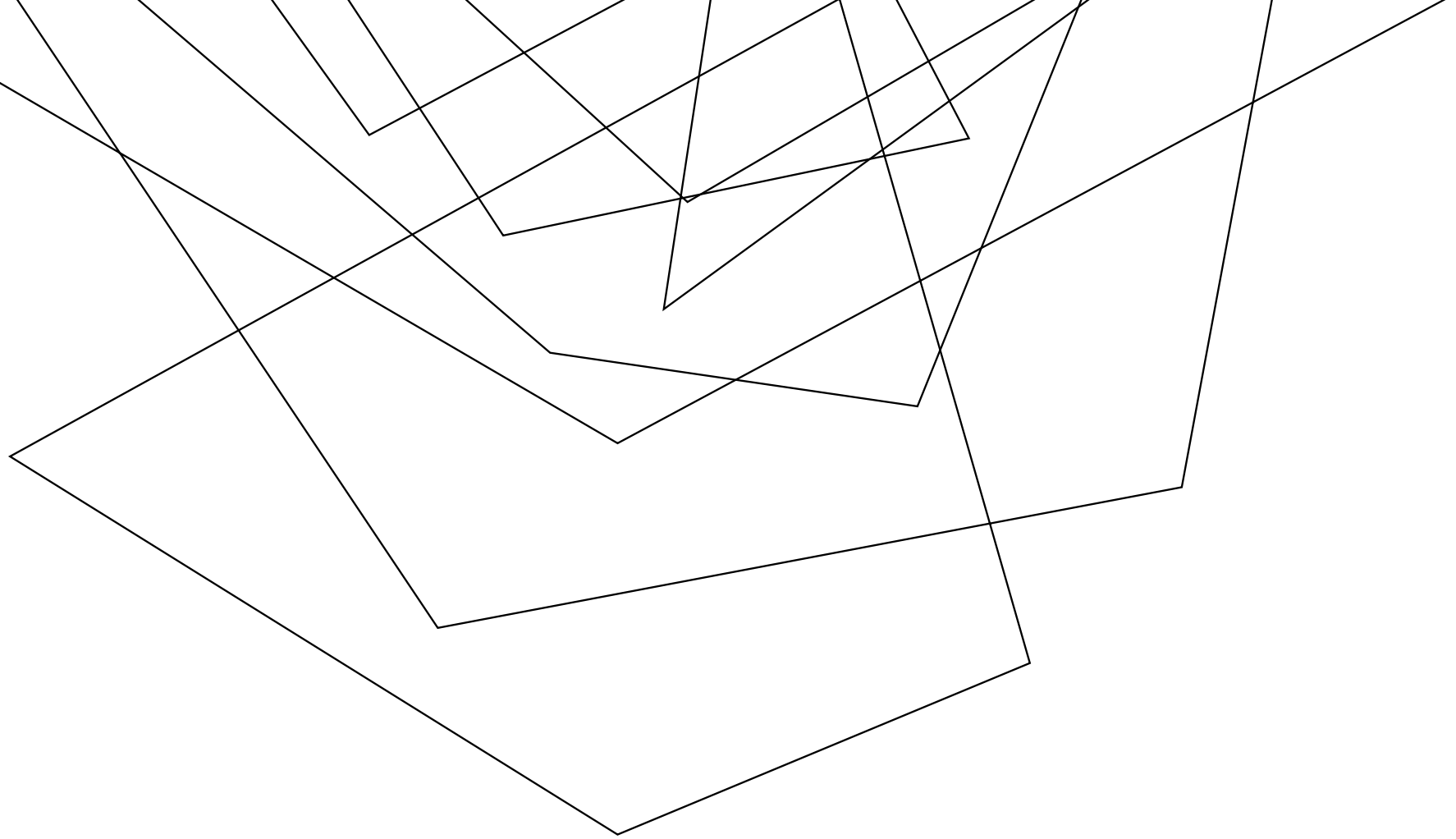
EXERCISE

SMT SOLVING REVIEW

Write your name and answer the following on a piece of paper

To what theory does EUF refer in SMT Solving? How is EUF useful?

