DELF: Safeguarding deletion correctness in Online Social Networks

Katriel Cohn-Gordon, Georgios Damaskinos, Joshi Cordova, Benoît Reitz, Benjamin Strahs, Daniel Obenshain, Paul Pearce, Ioannis Papagiannis

USENIX Security 2020
Deletion is a core privacy expectation. Yet mistakes in deletion are common.
Deletion Correctness

Great presentation! 😊
So why is it so hard?

Identifying what to delete according to policy is complex.

Deleting the wrong data is very bad.

Developers must remember to implement deletion logic.

Applications change frequently.

Applications have complex data models.

Deletion is seldom top-of-mind when creating new features.

Data is stored across multiple distributed data stores.

Deletions may execute for days and face reliability issues.
Deletion Correctness

Types of mistakes

Developers may omit to specify what to delete

Developers may incorrectly specify to retain data that should be deleted

Developers may inadvertently delete the wrong data

Developers may fail to execute the specification they provided

*NB: We consider benign developers who faithfully try to implement deletion*
DELF Design

- Declarative API for deletion behavior
- Centralized execution of all deletions
- Static, dynamic, and data type correctness validation
- Usable restoration infrastructure
Developers write structured data type specifications

```plaintext
object_type:
  name: photo
storage:
  type: TAO
  deletion: directly
id:
  photo_id:
    type: integer_autoincr
  attributes:
    created_on: datetime
    caption: string
edge_types:
  handle:
    to: photo_blob
    deletion: deep
  created_by:
    to: user
    deletion: shallow
  inverse:
    created_photo:
      deletion: deep
```

DELF annotations specify how data of this type should be deleted

**Objects**

Annotations on objects specify when and how the object is deleted e.g., via cascade from a user deletion.

**Edges**

Annotations on edges specify what happens to related objects upon deletion e.g., cascade and delete the other object.
Types of mistakes

Developers may omit to specify what to delete

Developers may incorrectly specify to retain data that should be deleted

Developers may inadvertently delete the wrong data

Developers may fail to execute the specification they provided

Static Validation

• DELF requires deletion annotations at commit time. Developers cannot create new data types without specifying their behavior when deletion occurs.

Dynamic Validation

• N/A

Data Type Validation

• DELF performs periodic data scans. It pinpoints common types of identifiers such as 64-bit integers. Any data types found to store dangling references are flagged for developers to annotate.
Types of mistakes

Developers may omit to specify what to delete

Developers may incorrectly specify to retain data that should be deleted

Developers may inadvertently delete the wrong data

Developers may fail to execute the specification they provided

Static Validation

- DELF statically validates each object specification, ensuring that every object type contains at least one path leading to deletion of this specific object type

Dynamic Validation

- Heuristics retroactively surface edges that should be marked deep yet developers misclassified as shallow
- DELF confirms objects of types marked directly are being deleted at runtime

Data Type Validation

- N/A
Types of mistakes

Developers may omit to specify what to delete

Developers may incorrectly specify to retain data that should be deleted

Developers may inadvertently delete the wrong data

Developers may fail to execute the specification they provided

Static Validation

- DELF statically suppresses write/delete code generation for objects of type with an annotation that precludes deletion, e.g., `not_deleted`.

Dynamic Validation

- Heuristics retroactively surface edges that should be marked shallow yet developers misclassified as deep
- Privilege escalation checks to preclude exploitation of misclassified deep edge types.
- DELF rejects all calls to delete objects of type with an annotation that precludes deletion, e.g., `not_deleted`.

Data Type Validation

- N/A

DELF also maintains restoration logs for effortless undo!
Types of mistakes

Developers may omit to specify what to delete

Developers may incorrectly specify to retain data that should be deleted

Developers may inadvertently delete the wrong data

Developers may fail to execute the specification they provided

DELF Design: Correctness Validation

Initiation
- When deletion of an object is requested, immediately hide the object.
- Use read-time checks on child objects to hide them as well.

Execution
- Asynchronously traverse the graph and issue point deletes.
- Record deleted data in a restoration log in case of errors.

Completion
- Continue graph traversal until all necessary deletions occur.
- Retry any errors and surface continued failures to an engineer.
Evaluation Highlights

1. We deployed DELF on Facebook’s backend, supporting deletions across many products.
2. We discovered & fixed thousands of edges engineers historically omitted specifying deletion annotations for.
3. Static validation kicks in and helps developers correct deletion annotations for 62.2% of object types they create.
4. Dynamic validation heuristics discover independently most edge type annotations in our sample.
5. DELF restoration logs simplified user data recovery in 20/21 data loss incidents.

<table>
<thead>
<tr>
<th>HEURISTIC</th>
<th>PREDICTS</th>
<th>Prec. (%)</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>to_new_object</td>
<td>deep</td>
<td>86.9</td>
<td>4.7</td>
</tr>
<tr>
<td>to_leaf_object</td>
<td>deep</td>
<td>86.3</td>
<td>12.6</td>
</tr>
<tr>
<td>to_owned_object</td>
<td>deep</td>
<td>91.9</td>
<td>19.3</td>
</tr>
<tr>
<td>id1_ref_in_id2</td>
<td>deep</td>
<td>88.9</td>
<td>29.7</td>
</tr>
<tr>
<td>to_old_object</td>
<td>shallow</td>
<td>94.5</td>
<td>80.2</td>
</tr>
<tr>
<td>self_reference</td>
<td>shallow</td>
<td>100.0</td>
<td>12.0</td>
</tr>
<tr>
<td>many_to_one</td>
<td>shallow</td>
<td>96.4</td>
<td>54.5</td>
</tr>
<tr>
<td>same_obj_type</td>
<td>shallow</td>
<td>90.8</td>
<td>13.9</td>
</tr>
<tr>
<td>to_deleted</td>
<td>shallow</td>
<td>91.7</td>
<td>0.6</td>
</tr>
<tr>
<td><strong>OVERALL</strong></td>
<td><strong>deep</strong></td>
<td><strong>89.7</strong></td>
<td><strong>60.7</strong></td>
</tr>
<tr>
<td></td>
<td><strong>shallow</strong></td>
<td><strong>93.0</strong></td>
<td><strong>89.5</strong></td>
</tr>
<tr>
<td></td>
<td><strong>either</strong></td>
<td><strong>95.0</strong></td>
<td><strong>74.8</strong></td>
</tr>
</tbody>
</table>
Thank you

Follow-up questions: katriel@fb.com