Flag Gearup

Goals

Learn about web security by attacking a broken, unknown website:

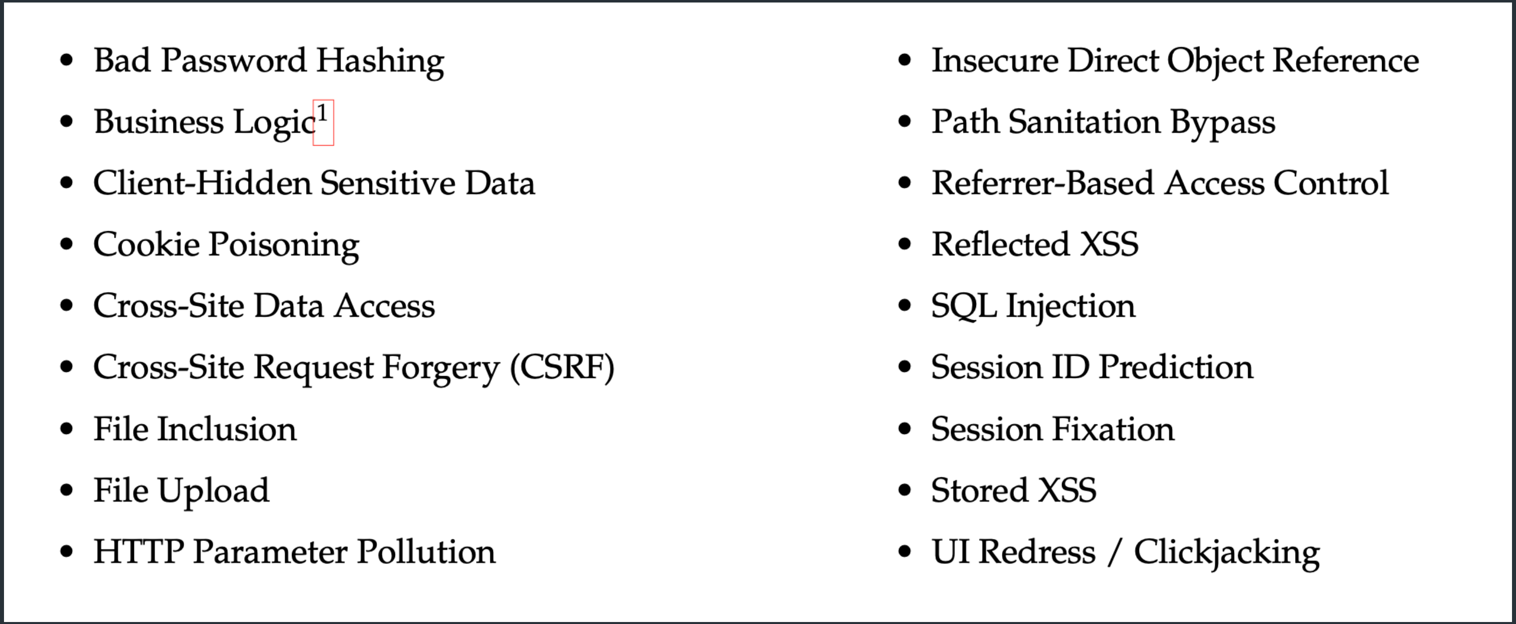
- Poke around the site to figure out how it works
  - You <u>don't</u> access to the code! Learn about the system by testing
- … then break it!
- After that, write *vulnerability reports* about each vulnerability

- <u>CS1620/CS2660</u>: Additional, multi-step attack: Bob's Router

The assignment

FOR MORE INFO,
SEE LECTURES 7-11

- Find and write up at least five (5) vulnerabilities
- Each must be from a distinct *vulnerability category*
  - Can't count the same category more than once



- Bad Password Hashing
- Business Logic[1]
- Client-Hidden Sensitive Data
- Cookie Poisoning
- Cross-Site Data Access
- Cross-Site Request Forgery (CSRF)
- File Inclusion
- File Upload
- HTTP Parameter Pollution
- Insecure Direct Object Reference
- Path Sanitation Bypass
- Referrer-Based Access Control
- Reflected XSS
- SQL Injection
- Session ID Prediction
- Session Fixation
- Stored XSS
- UI Redress / Clickjacking

Haven't heard of some of these before? That's okay!
=> We provide resources to help

The Wiki

We've provided each wiki that explains each vulnerability in detail

- Find it here: https://brown-csci1660.github.io/flag-wiki/

Use the wiki to...