ADMINISTRIVIA AND ANNOUNCEMENTS



WEB SECURITY

EECS 677: Software Security Evaluation

Drew Davidson

SHIFTING GEARS

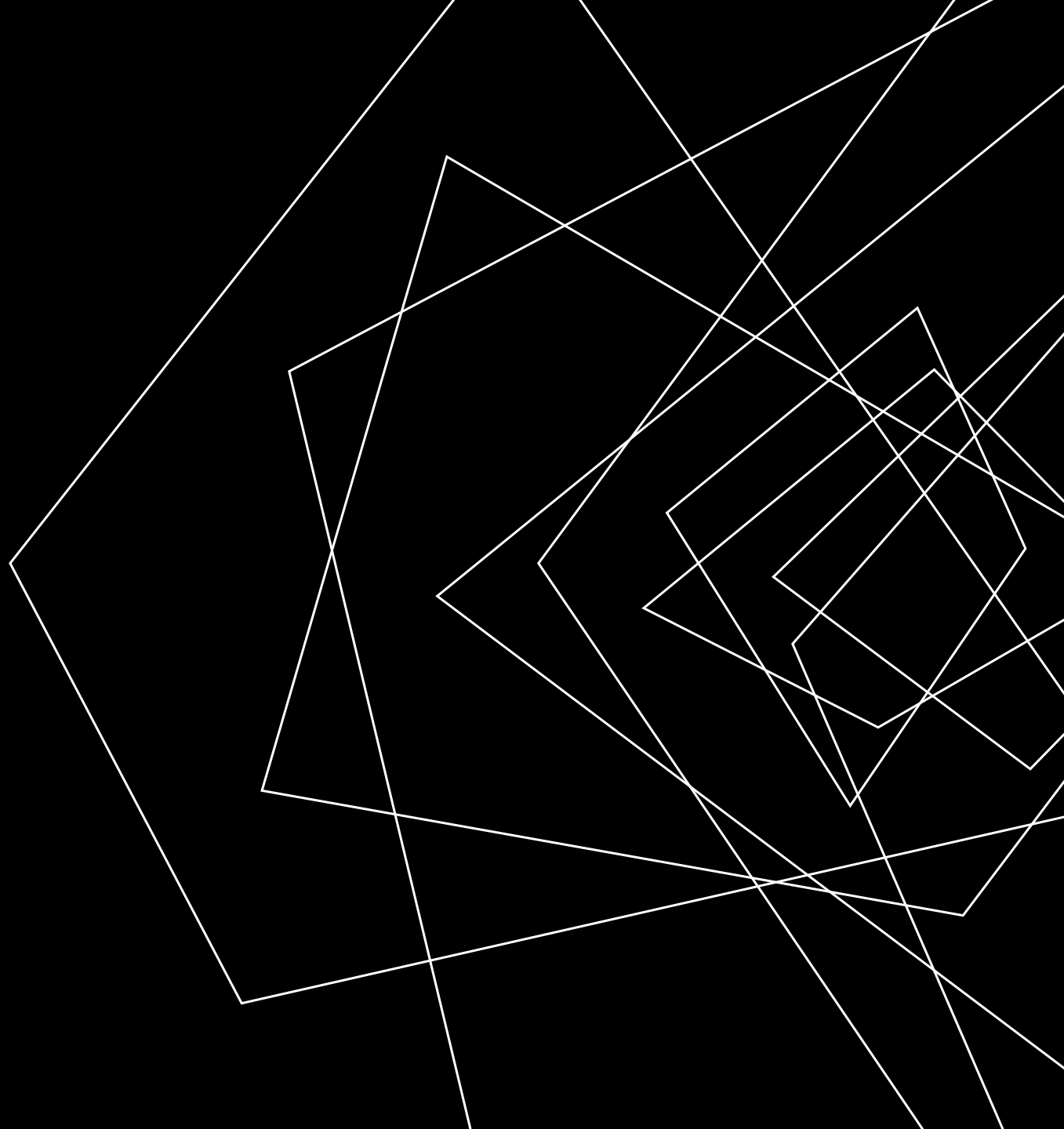
OVERVIEW

On to a different topic: web security!



