Detecting Insufficient Access Control in Web Applications

George Noseevich, Andrew Petukhov
{ngo, petand}@lvk.cs.msu.ru

Security research group of the Computer Systems Lab,
Computer Science Department, Lomonosov Moscow State University,

1st SysSec Workshop

Overview

What?

- Detecting broken access control in web applications

How?

- Modified “differential analysis”, black-box

Results

- A method and a tool, AcCoRuTe
- Evaluation on real-word web applications
- Previously-unknown vulnerabilities discovered

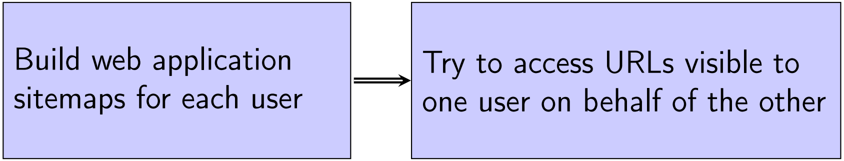
Access control testing - challenges

- Web applications provide for virtually unlimited set of interactions and sequences thereof
- How do we distinguish an authorized worflow from unauthorized without explicit specifications?
- How do we select a limited subset of actions to check for access control violations?

Assumption

User should only be allowed to perform actions listed in his web interface

Basic “differential analysis”



Build web application sitemaps for each user

Try to access URLs visible to one user on behalf of the other

Limitations

- Failiure to capture action interdependencies leads to incomplete sitemaps
- Uncontrolled state changes during sitemap crawling result in incorrect testing conditions

Possible solution

- Perform “differential analysis” in a series of web application states
- Preserve state whithin each “differential analysis” round

Questions arise

- How do we select appropriate states?
- How do we tell apart state-changing and state-preserving requests?

Proposed approach: information gathering step

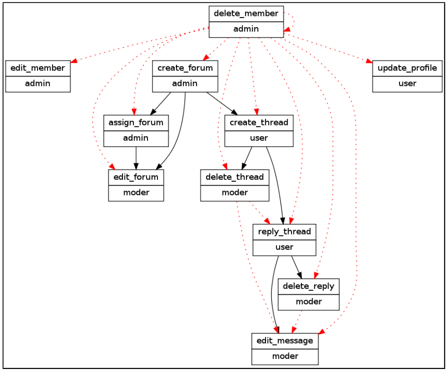
Browser extension captures operator's knowledge about web application business logic

- Roles, users and their credentials
    - *Administrator, Moderator, User*
- State-changing actions
    - *Post message, Delete forum, Assign forum to moderator*
- Action dependencies and cancellations
    - *to delete a message one must write a message*
    - *after a message is deleted it can no longer be modified*

Proposed approach: automated scanning step

Web application scanner performs automated access control test using gathered information

- Recorded actions are organized in a *use-case graph*
- Actions from the graph are carried out in a specific order
- After each performed action, “differential analysis” is performed
- State-changing actions are not performed during the sitemap crawling



Alternative method

White-box approach [Felmetsger et al, 2010]:

- Extract “likely invariants” during web application normal operation using dynamic analysis
- Use model checking to check web application source code for invariant violations
- Was evaluated on `Easy JSP forum` web application (open source message board, approx. 1500 lines of code)
  3 vulnerabilities found, 1 false positive, 5 h. running time

Evaluation

- Easy JSP forum: 5 vulnerabilities found and 1 missed, 1 false positive, 1 h running time (incl. 25 minutes of operator work)
- PyForum: discovered previously-unknown vulnerability that allows editing arbitrary user profiles, including the ability to change passwords (confirmed by developer).

Work in progress

Limitations

- Limited (yet) javascript and AJAX support
- Some alerts do not represent real vulnerabilities
- Hidden content is not discovered

Next steps

- Further automate the process by using static analysis to separate state-changing and state-preserving actions

Questions?