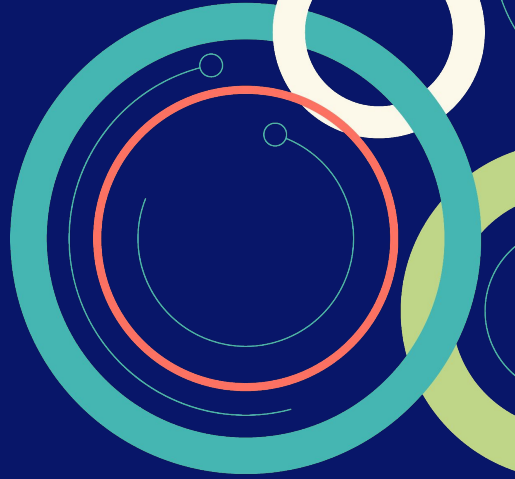


Katie Kodes

Secure by design

Rethinking test credentials for synthetic monitoring

Run check

April 30, 2026 at 10:00:02 PM | Columbus, OH, USA

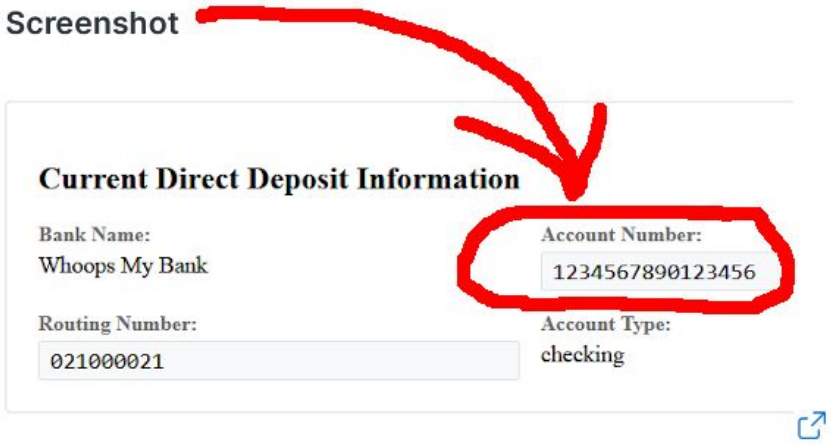
Failed	1	3.26 s	Chrome	Desktop
Result	Resources with errors	Execution Duration	Browser Type	Device Type