LECTURE OUTLINE

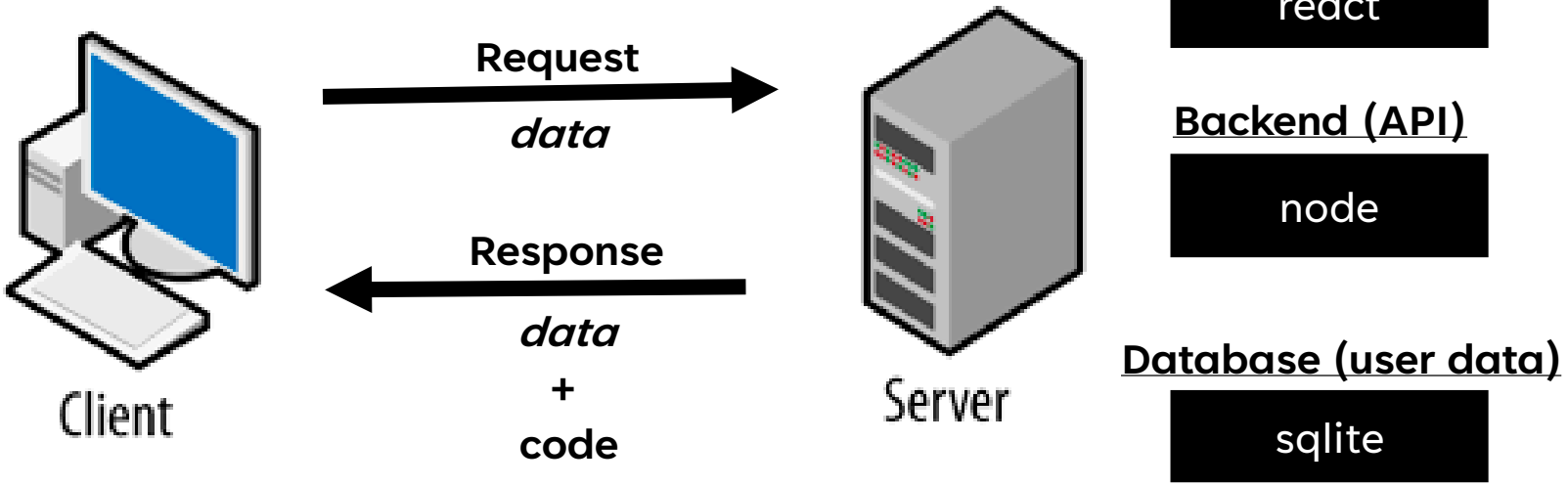
- Basics / Introduction
- Endpoint Security
- Resource Security



