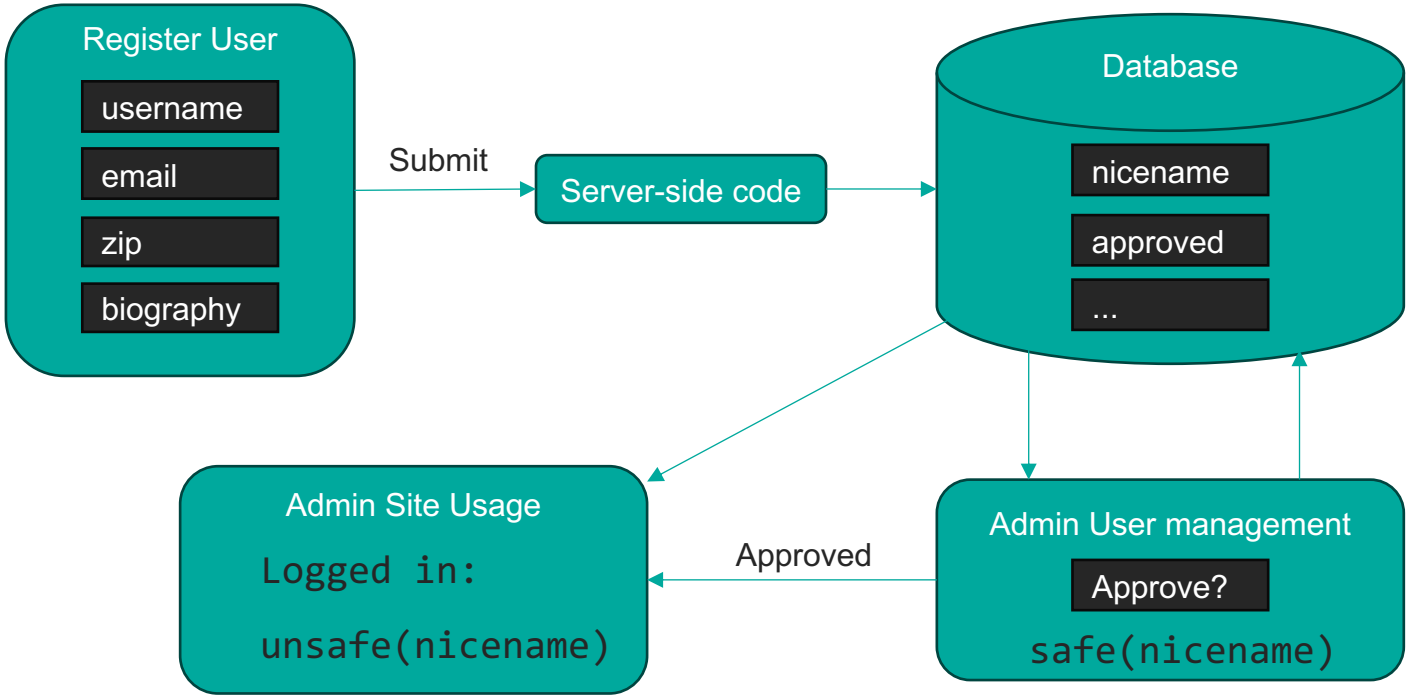
Spider-Scents: Grey-box Database-aware Web Scanning for Stored XSS

