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

FACEBOOK

Deletion is a core privacy expectation. Yet mistakes in deletion are common.

TC Join Extra Crunch

Facebook mistakenly deleted videos

CBS New York NEWS WEATHER SPORTS VIDEO BEST OF MORE

Facebook Blames A 'Bug' For Not Deleting Y

THE VERGE TECH REVIEWS SCIENCE CREATORS ENTERTAINMENT VIDEO MORE

APPS \ TECH

TikTok users over 13 are having their accounts deleted after putting in the wrong birthdays

Double-check those years

By Dami Lee | @dami_lee | Feb 28, 2019, 3:33pm EST

SHARE

[redacted] VERGE

TC Join Extra Crunch Login

Even years later, Twitter doesn't delete your direct messages

cnet COVID-19 BEST PRODUCTS REVIEWS NEWS HOW TO FINANCE HEALTH SMART HOME CARS DEALS

Amazon Alexa transcripts live on, even after you delete voice records

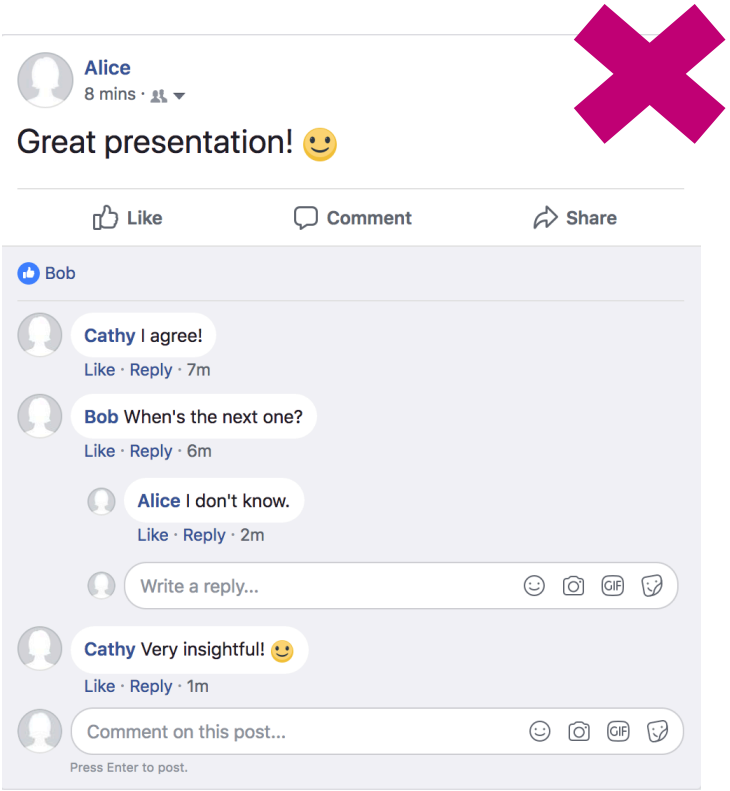
TechRepublic

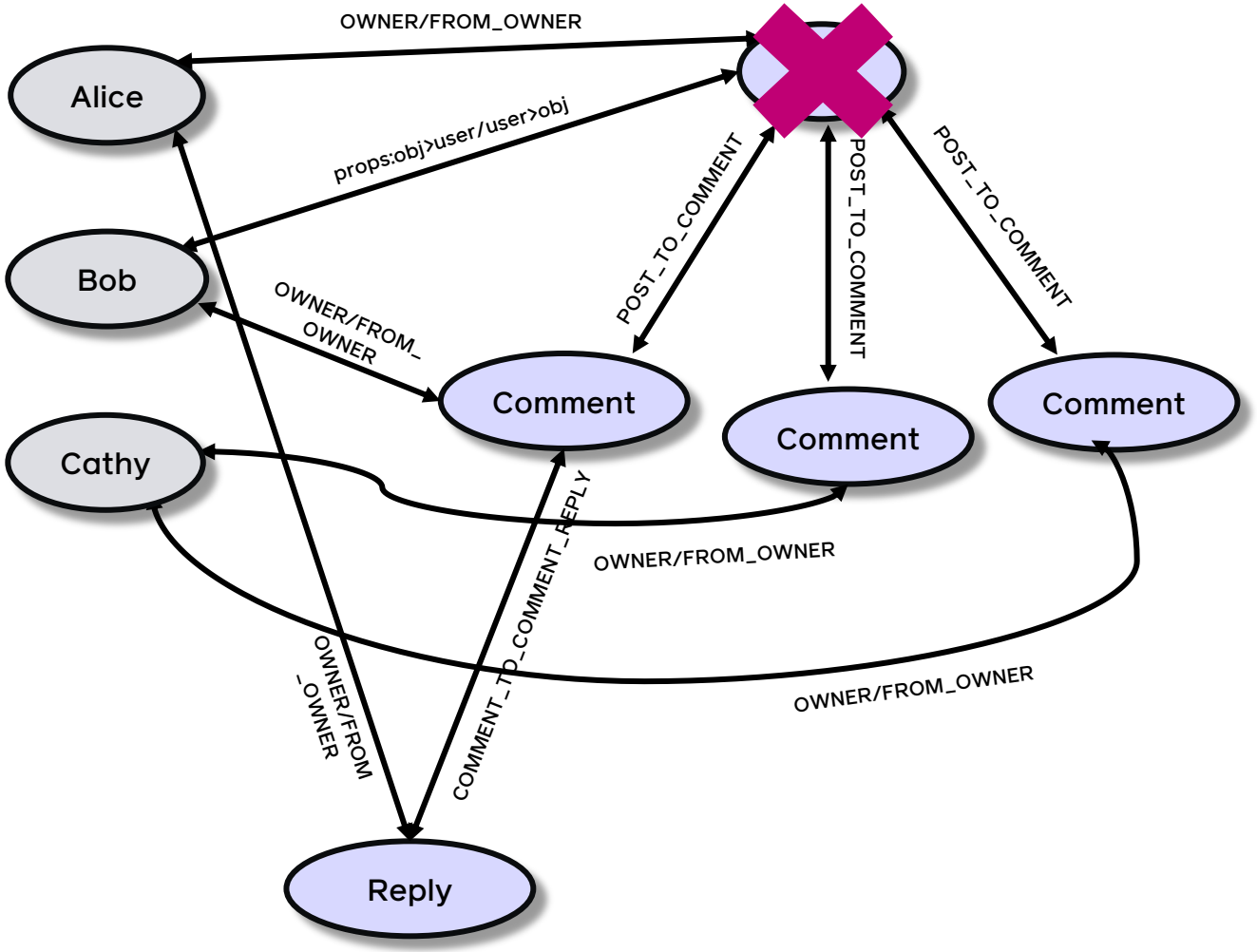
Dropbox bug sends years-old deleted files back to user accounts

by Conner Forrest in Security on January 25, 2017, 7:08 AM PST

A recently discovered bug affecting cloud company Dropbox resulted in data that was

Deletion Correctness





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.

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

```
object_type:
  name: photo
  storage:
    type: TAO
    deletion: directly
  id:
    photo_id:
    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`.

DELF also maintains restoration logs for effortless undo!

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

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.

HEURISTIC	PREDICTS	PREC. (%)	RECALL (%)
to_new_object	deep	86.9	4.7
to_leaf_object	deep	86.3	12.6
to_owned_object	deep	91.9	19.3
id1_ref_in_id2	deep	88.9	29.7
to_old_object	shallow	94.5	80.2
self_reference	shallow	100.0	12.0
many_to_one	shallow	96.4	54.5
same_obj_type	shallow	90.8	13.9
to_deleted	shallow	91.7	0.6
OVERALL	deep	89.7	60.7
	shallow	93.0	89.5
	either	95.0	74.8

Thank you

Follow-up questions: katriel@fb.com

FACEBOOK