MODERN WEB DEPLOYMENTS

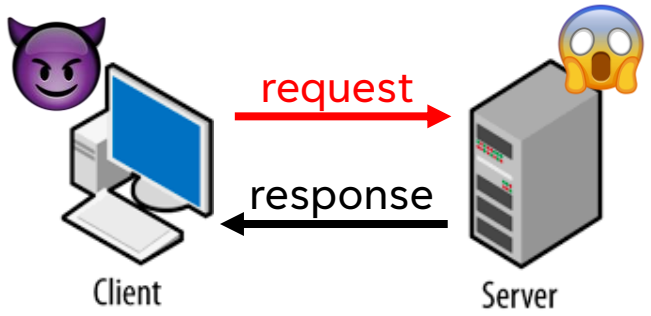
WEB SECURITY: BASICS / INTRODUCTION

Computers (largely) communicate in a client-server relationship



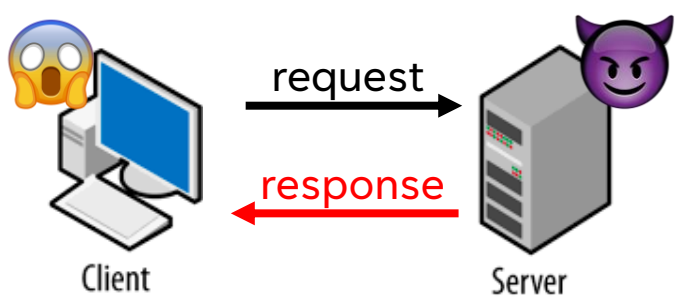
THREAT MODELS

WEB SECURITY: BASICS / INTRODUCTION



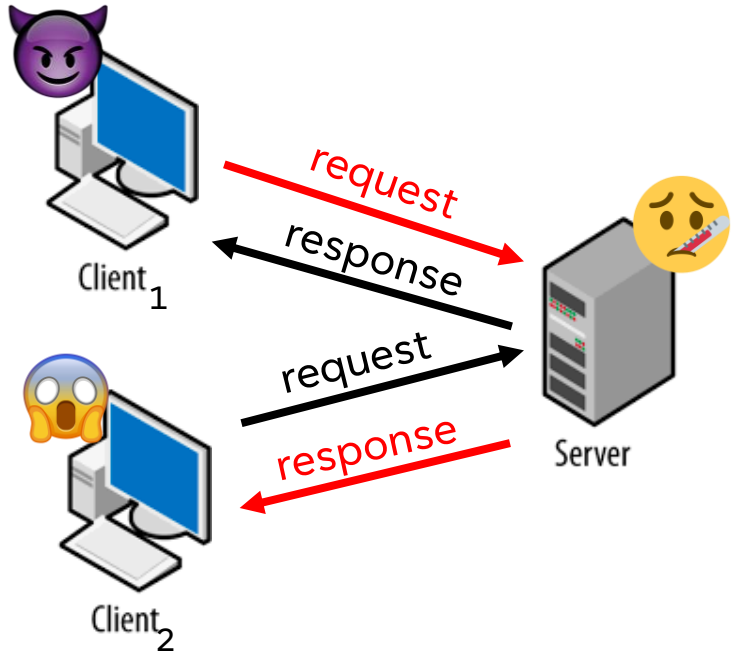
Malicious client

Victim server



Malicious server

Victim client



Malicious client 1

victim client 2, vector server

SETTING UP WEBSERVER TOOLING

WEB SECURITY: BASICS / INTRODUCTION

Installing node

Begin with nvm

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.41.0/install.sh | bash
```

Use nvm to get the latest npm

```
nvm install 26.0.0
```

Uninstalling

Delete tools

```
rm –rf ~/.nvm
```

Reverse any shell configuration

```
vim ~/.bashrc
(remove any line with $NVM_DIR in it)
```

SIMPLE SERVER

OVERVIEW

Set up a backend codebase with node

```
npm init endpoint ; cd endpoint
npm install express
vim index.mjs
```

Run the server

```
node index.mjs
```

```
import express from "express";
const app = express();

app.get("/page", (req, res) => {
  let v = "world";
  res.send(`<h1>Hello ${v}!</h1>`);
});
app.listen(3001);
```

SIMPLE SERVER
OVERVIEW

Set up a backend codebase with node

```
npm init endpoint ; cd endpoint
npm install express
vim index.mjs
```

Run the server

```
node index.mjs
```

```
import express from "express";
const app = express();

app.get("/page", (req, res) => {
 let v = req.query.v ? req.query.v : "world";
 res.send(`<h1>Hello ${v}!</h1>`);
});
app.listen(3001);
```

CODE AS CONTENT

OVERVIEW

What happens when a user gives CODE in a content field?



https://website.com?coolVar=<script>alert(‘hello’)</script>



print $_GET[‘coolVar’]

RELEVANT ATTACKS

OVERVIEW

Cross-Site Scripting Attacks

Injecting (malicious) client-side scripts into web pages

Cross-Site Request Forgery

Unauthorized commands are submitted on behalf of a trusting user

SANITIZATION

WEB PROTECTIONS: XSS

Example XSS was enabled by user data being treated as code

(special characters escaped the string)

```
import express from "express";
import validator from "validator";
const app = express();

app.get("/page", (req, res) => {
  let v = req.query.v ? req.query.v : "world";
  res.send(`<h1>Hello ${v}!</h1>`);
});
app.listen(3001);
```

SANITIZATION

WEB PROTECTIONS: XSS

Example XSS was enabled by user data being treated as code

(special characters escaped the string)