Eric Olsson, Benjamin Eriksson, Adam Doupé, Andrei Sabelfeld

Chalmers University of Technology, Arizona State University

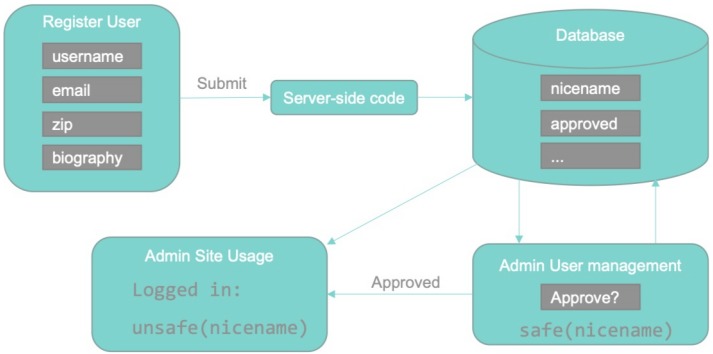
laro@chalmers.se

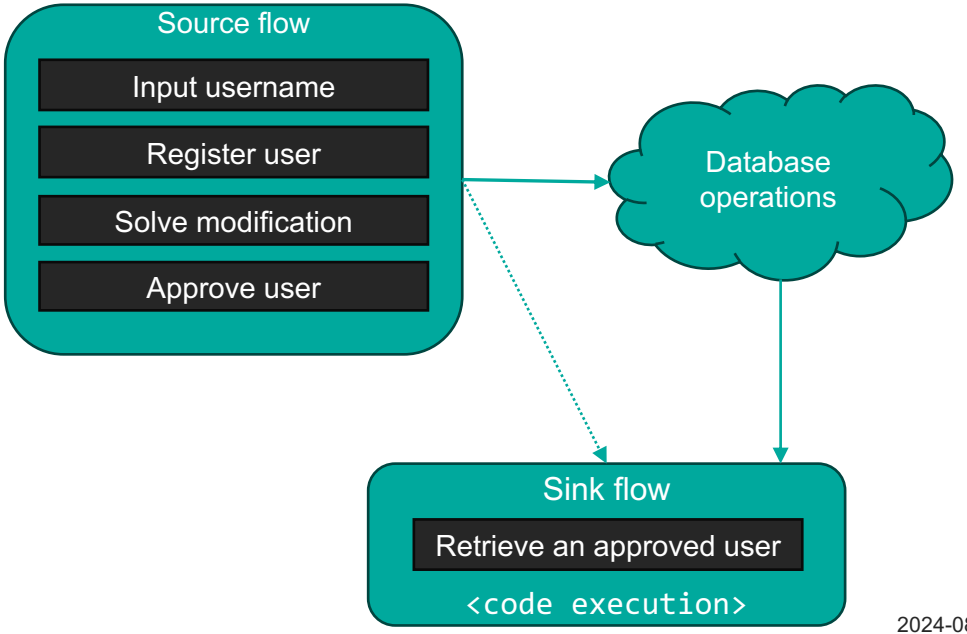
XSS remain elusive, especially stored



Insufficiency of *-box approaches

Problem: find both source and sink flows, and understand their relationship



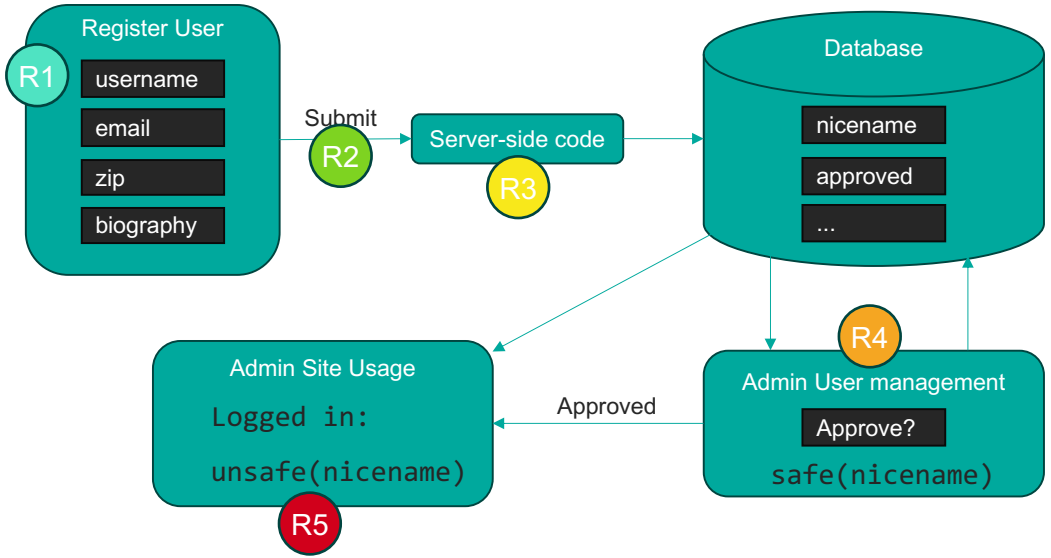


Stored-XSS Scanners face a hard task

5 Roadblocks for current XSS scanners

- R1: Vulnerable input validation