Error details

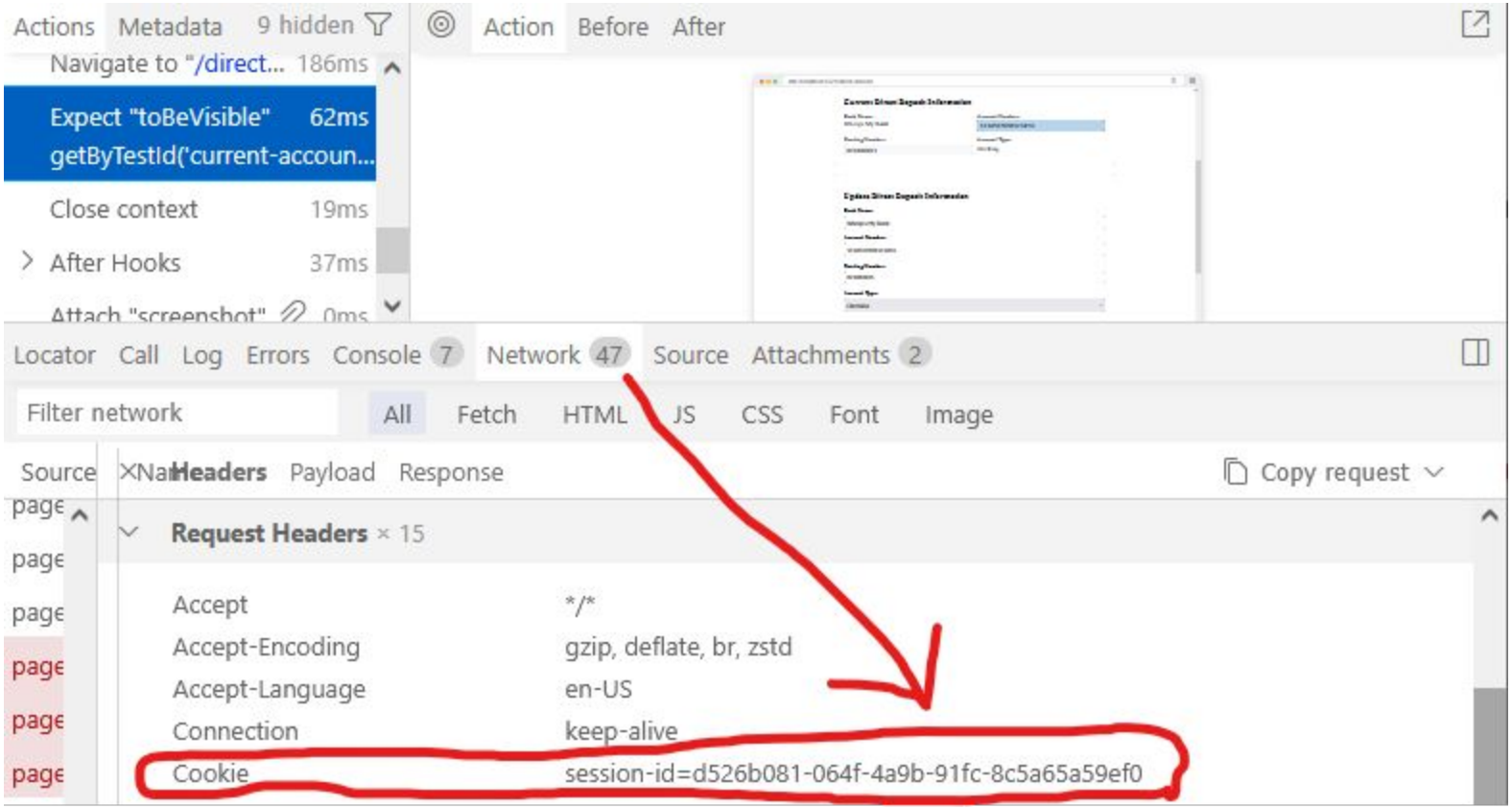
Check failed with assertion error

Error More information link did not match

Screenshot



Waterfall | Script Log (9) | Browser Log (1)


Actions
Metadata
9 hidden
Navigate to "/direct... 186ms
Expect "toBeVisible" 62ms
getByTestId('current-accoun...
Close context 19ms
After Hooks 37ms
Action
Before
After
Locator
Call
Log
Errors
Console 7
Network 47
Source
Attachments 2
Filter network
All
Fetch
HTML
JS
CSS
Font
Image
Source
Headers
Payload
Response
Copy request
Request Headers × 15
Accept */*
Accept-Encoding gzip, deflate, br, zstd
Accept-Language en-US
Connection keep-alive
Cookie session-id=d526b081-064f-4a9b-91fc-8c5a65a59ef0

What is synthetic testing / monitoring?

Synthetic testing / monitoring

Repeated imitations of real user behavior, just to validate whether things still work.

Very limited test suite, focused just on site reliability engineering / uptime / observability.

Typically against “prod” environment

Related: End-to-end (“E2E”) testing

Usually just run once right after feature/fix release

Much broader test suite

Often, but not always, against nonprod environment

For web applications, both typically use a web **browser “driver”** (the same technology used in “screen scraping”)

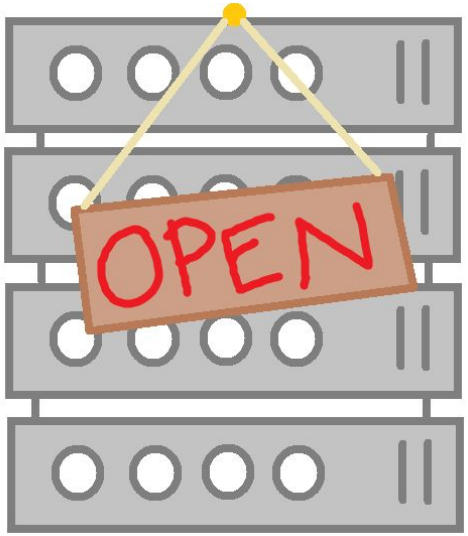
3 vulnerabilities inherent to automating login