- Learn about each type of attack and how it works
- See "Criteria for Demonstration" => what you need to show us to count as a vulnerability
- Find more references for further reading

Feel free to search for more resources online too!

How you'll work on the project

(From this point forward, we'll call the container you used in the last project your "development container")

- "Flag portal container": download a container on your system
  - Similar to dev environment from Project 1
  - Hosts website for you to attack

How you'll work on the project

- "Flag portal container": download a container on your system
  - Similar to dev environment from Project 1
  - Hosts website for you to attack
- Use (almost) any other tools on your computer
  - "Developer tools" in your browser (Firefox <u>highly</u> recommended)
  - Your dev container from Project 1 (for Linux tools, running scripts, etc.)
  - Burp suite
  - Anything else so long as it doesn't automatically find vulnerabilities for you

You won't be writing a lot of code—most of your time will be trying out things, maybe making small code snippets/scripts, etc.

How to get started

Project setup guide:
https://hackmd.io/@cs1660/flag-setup-guide

What's in this guide

- How to clone/set up the flag container
- Helpful resources if things go wrong with the containers
- More tutorials and resources

About the container environments

- Flag uses a new container, separate from your "dev container" from project 1
  - Bob's router has one more container
- Each container has a `run-container` script just like the one for your dev container

Important container terminology

- Container image ("image"): read-only package of the files/settings for how the container runs

  ⇒ YOU DOWNLOAD FROM US.

- Container instance ("container"): created when container started, read-write

  ⇒ CREATED WHEN YOU RUN ./RUN-CONTAINER

  → YOUR CHANGES LIVE HERE!

=> At any point, you can "reset" the container's state by discarding the container instance by running:

```
./run-container --clean
```

Demo: Container setup

How to get started

- Don't panic. You can do this.
- Play around with the site
  - What can the user do?
  - How might each site feature work? ⇒ HOW CAN YOU CONTROL IT TO YOUR ADVANTAGE?

How to get started

- Don’t panic. You can do this.

- Play around with the site
  - What can the user do?
  - How might each site feature work?
- See lectures 7-10, wiki for examples of attacks!
  - Lectures are starting point, wiki has more details

See the “Web lecture demos” for some example code from lectures

Online resources

You're welcome to look for guidance and examples online (and collaborate with your peers), so long as the attacks you carry out are your own work

All of these are fine:

- *"How do I submit a form in javascript?"*
- *"How do I write a file in PHP?"*
- *"What are some ways to defeat input sanitization?"*

We want you to learn to use online tools to help you solve problems!

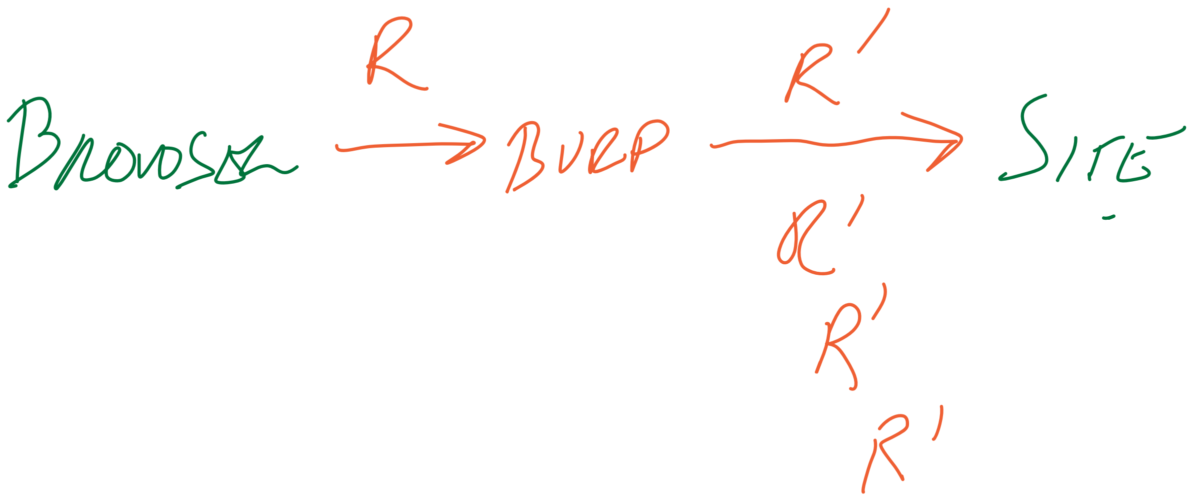
Tools and resources

- Web lecture demos: demo website and attack payloads from lecture
- webhook.site: A temporary URL that logs any requests
  - Can use as a target for CSRF attacks, etc.
  - Alternate version: listening on a port with netcat
- Simple webserver: a tiny container to host arbitrary pages
  - Need to have your target download something? Use this!
- PHP reverse shell: see lecture 9 (Tuesday, Feb 27)

For detailed instructions, see the setup guide!