- R2: Interdependent vulnerable input validation
- R3: Vulnerable input modification
- R4: Multi-step vulnerable input
- R5: Vulnerable input identification

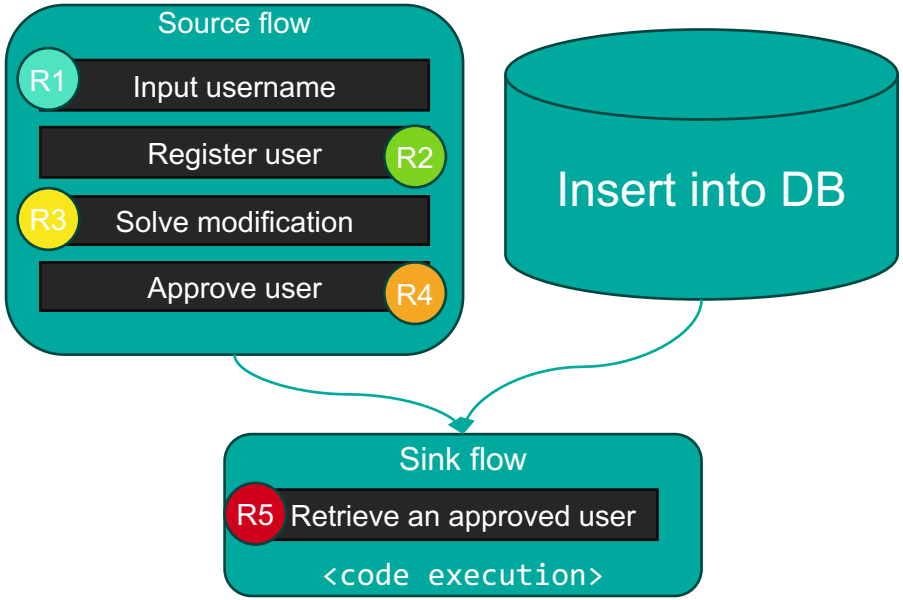
- Fundamentally hard



Novel approach: Relax the problem

Cut the problem in half. Put a payload in the database then check for its execution.

- Inspired by binary fuzzing
- Input is especially hard.
- The database cleanly separates source and sink flows.
- Shift from *application input* to *application state*