```
import express from "express";
import validator from "validator";
const app = express();

app.get("/page", (req, res) => {
  let v = req.query.v ? req.query.v : "world";
  v = validator.escape(v);
  res.send(`<h1>Hello ${v}!</h1>`);
});
app.listen(3001);
```

SERVERS AS ATTACK TARGETS

OVERVIEW

Spam

- Sent from legitimate IP address, less likely to be blocked

Denial of service

- Attack competitors, or seek ransom

Data theft

- Steal credentials, credit card numbers, intellectual property

Ransomware

- Lock data, only release it for a price

CryptoMiners

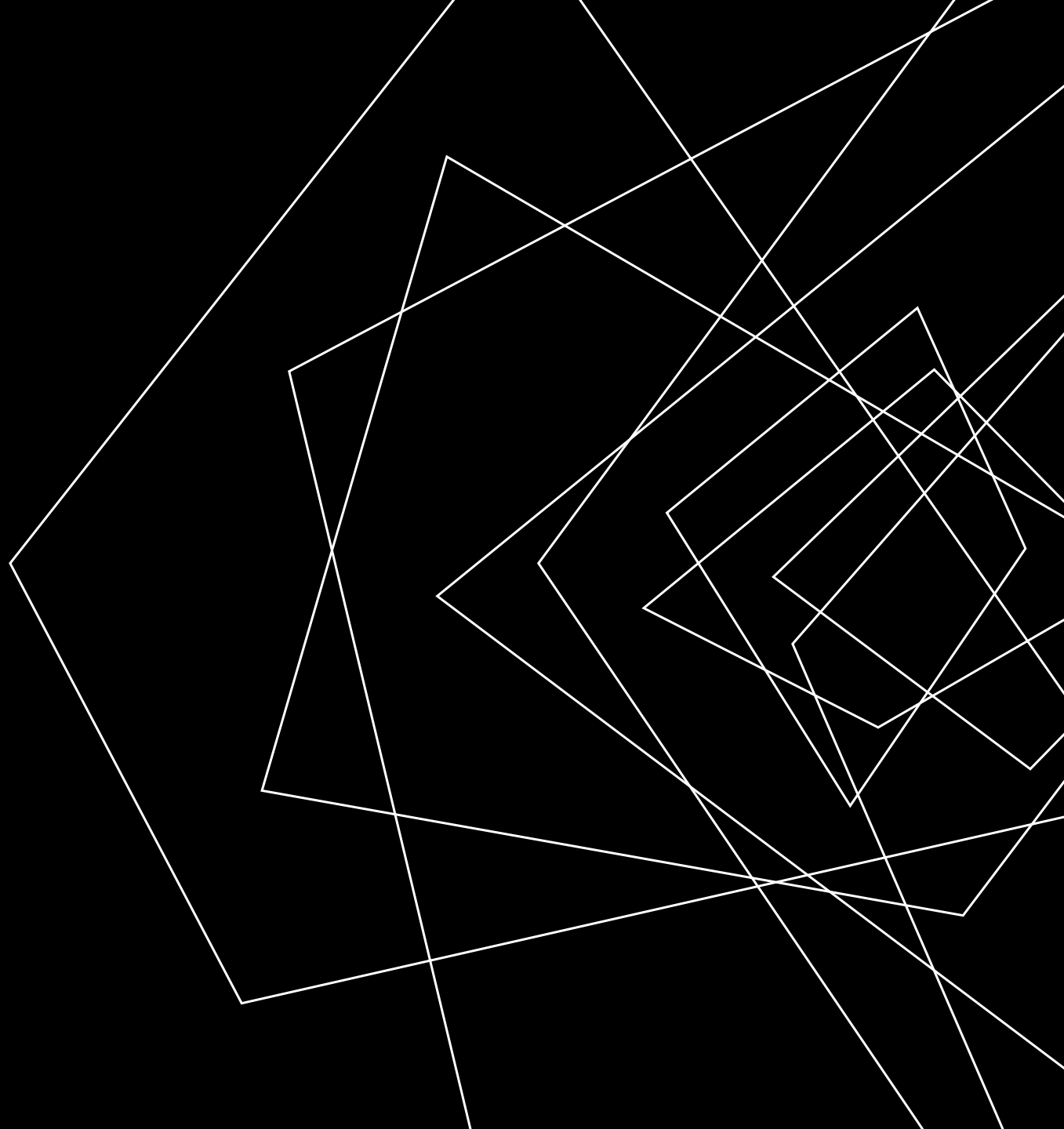
- Use somebody else's compute to mine crypto

Political reasons

- Suppress info you don't like

LECTURE OUTLINE

- Basics / Introduction
- Endpoint Security
- Resource Security