Q: What's the difference between Burp and something like a webhook?

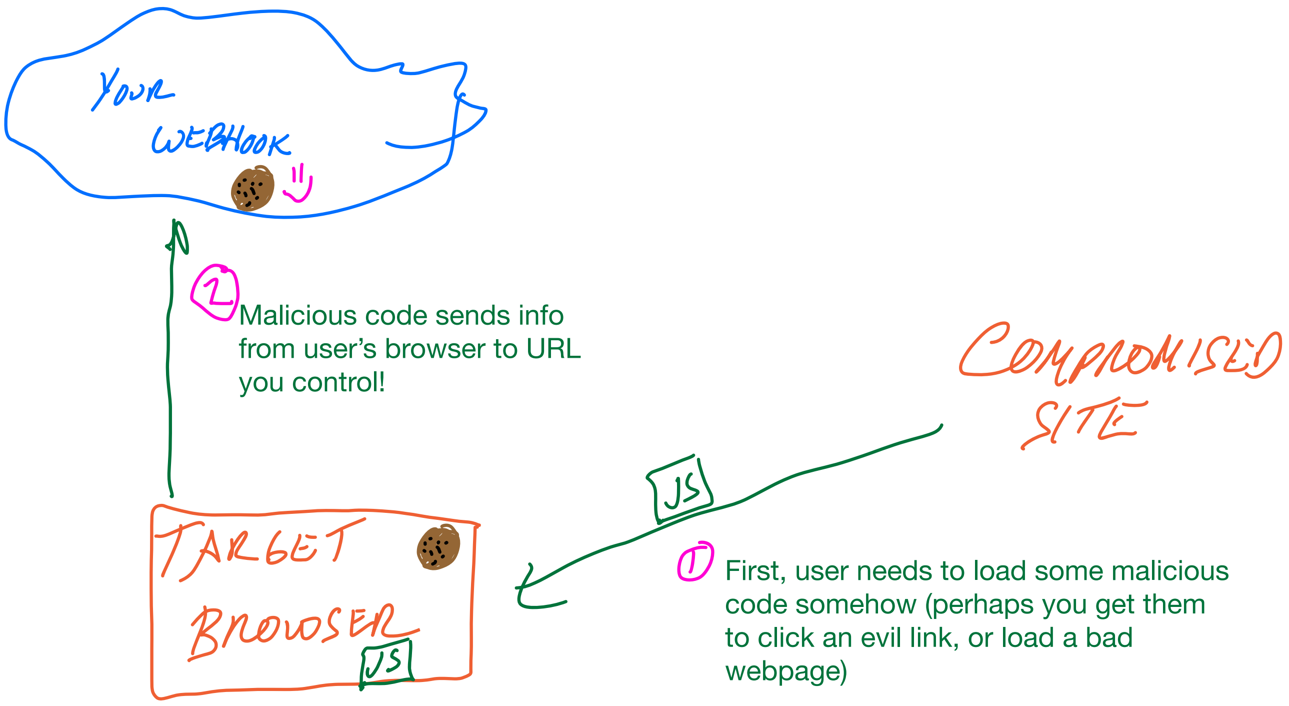
Burp is a request proxy: you can use it to
- Log requests you've sent as you explore the website
- Intercept and modify requests before they go to the site
- Craft arbitrary requests based on previous ones (independent of any client-side controls)



Tools like webhook.site are for a much more specific use case—usually, you'd use it for carrying out a CSRF attack (most probably for Bob's router)

For certain kinds of attacks (usually CSRF attacks), you might be trying to steal info from a user's browser (1, below). In this case, your attack will probably run some javascript that reads the user's info (eg. their cookies). Then, your exploit code needs to send that data somewhere (2) by making a web request to some site where you can read the output.