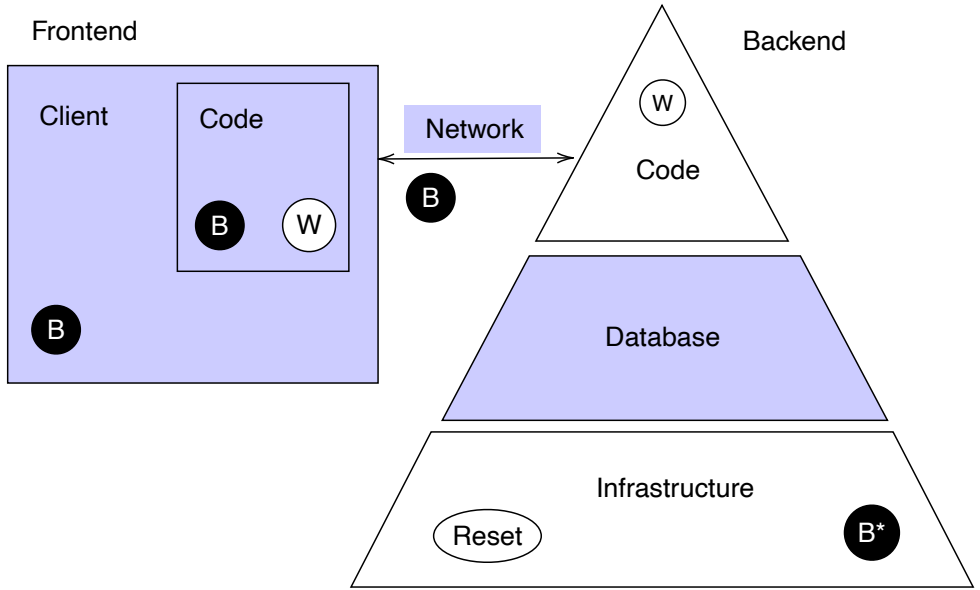


Database-aware web scanning

Add database access to a black-box scanner

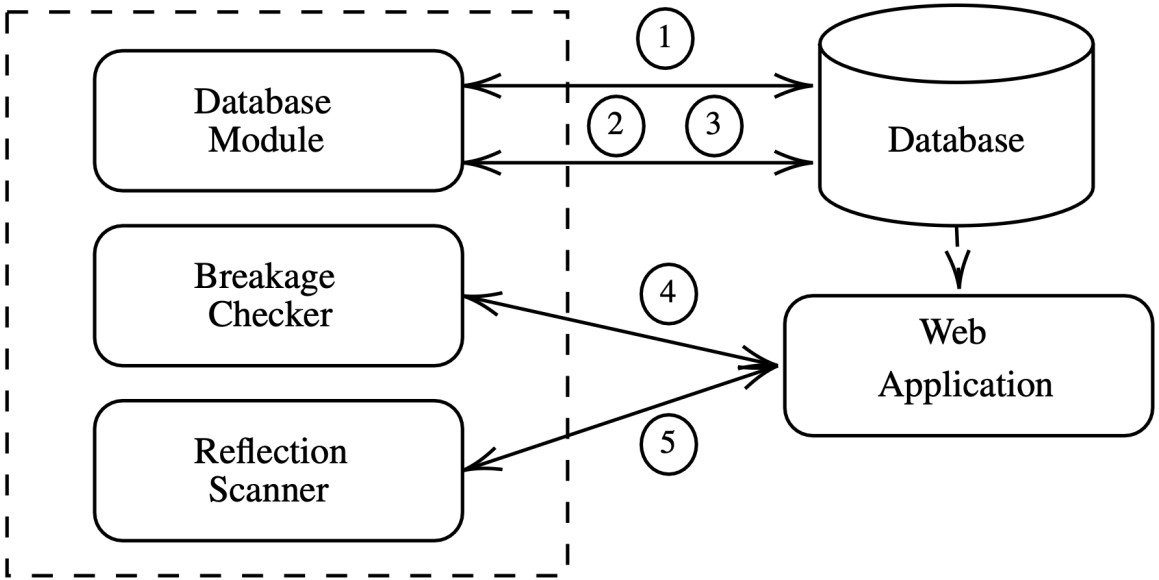
- Client-side access
- Schema
- Data

- Discover unprotected outputs
- These represent code smells
- Dormant XSS or vulnerabilities