Observability Summit North America

Log Reliance

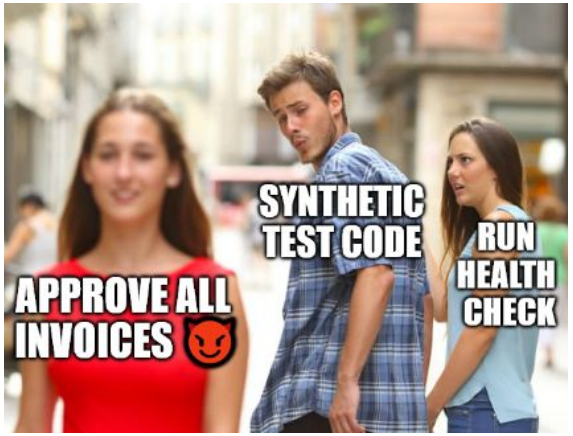
Logs (e.g. network), Screenshots, Videos can leak:

- sensitive data
- credentials (e.g. cookies)



Exfiltration goes 24/7

Whatever happens, is likelier to happen at 3AM



24/7 Reprogrammability

Is your app's status quo "least privilege" still WAY more than is needed for automations to validate whether the system is "working," because **accounts** were presumed human?

4 layers of mitigation

Vocab 101: authN vs. authZ

Authe**N**tication (authN)

"Verifying the **identity** of a user, process, or machine."

("Are you really **who** you say you are?")

e.g. HTTPS 401 error

Authori**Z**ation (authZ)

"When one entity **grants permission** to another entity to engage with a resource, within a set of boundaries."

("Can you really do **what** you say you can?")

(authZ checks typically not reached if authN fails – no point in checking details if known intruder.)

e.g. HTTPS 403 error

(from devopsdaysMSP 2024 - Jenessa Petersen: Back to the Basics: Authentication and Authorization 101)

Mitigation 1: identity separation (authN)

1. Don’t reuse human accounts for automated testing.
2. One account per tool.
3. Maybe even one account per test-suite purpose per tool.

Mitigation 2: find implicit authZ surprises

Did you know default MS Active Directory lets every user see every other user?

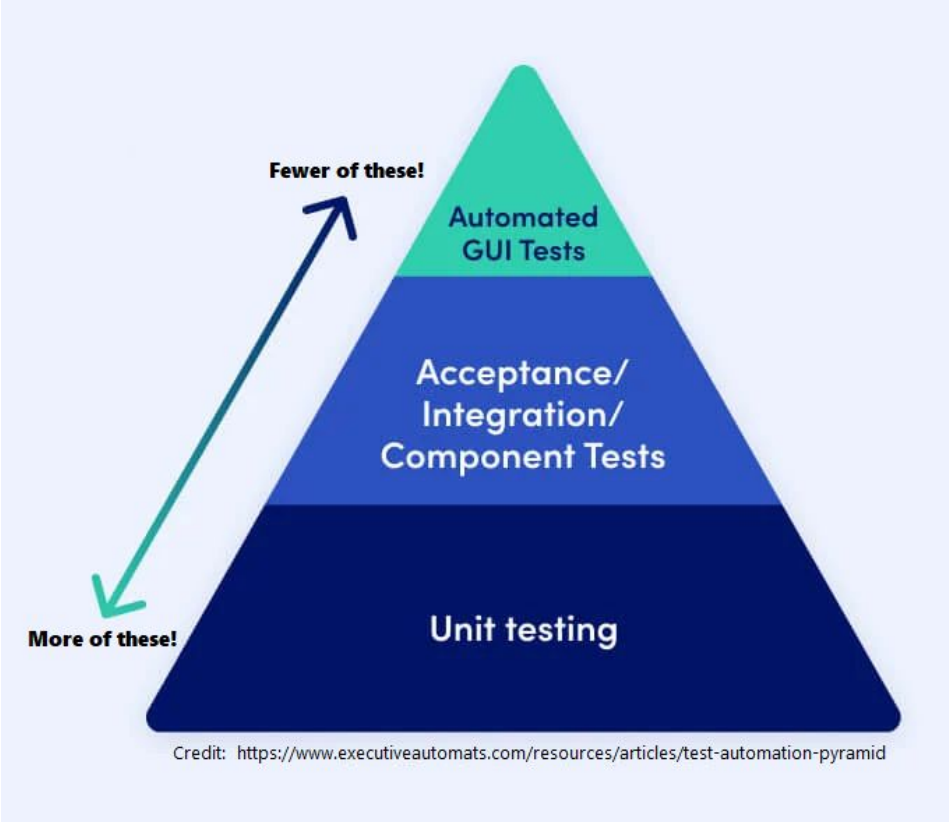
(Your identity team knew.)

So buddy up with other departments and ask!