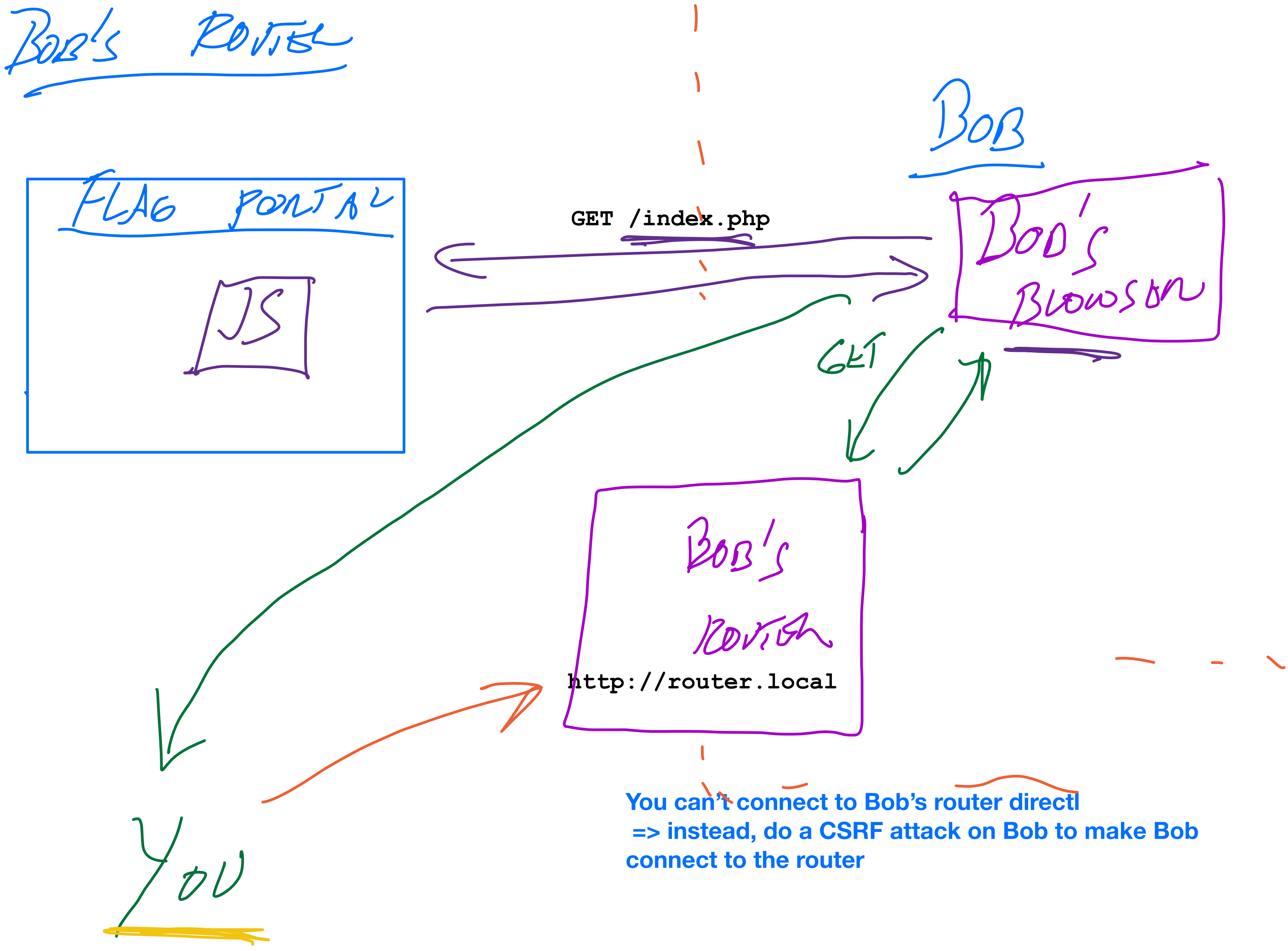
You could do this by setting up your own webserver, but this would be complicated. webhook.site is a free tool that gives you a URL and will log all requests sent to it—which is enough to show you the stolen data!



Malicious code sends info from user's browser to URL you control!

First, user needs to load some malicious code somehow (perhaps you get them to click an evil link, or load a bad webpage)