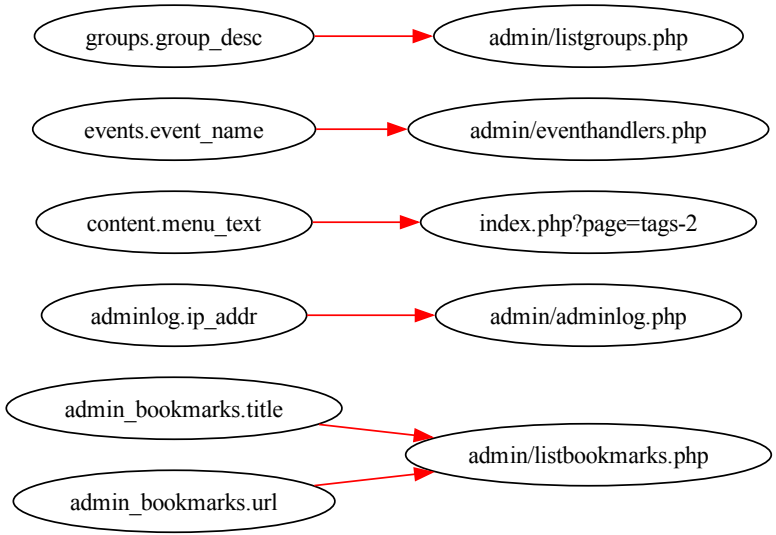
Method

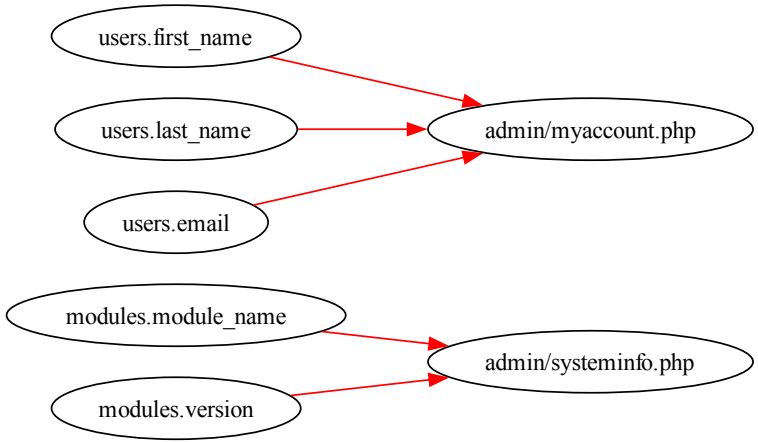
1. Prepare application
   - Insert benign correlated data
2. Choose modification
   - Database cell-level
   - Avoid sensitive rows
3. Inject payload
   - Unique payload
   - Structured data
4. Validate injection
   - Application breakage
5. Look for payload
   - Black Widow



Payload reflections

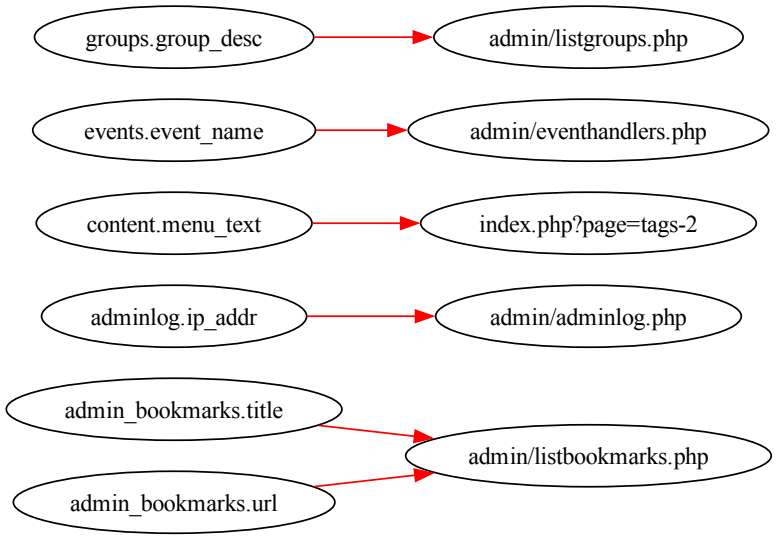
Unprotected outputs from the database





Code smells

Output should be protected - Indication of an underlying fault.