Mitigation 3: rearrange test cases (authN)

Heavily using "**mocks**," test these **lower** down the test pyramid:

1. Things that aren't about "uptime" vs. "downtime"
2. Things unrelated to login
3. The **five senses** (e.g. accessibility, layout, color)



Mitigation 4: hide / replace data (authZ)

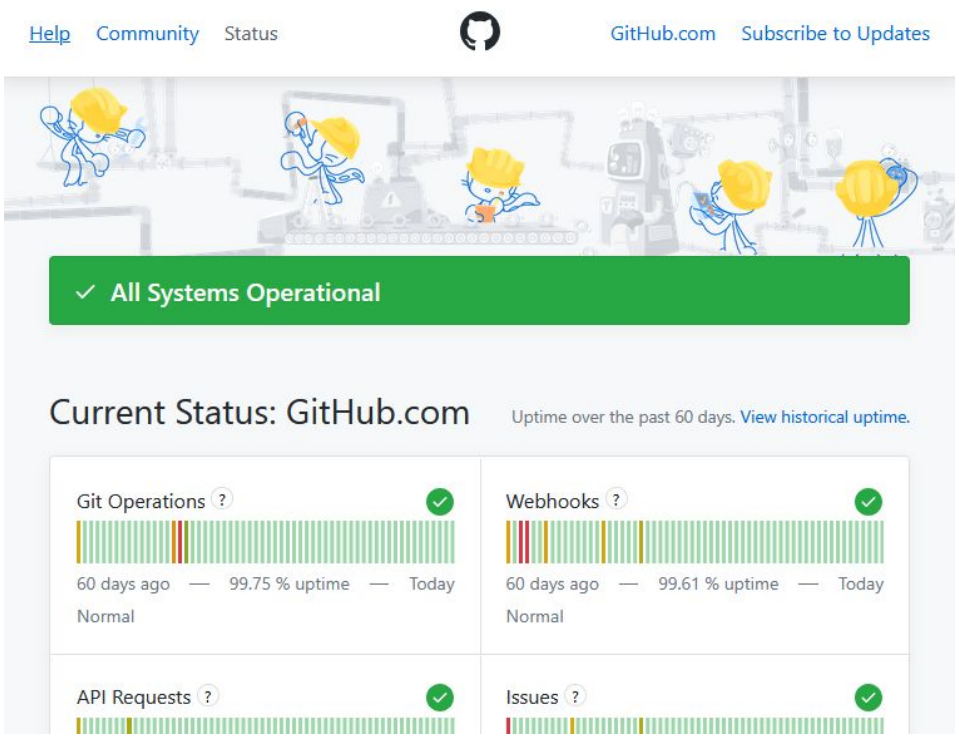
1. Don't give test accounts access to anything but a **"health check" page** (even better: no-auth needed, like githubstatus.com)
   Caveat: hard if vendor app

2. Only let test accounts see **"synthetic" data**
   Caveat: managing data hard

Mitigation 4-A: hiding data (authZ)

Don’t give test accounts access to anything but a **“health check” page** (even better: no-auth needed, like githubstatus.com)

