and queries on rules, iterative process
  1. Primitive rules + test
  2. Generic rules (built on 1) + test
  3. Specific rules (built on 1 and 2)

Primitive Rules

- WritePerms, ReadPerms
  - defines pertinent dataflow permissions for all classes
- Flow
  - uses WritePerms and ReadPerms to define situations where data may flow directly from A to B (class-aware)
- TraceFlow
  - computes the transitive closure of "flow" for basic reachability analysis (class-agnostic)
- TypeAlias
  - matches all aliases of a type

Generic Rules

- PrivilegedType
  - defines types deemed to be "privileged"
  - could be simply a list, or in terms of attributes, reachability, etc.
  - expands membership, using TypeAlias
- TrustedIntermediaries
  - defines "gatekeepers" that can safely mitigate information flow
- TrustedTraceFlow
  - built on TrustedIntermediaries and TraceFlow for a more accurate information flow model

Specific Rules

- SourceType
  - types that may yield vulnerabilities if given access to privileged types
- Query1
  - uses SourceType, TrustedTraceFlow, and PrivilegeType to yield policy flaws

ReadPerms, WritePerms


```
WritePerms(c, p) :- \
  classis(c, "file"),
  permissionis(p, "write").
WritePerms(c, p) :- \
  classis(c, "file"),
  permissionis(p, "append").
# ...many more

ReadPerms(c, p) :- \
  classis(c, "file"),
  permissionis(p, "read").
ReadPerms(c, p) :- \
  classis(c, "file"),
  permissionis(p, "getattr").
# ...many more
```

Flows


```
Flow(t1, t2) :- \
  WritePerms(c, p), \
  avallow(t1, t2, c, p).

Flow(t1, t2) :- \
  ReadPerms(c, p), \
  avallow(t2, t1, c, p).

TraceFlow(a, b) :- Flow(a,b).
TraceFlow(a, c) :- Flow(a,b), TraceFlow(b,c).

# stub
TrustedTraceFlow(a,b) :- TraceFlow(a,b).
```

A Better Flow

01001010100101010101010011001010101010000101010101010101010100010100100

- Generalize by binding types to types + aliases

```
Flow(t1, t2) :- \
  WritePerms(c, p), \
  TypeAlias(t1, ta1), \
  TypeAlias(t2, ta2), \
  avallow(ta1, ta2, c, p).

Flow(t1, t2) :- \
  ReadPerms(c, p), \
  TypeAlias(t1, ta1), \
  TypeAlias(t2, ta2), \
  avallow(ta2, ta1, c, p).
```

The Query

01001010100101010101010011001010101010000101010101010101010100010100100

```
SourceType(t) :- typeis(t, "httpd_t").
SourceType(t) :- typeis(t, "sshd_t").

Query1(source, sink) :-
  SourceType(source), \
  TrustedTraceFlow(source, sink), \
  PrivilegeType(sink).
```

```
b5b> Query1(x,y)?
  ...

Tuples in query@0: (1.)
        (x=httpd_t(471),y=sysadm_t(198))
```

Information Extraction

- Successive queries
- Rule expansion traces/hooks
  - Attaching code to rules
  - To be implemented

Conclusion

- Iterative, testing-based
- Lopol's power depends on good rules
- Well suited to dataflow analysis, or others with a dataflow component
- Rules will become more flexible upon completion of b5b's implementation of Datalog (e.g. negated relations, boolean comparisons)

Future Work

- Modularization of Lopol hooks in checkpolicy
- Rule refinement, tracing features
- GUI
- Trusted type rules
- Core library of primitive and generic rules
- Integration with SLAT, model checkers, libapol