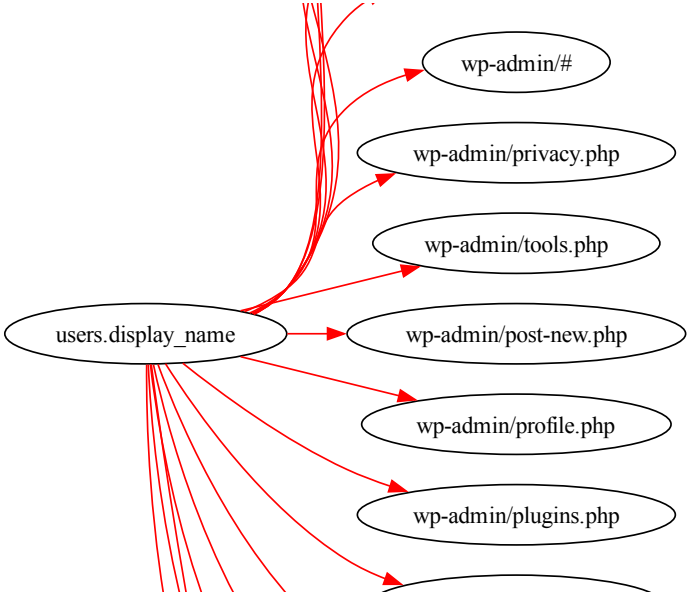

```
echo "<td><a
href=\"editbookmark.php".$urlext."&amp;id=".$id."\">".
$title."</a></td>\n";
echo "<td>".$url."</td>\n";
echo "<td><a
href=\"editbookmark.php".$urlext."&amp;id=".$id."\">";
echo $DisplayImage('edit.gif', 'Edit');
echo "</a></td>\n";
echo "<td><a
href=\"deletebookmark.php".$urlext."&amp;id=".$id."\"
onclick=\"return
confirm('".cms_html_entity_decode($title)."');\">";
echo $DisplayImage('delete.gif', 'Delete');
echo "</a></td>\n";
```

edited for clarity

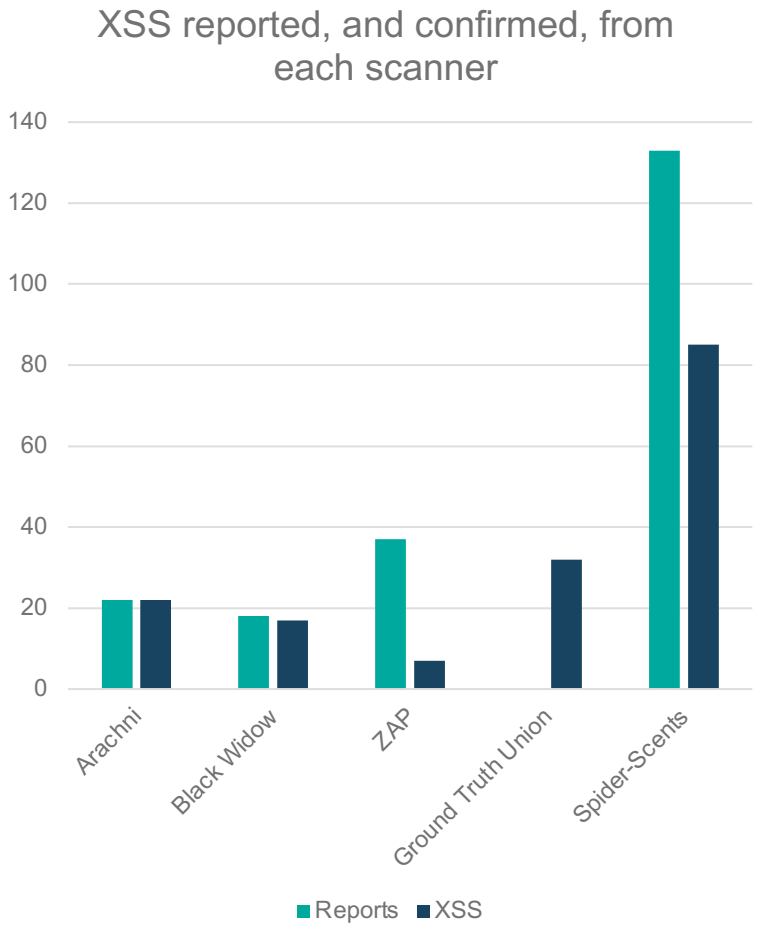
Dormant vulnerabilities

Incomplete vulnerabilities can easily be completed, such as with plugins.