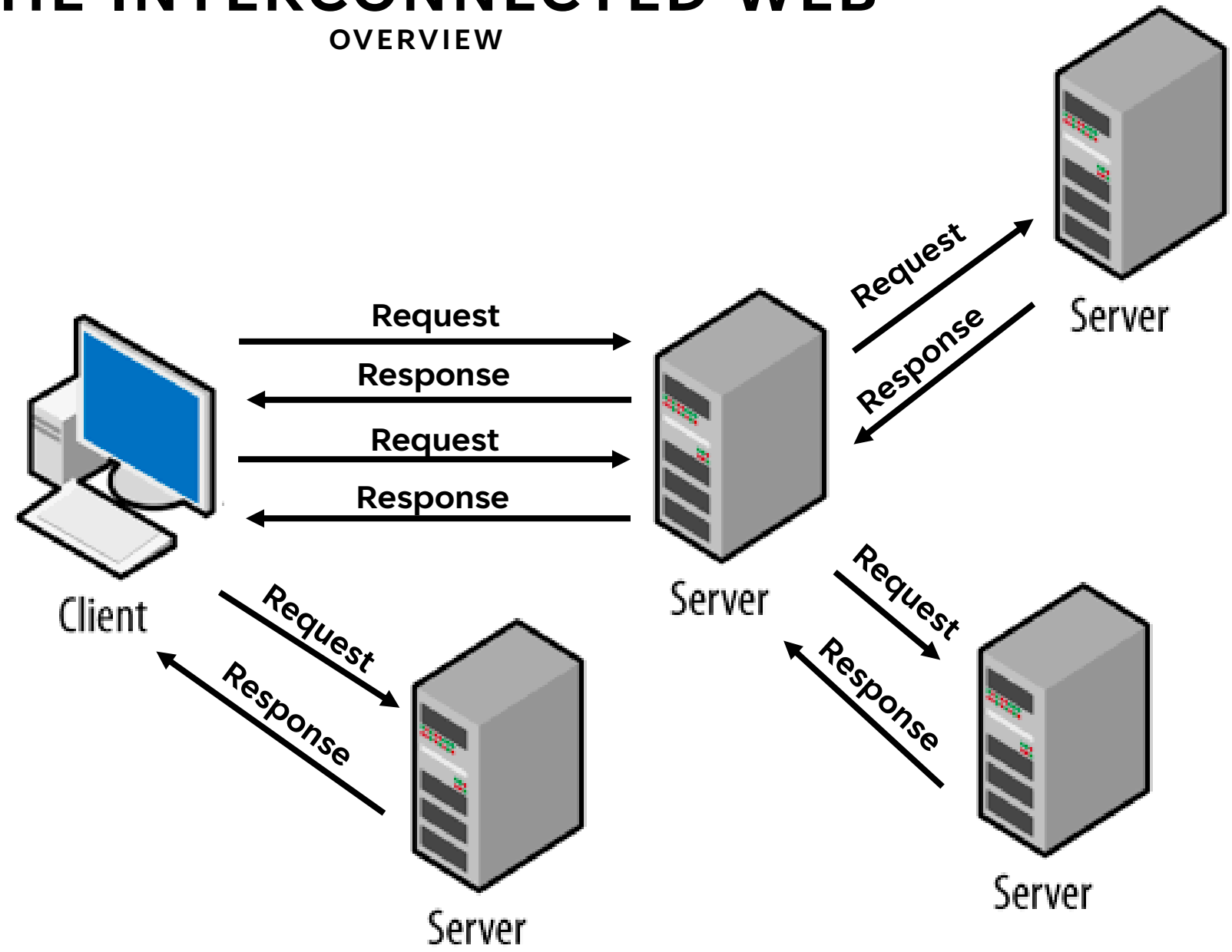


THE INTERCONNECTED WEB

OVERVIEW

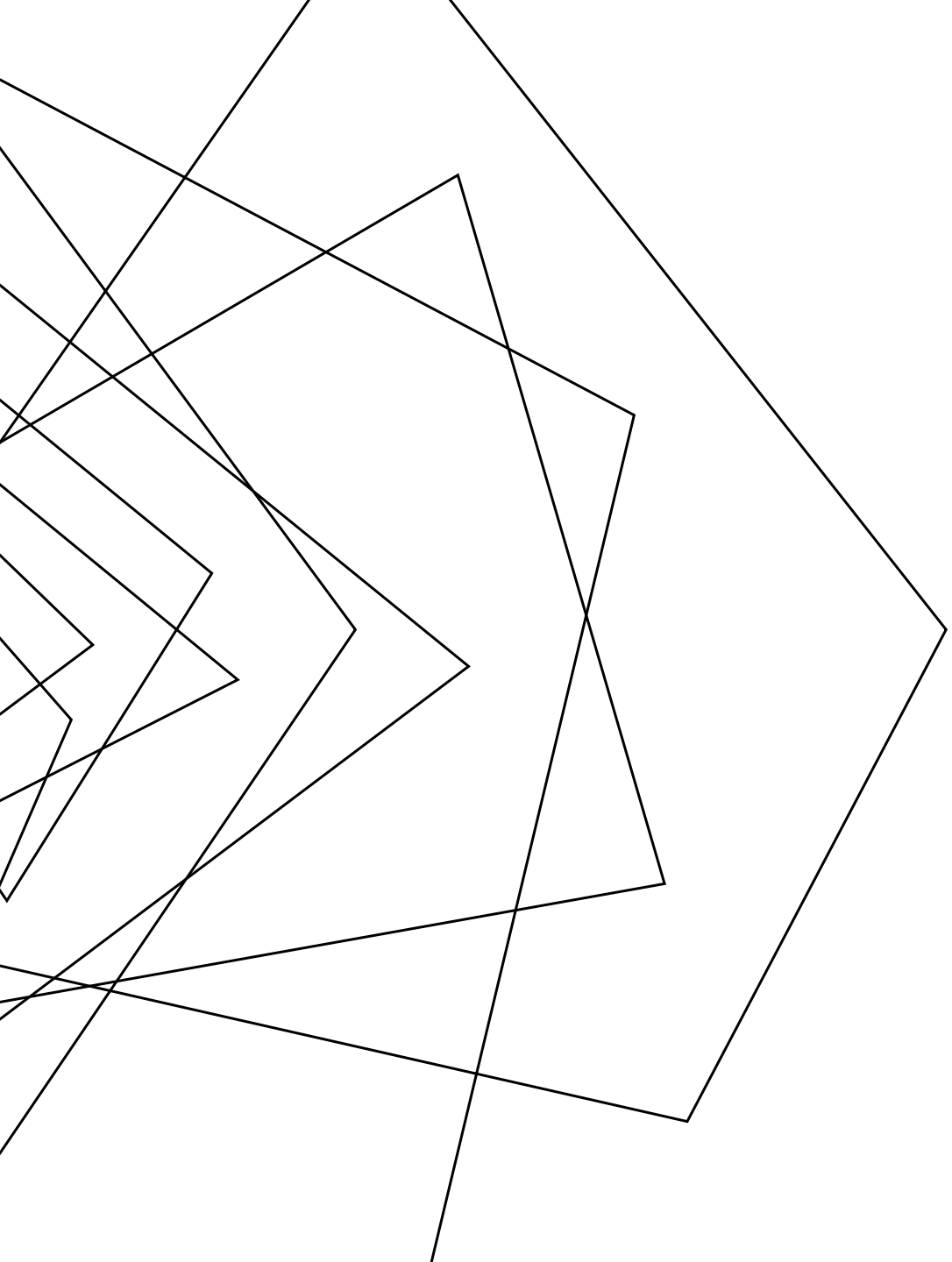
A single page visit is likely to kick off many requests

- Resources from the first server
- Resources fetched by the first server
- Resources from third-party servers at the first server's behest



THE SAME ORIGIN POLICY

WEB PROTECTIONS



LECTURE END!

A BRIEF INTRODUCTION TO THE PROBLEM SPACE