Demo: Bob’s Router



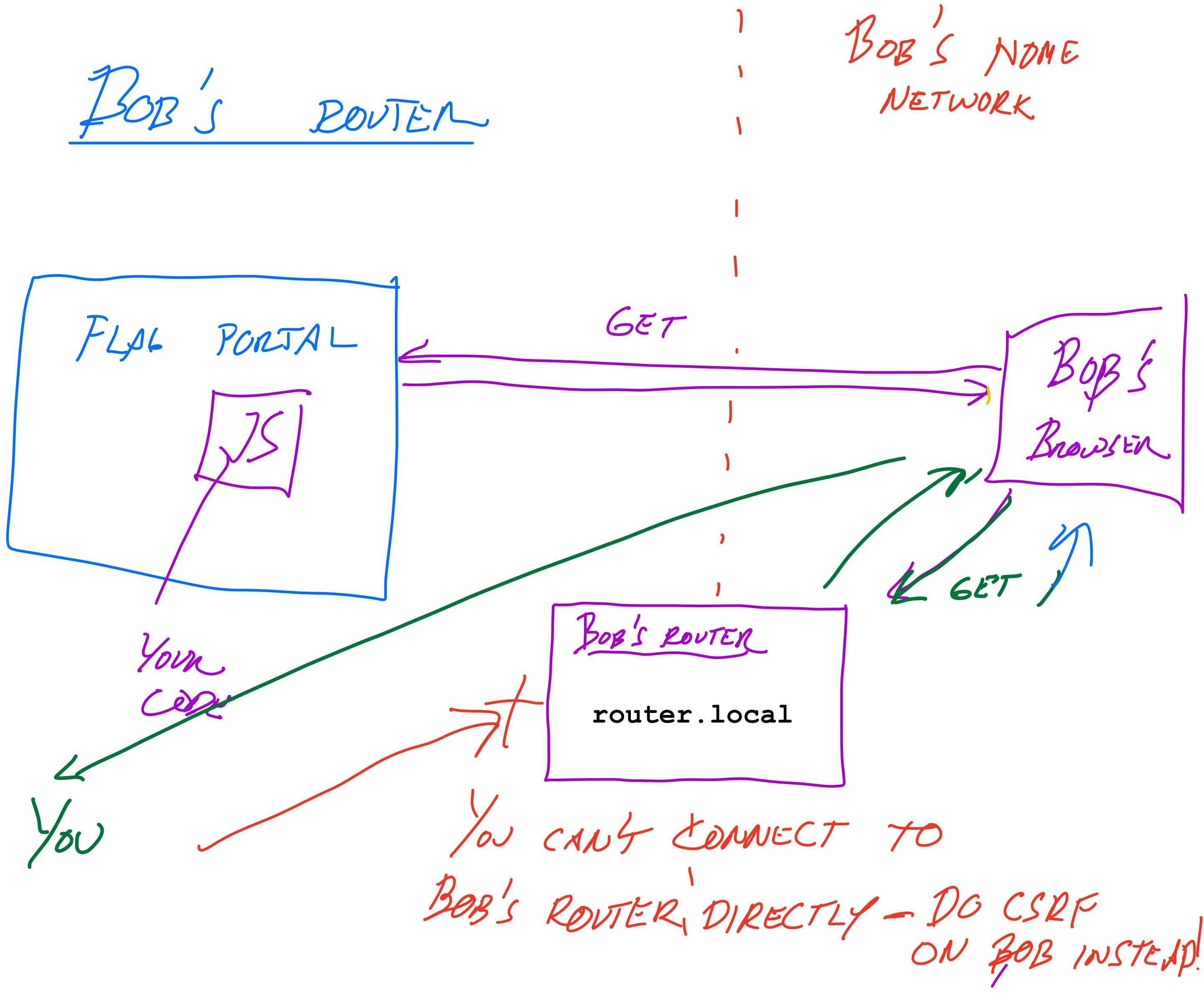
You can't connect to Bob's router directl
=> instead, do a CSRF attack on Bob to make Bob connect to the router

Goals

1. RUn arbitrary javascript on Bob's browser => to start: fetch main page of Bob's router (http://router.local)

2. Look at content of page that comes pack => learn another exploit you can run on the router to get it to run arbitrary PHP code
   => Run a "reverse shell"

3. Poke around on Bob's router to find the flag!



Goals

Step 1: Run arbitrary JS on Bob's router (CSRF atttack)
=> Starting point: try to fetch main page of Bob's router (http://router.local)

Step 2: The main page will be a login page => based on what you know about the router, try to log in!

Step 3: Based on the content of the router's page, you'll learn about an exploit you can run on the router to run arbitrary PHP code
=> Try to run a reverse shell on the router (more on this in class lectures 9-10)

3. Once you have a reverse shell on Bob's router, poke around the filesystem until you get the flag!

Tools that may be helpful for Bob's router

1. For CSRF attack: need to view contents of router's webpage

=> Send page contents to webhook.site, or netcat example

2. Get Bob/router/user to load your webpage

=> Use "simple webserver" to host file locally on your system

=> Look for more demos in upcoming lectures!