Relationship to stored XSS?

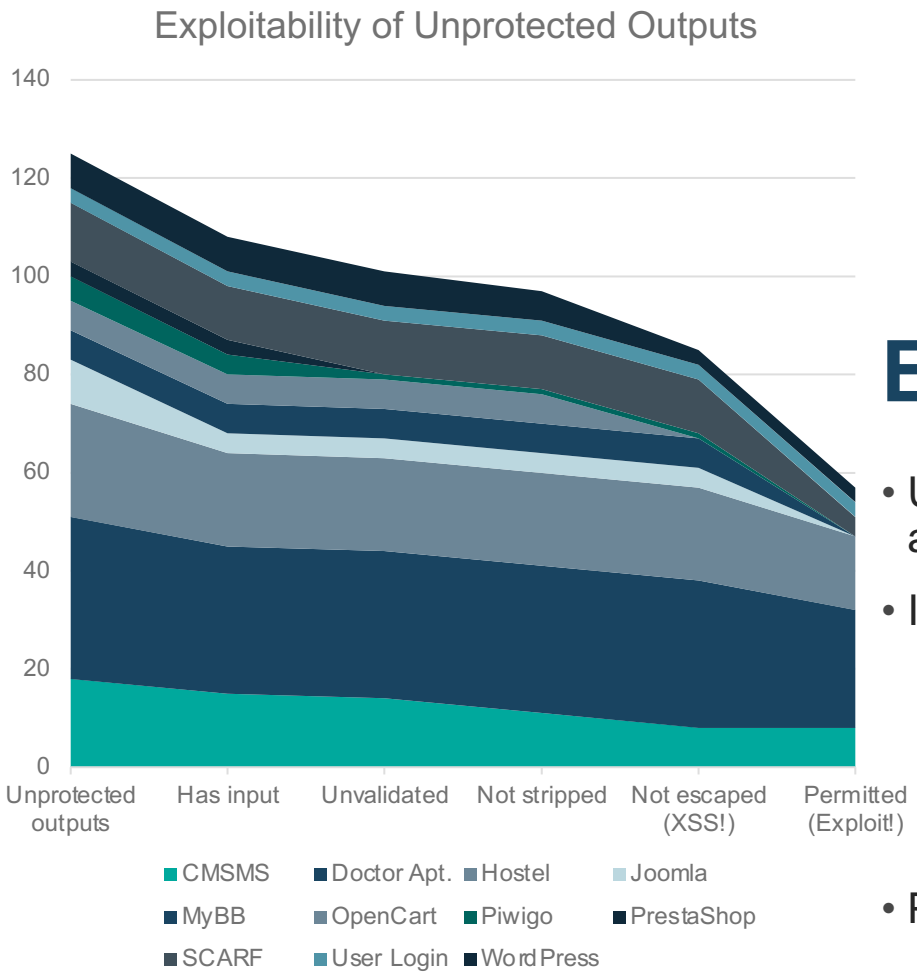



```
$qnn = $wpdb->prepare( "UPDATE $wpdb->users SET
user_nicename = %s WHERE user_login = %s AND
user_nicename = %s", $new_username, $new_username,
$old_username );

$wpdb->query( $qnn );

$qdn = $wpdb->prepare( "UPDATE $wpdb->users SET
display_name = %s WHERE user_login = %s AND
display_name = %s", $new_username, $new_username,
$old_username );

$wpdb->query( $qdn );
```



Evaluation

- More (85) new (53) stored XSS
  - from 133 unprotected outputs
- 8-hour scans
- 10 minutes of manual analysis
- Reference (5) and modern (7) applications
- Ground-truth union of 32 stored XSS
- Cross-reference CVEs



