Detecting Access Control Errors

- Given program and set of known program input locations.
- Generate access dependence graphs.
- Generate program dependence graph to capture all information flows.
- Leverage taint analysis to identify security sensitive operations.
- Inject provenance hooks for each security sensitive operation.
- Generate access provenance graphs using additional runtime data.
- Use statically generated access dependence graphs to identify matching access provenance graphs in runtime data and present them to an analyst.

Detecting Data Tampering

- Given program and known locations of user credentials (username, password).
- Leverage taint analysis to identify candidate authorization code within program by intersecting taint labels.
- Generate program chops to understand the relationship of user input to authorization and security sensitive operations.
- Pre-authorization chops.
- Intra-authorization chops.
- Post-authorization chops.
- Analyze computed chops for instruction sequences that dictate data tampering.
- Present tampering instructions to analyst for further investigation.

Evaluation

- Evaluated provenance tracking technique on OpenMRS’s test suite.
- Found 29 cases where authorization was not present.
- Found a single case where permissions were not consistent with similar authorization elsewhere in the program.
- Found a case where a single authorization hook dominated several security sensitive operations, where additional permissions should have been checked.
- Hook injection only induced a 2.1% performance overhead when running the test suite.

Partial Authorization:
- Subject “Admin123” is authorized to getUser() from database.
- Uses object to get login credentials of User without additional authorization.
- Gathers secret question from users credentials.

Consistency:
- Subject “Admin123” is authorized to perform two distinct operations to “edit” and “delete” a person from database.
- Same permission set is used for both operations, which is inconsistent to similar operations elsewhere in the program.

Analysis of maliciously modified version of vsftpd:
- Taint analysis identified 2 functions that perform authorization.
- Generated intra-authorization programs identified 7 LLVM instructions corresponding to a single source line of code related to data tampering.
- Bit manipulation instructions check whether the first 4 characters of the username match “KU3p”. If they do, the username is changed to “root”.

Publications


Supported by National Science Foundation, DARPA, and Lincoln Lab.