DISCUSSION 5/13

WEB SECURITY CONTINUED

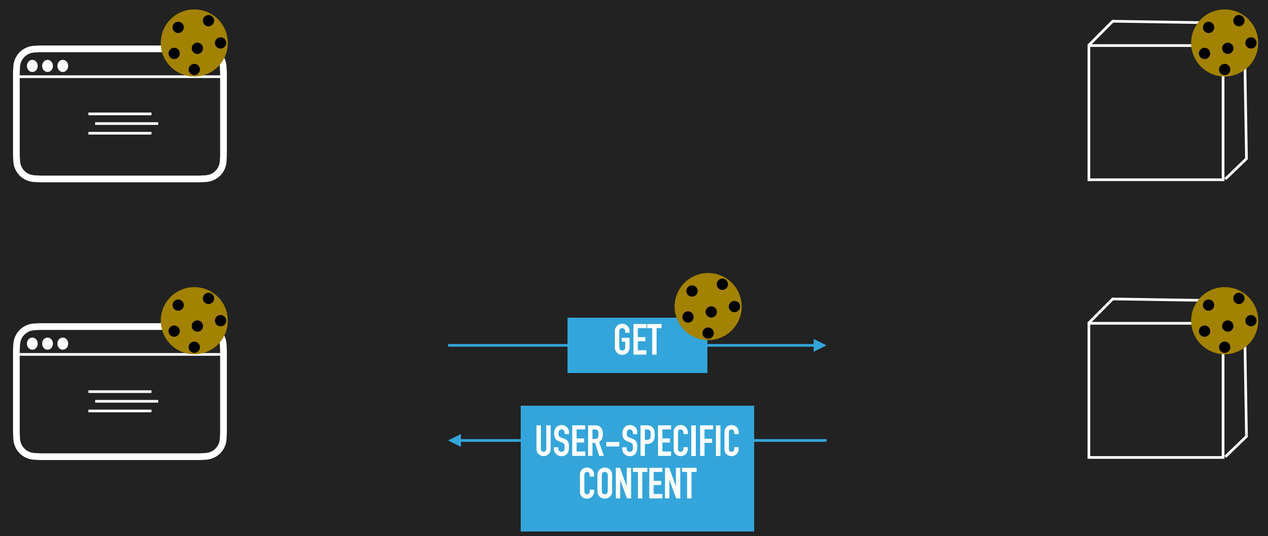
WHO WAS I TALKING TO AGAIN?

- HTTP is a stateless protocol
  - no notion of session
  - would like to customize interaction based on user
    - e.g. shopping cart, user preferences, access

- browser and server establish shared secret in the form of a cookie
- cookie uniquely distinguishes browser-server conversations

HOW DOES THAT HELP?

- browser and server store a copy of the cookie
- browser attaches cookie to any messages it sends to server
- server establishes requestor's identity by looking up cookie

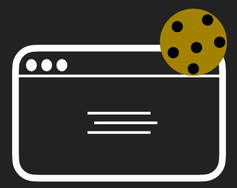


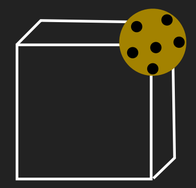
WAIT WHO GETS MY COOKIES?

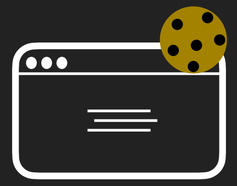
- Cookie Same Origin Policy: ([scheme], domain, path)

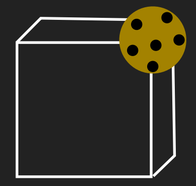
scheme://domain:port/path?params

- Browser sends all cookies which match:
  - scheme (https if cookie marked secure)
  - domain suffix
  - path prefix









CROSS SITE REQUEST FORGERY

Key Features Enabling Exploit:

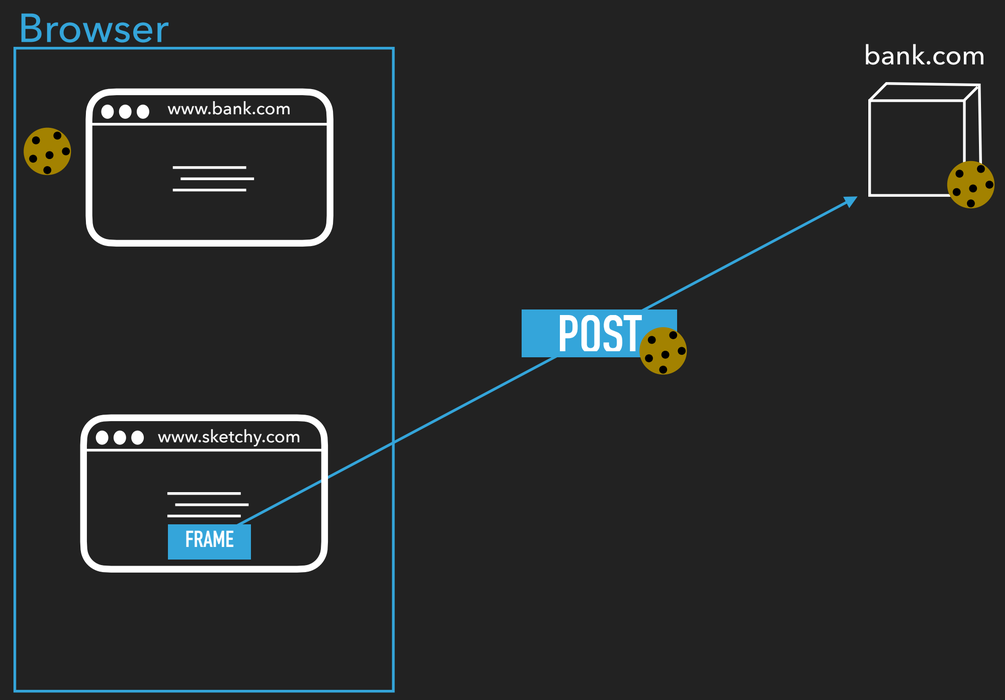
- Cookies stored in the browser (i.e. shared across tabs/windows)
- sent based on request destination not source

Scenario:

- Bob opens browser and goes to bank.com for some important business
- browser stores bank.com cookie
- Bob opens tab and goes to sketchy.com
- sketchy.com contains request to bank.com

Mitigations:

- Secret Token Validation
- Referer/Origin Validation
- SameSite cookies



CROSS SITE SCRIPTING

Key Features Motivating Exploit:

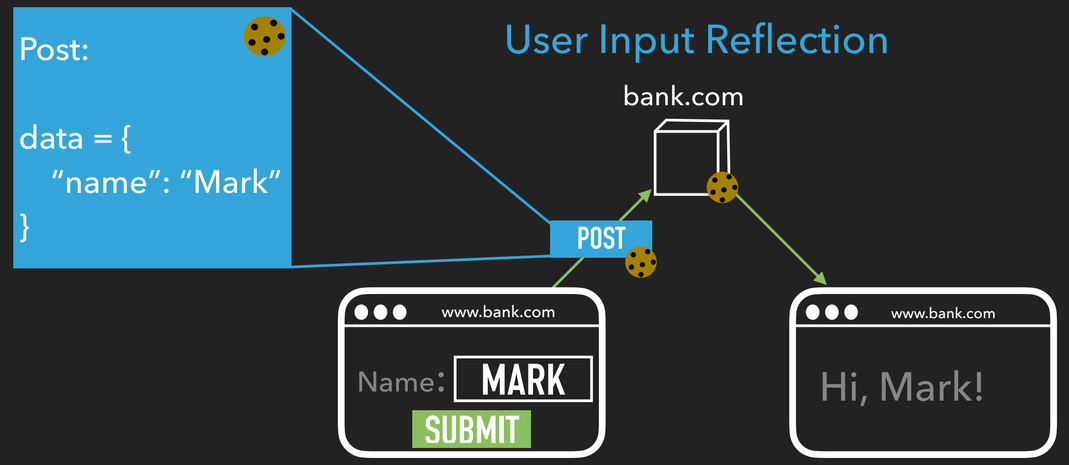
- SOP for DOM prevents sketchy.com from accessing bank.com cookie
- sketchy.com can add a bank.com frame which can access the cookie
  - cannot inspect embedded frame