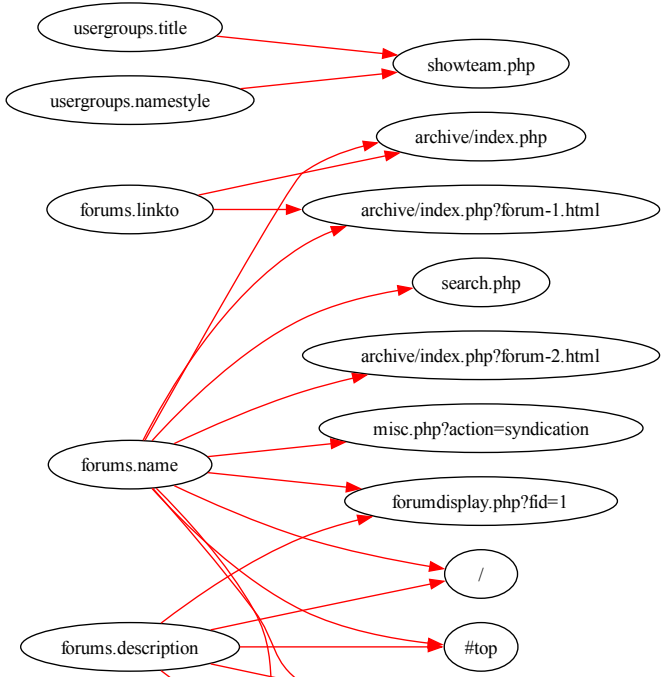
Exploitability of reports

- Unprotected outputs (133), vulnerabilities (85), and exploits (59)
- Input protections (-48)
  - No input (-21)
  - Validation (-7)
  - Stripping (-8)
  - Escaping (-12)
- Permissions (-26)

Evaluation: MyBB (6 new XSS)

Roadblocks R3, R4 (and R5)

- No other scanners detect these XSS
  - Not exploitable; Admin self-XSS
- Other scanners could reach inputs
  - forums.description *(bw)*
  - forums.linkto *(arachni, bw)*
  - forums.name *(bw)*
  - templates.template *(bw, zap)*
  - usergroups.namestyle
  - usergroups.title

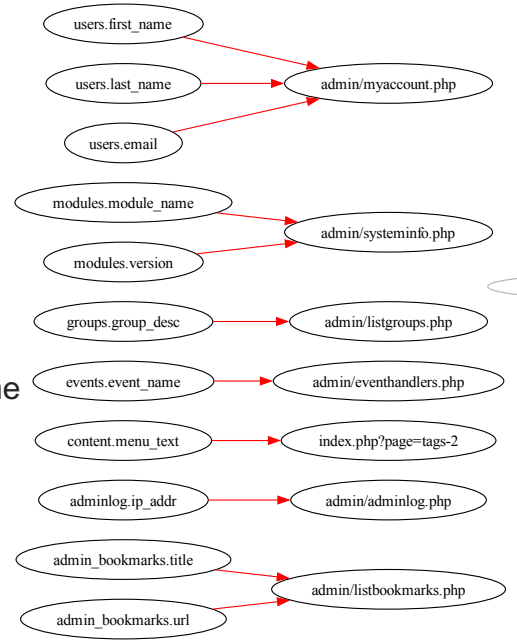


simplified graph to remove templates.template

Evaluation: CMSMS (8 new XSS)

Roadblocks R1, R2, R4; We miss 2 XSS due to payload length

- Unprotected outputs (18)
  - No input: 3, Validation: 1
  - Stripped: 3, Escaped: 3
- Other scanners could not reach inputs
  - admin_bookmarks.title
  - content.metadata
  - layout_stylesheets.media_query/media_type
  - module_news_categories.news_category_name
  - module_news.news_data/summary
  - siteprefs.sitepref_value *(arachni, bw, zap)*



Spider-Scents: Grey-box Database-aware Web Scanning for Stored XSS

Contributions

- Novel grey-box database-aware approach
- Implemented as an open-source prototype
- Practical evaluation
- Systematize the relationship between
  - unprotected outputs
  - stored XSS vulnerabilities
  - exploits

- Code smells – dormant XSS
- Grey-box setting empowers defense

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

www.cse.chalmers.se/research/group/security/spider-scents/

github.com/Spider-Scents/dbfuzz