forensic analysis of cloud-native artifacts
≡ driving problem
  ‡ cloud drive acquisition & analysis
≡ traditional solution
  ‡ go to the client, look for the leftovers
≡ small detail → it does NOT work:
  ‡ partial replication (data may not be on device)
  ‡ versions (only one on the client)
  ‡ cloud-native artifacts (e.g. Google Docs)
client acquisition is **unsound**

- **-secondary source**
- **AUTHORITATIVE source**
approach: use the API
problems solved
  ‡ partial replication ✓
  ‡ revision acquisition ✓
  ‡ cloud natives? ✓ ✗
this work:
  ‡ provide a solution for cloud-native artifacts in Google Docs
≡ definition

† data objects which maintain the persistent state of web/SaaS applications,
† and have no external representation on the client.

→ these are *internal objects* for the app.
How I REVERSE ENGINEERED GOOGLE DOCS
To Play Back Any Document’s Keystrokes
JAMES SOMERS

The archaeology of great writing

Draftback lets you play back the revision history of any Google Doc you can edit. It's like going back in time to look over your own shoulder as you write. Notes:

- With Draftback, your data is kept entirely private. Draftback was purposely designed so that you could play back your own docs without having to share them with a third party. This is -your- data, Draftback just lets you see it in a new way.

- Draftback only needs access to docs.google.com to get the revision data for
Common mistakes from CSCI 4311 PA2:

Some test text

1a) Using available() to check for the end of a stream
Classes implementing available() return an “estimate” of the number of bytes that can be read without blocking. If this returns zero, it only means there is no data to read now, but there may be data to read in the future. Streams backed by sockets or some other source that buffers data often need more time for data to become available.

A more reliable indicator of the end of a stream is a return value of -1 from a read() method, or null from readLine(). This is the only guarantee that no more data will be available, aside from an IOException for some other reason.

1b) Using available() to check whether an input stream has been redirected
Unfortunately, there does not seem to be a portable way to check whether stdin has been redirected in Java. In C, the “isatty” function will let you check whether stdin or stdout is open in a terminal.
Common mistakes from CSCI 4311 PA2:

1a) Using available() to check for the end of a stream
Classes implementing available() return an “estimate” of the number of bytes that can be read without blocking. If this returns zero, it only means there is no data to read now, but there may
Common issues from CSCI 4311 PA1:

Using line-buffered I/O.
BufferedReader.readLine() “A line is considered to be terminated by any one of a line feed (\n), a carriage return (\r), or a carriage return followed immediately by a linefeed.”

For text data, this is generally not a problem, as long as platform-independent linebreaks are added to output, e.g. with BufferedWriter.newLine().
behind the scenes

≡ clearly, gDocs stores *everything* you did!
≡ why?

‡ why not?
  ~ bandwidth & storage on the house!
‡ user analytics
  ~ "if you are not paying ... you *are* the product"
‡ user convenience?
  ~ can you handle 10k revisions?
‡ programmer convenience?
  ~ works with the real-time collaboration concept
meet the changelog
"chunkedSnapshot": [
  
  
  "ty": "is","s": "Test document" ,"ibi": 1,
  
  {"ty": "as","sm": {"hs_hl": {"sdef_ts": {"ts_fs": 18.0 ,"ts_fs_i": false} }, 

  "as": "lgs_l": "en"},"ei": 0 ,"st": "language","fm": false,"si": 0},

embedded images

≡ upon upload
  ‡ a temporary Google CDN link is provided (googleusercontent.com)
  ~ lasts ~1 hour
  ‡ a permanent CDN link is also provided
≡ CDN link is like a dead drop
  ‡ if you know the address, you can access it
  ~ no authentication
what happens if we delete an image:
a) CDN image is deleted and becomes unrecoverable, or
b) it’s kept around, in case the change is rolled back?

well, b) of course!
as long as any document revision references the image, the public CDN link will remain live!
if the whole document is deleted, embeds are garbage collected
... after ~1 hour ...
Q: What happens when we revert to a prior version?

A:

‡ a snapshot of the desired revision is created,
‡ a “revert” entry w/ the snapshot is appended to the log.
observations & challenges [1]

≡ changelogs cannot be spoiled
   † Google will only **add** things to the log; no way to permanently modify prior state

≡ **the golden CDN hour**
   † could recover "SWAT-triggered" deletions

≡ reverse engineering is still critical
   † but emphasis will shift to protocols

≡ how representative is gDocs?
   † somewhat, at least
collecting changelogs is easy ...

storing & replaying them, much less so:

- what format should they be in?
- how do you render them (years from now)?

~ changelog is an internal data structure, it can change at any time
how to edit-share a doc without sharing the history?

- you cannot \(\rightarrow\) the history is the document

workaround
- create a copy (zaps the history) and share that

privacy audit
- extract all "deleted" embeds \(\rightarrow\) make sure you still need them

CDN link as a dead man's switch
- remember the librarian ...
new problem formulation
  † requires different methods & tools

gDocs artifact & behavior analysis
  † *Documents & Slides*
  † protocol documentation

PoC development: kumodocs
  † github: kumofx/kumodocs (coming soon)
Questions?

vassil@roussev.net