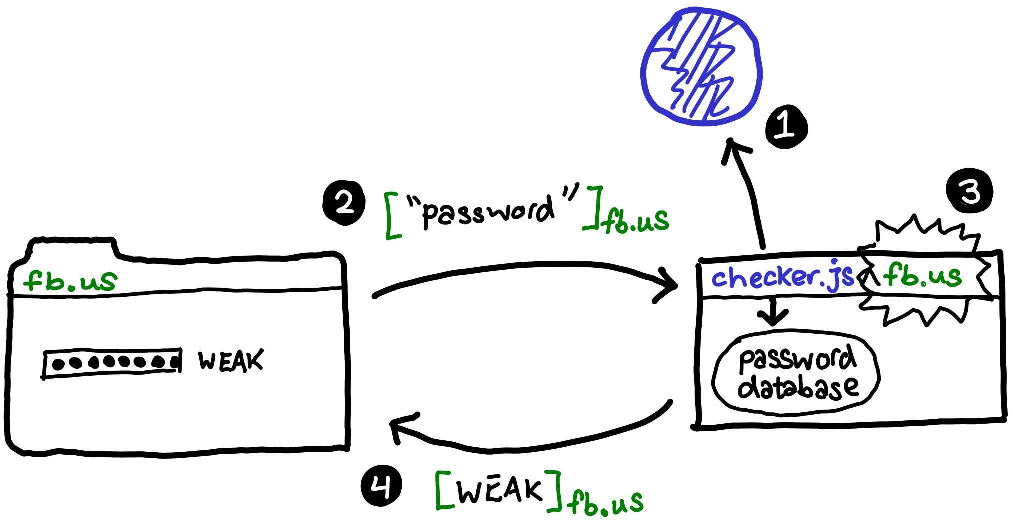
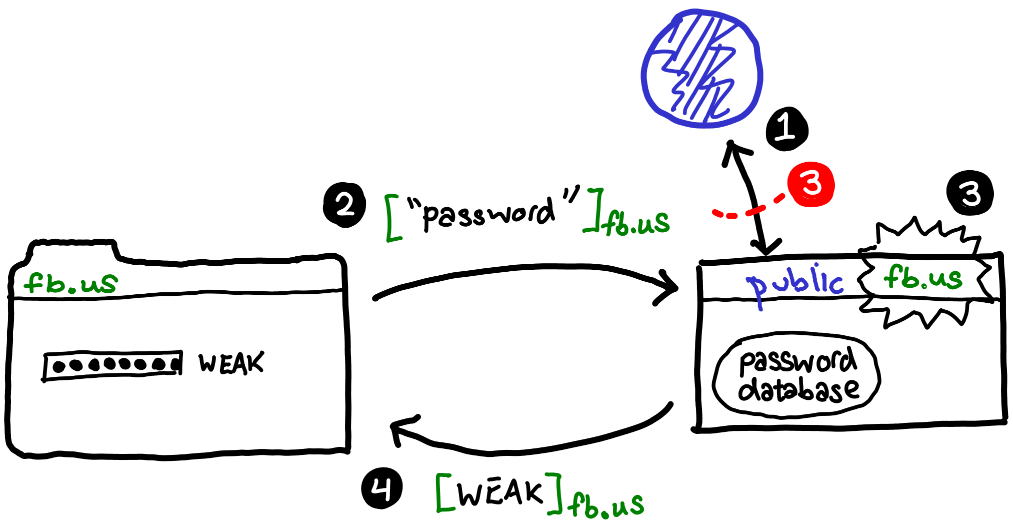


Figure 1. Password checker

This diagram shows the organization of an untrusted password checker script. The difficulty of this example is that we would like the checker script to be able to download a database of weak passwords without being able to leak the contents of the password being checked. Execution proceeds as follows: (1) the checker script starts off untainted (it's label being checker.js) and downloads the password database from the web. Next, (2) fb.us sends the password checker a labeled password ("password"), which (3) checker.js must raise its label in order to be able to read. Once its label is raised, it can no longer talk to the internet (thus, step (1) must be carried out first!) It computes the strength of the password, and (4) sends back a labeled result to the original page to be displayed. (Note: checker.js has to declassify the result so that it doesn't also have a checker.js label!)


1
2
["password"]fb.us
3
3
fb.us
WEAK
public
fb.us
password database
4
[WEAK]fb.us