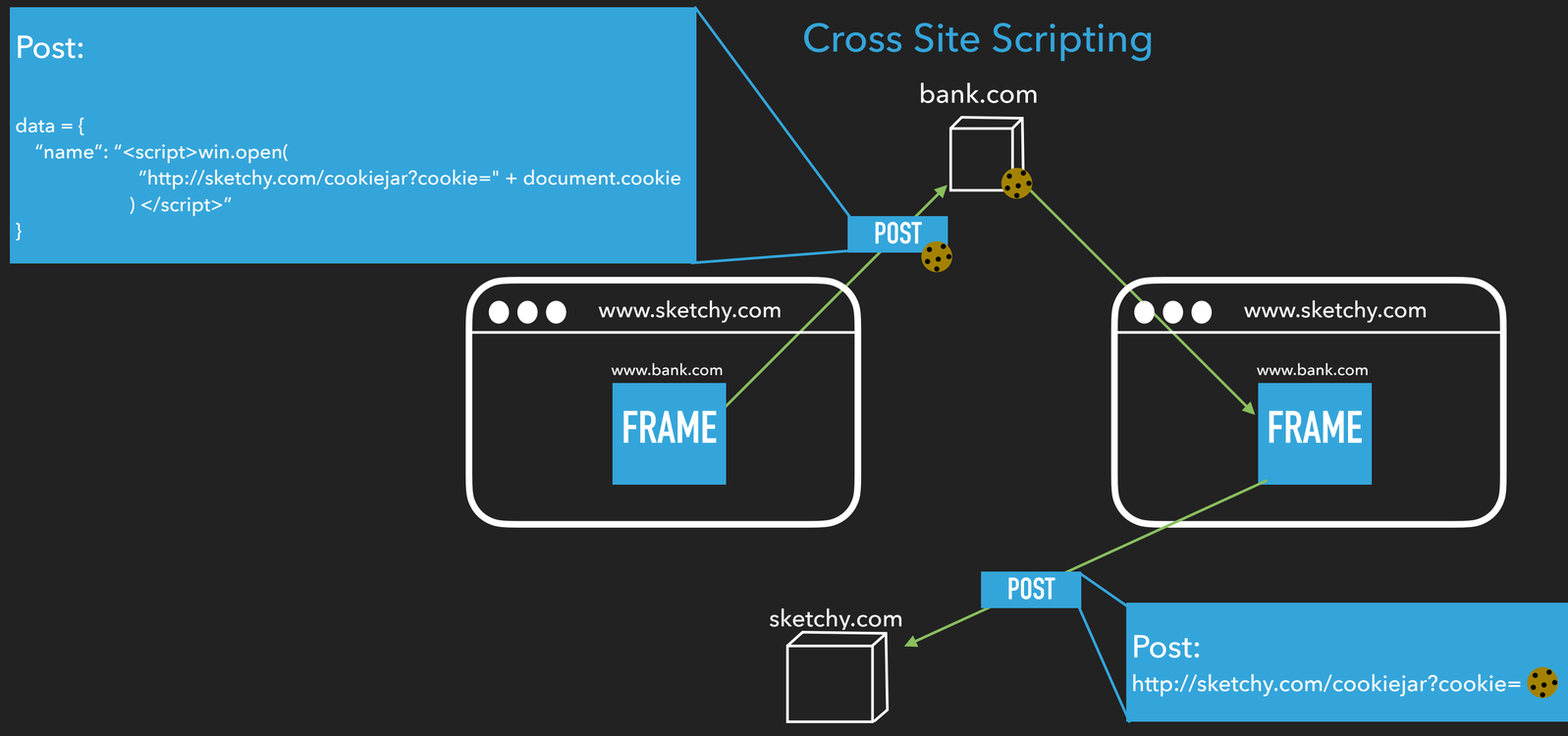
Key Features Enabling Exploit:

- Website reflects user input in response
  - e.g. bank.com has a form with name field

Scenario:

- sketchy.com sees bank.com reflects user input
- sketchy.com embeds frame posting to bank.com with malicious script
- bank.com reflects malicious script back to embedded frame
- embedded frame executes malicious script with privilege of bank.com





SUMMARY

Cross Site Request Forgery

- Target: attach user's cookie to malicious request
- Method:
    1. malicious site embeds frame pointing to target site
    2. browser attaches target site's cookie to malicious request

Cross Site Scripting

- Target: inject malicious code into pages served by target site (executed in user's browser)
- Method (one flavor):
    1. malicious site identifies target site which reflects user input
    2. malicious site embeds frame posting malicious script to target site
    3. target site reflects malicious script
    4. malicious script executes in user's browser with privileges of target site

SQL Injection

- Target: execute malicious SQL on target site's database
- Method:
    1. Post malicious input to target site
    2. unsanitized malicious SQL statements executed on target site's servers