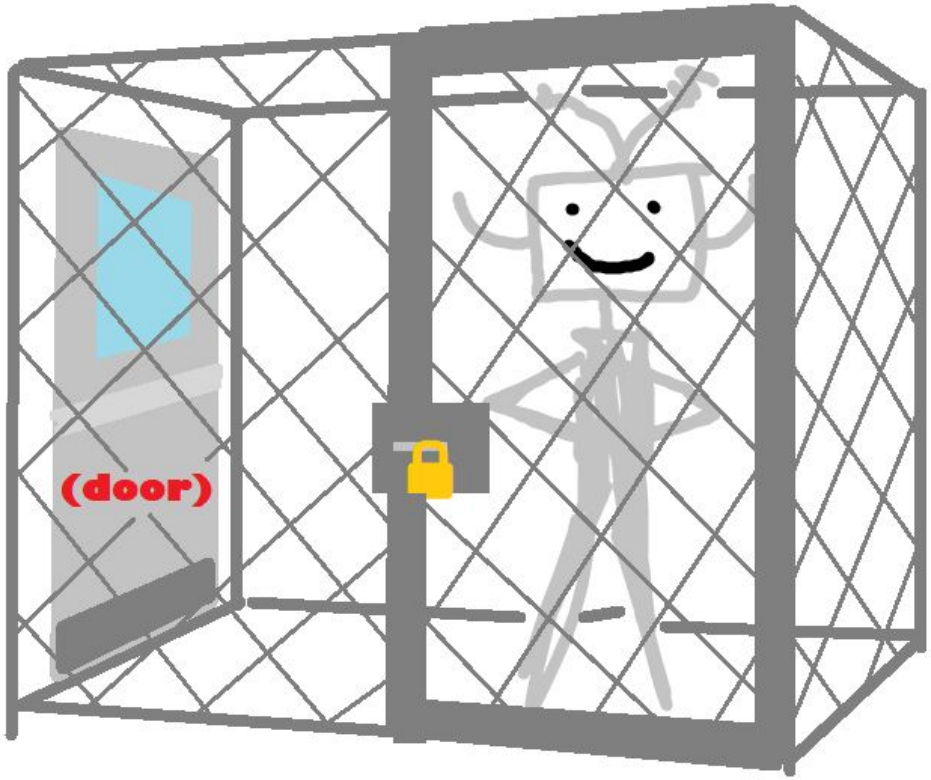
Caveat: hard if vendor app



Mitigation 4-A: hiding data (authZ)

Don’t give test accounts access to anything but a **“health check” page** (even better: no-auth needed, like githubstatus.com)

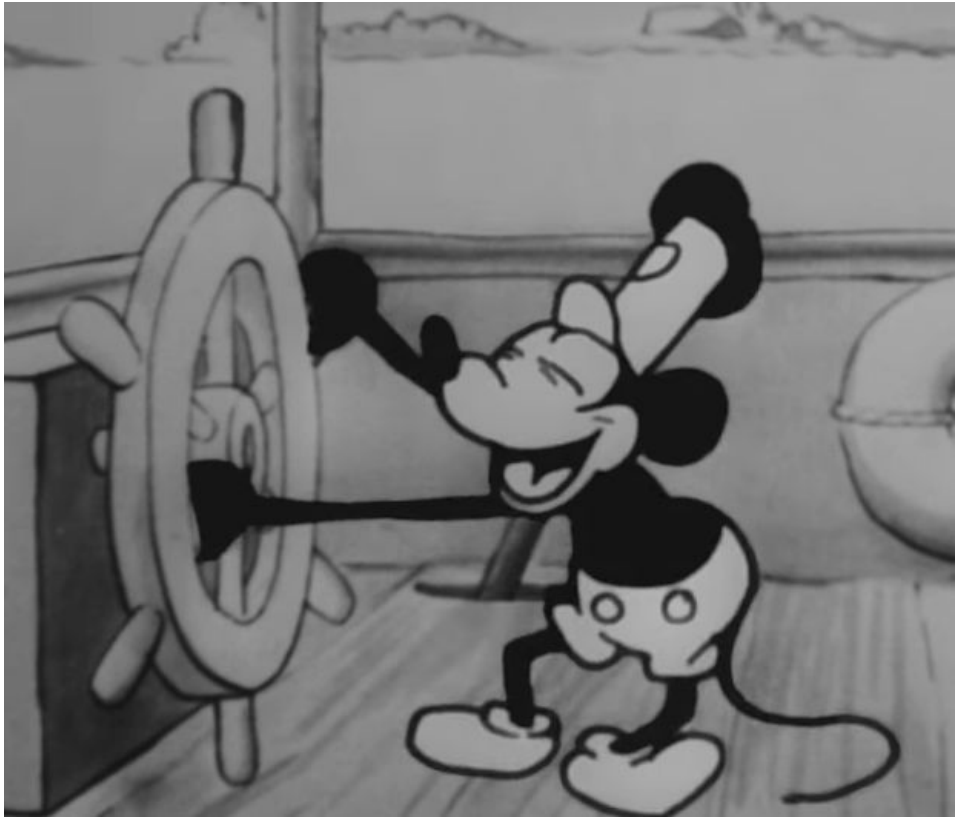
Caveat: hard if vendor app



Mitigation 4-B: synthetic data (authZ)

Only let test accounts see **“synthetic” data**

Caveat: managing data hard



Security needs everybody

Product team: architects, develops, and trains on real vs. synthetic data handling

Security: educates and reviews re: authentication and authorization

Frontend developers: special role as “how browsers work” subject matter experts

Observability: governs platform synthetic test results centralization platform (and maybe synthetic testing tool use)

Observability Summit North America

Thank you!

https://katiekodes.com/**o11ysummit-2026**