

An abstract composition of various geometric shapes. In the top left, there is a large orange triangle pointing right. Below it is a smaller orange circle. To the right of the orange triangle is a large green semi-circle. In the top right corner, there is a brown circle. On the right side, there are two vertical blue lines. In the bottom left, there is a large green circle. To its right, there are four small blue curved lines. In the bottom right, there is a large orange square.

The overall plan

- Walkthrough building a basic Node app
- Once basic app is built in 1-2 workshops, start fixing some vulnerabilities!
 - In case you miss a workshop, they will be recorded, and code will be available
- Learn about common web vulnerabilities along the way, but also have knowledge of how to secure your web app from these vulnerabilities

With that in mind, an intro to web security!



In summary,
if you can't
attend

- Web security has a LOT of parts to worry about
- Encrypt web traffic, stored data such as passwords
- Require more info than just passwords to identify users
- Clean user input sufficiently to avoid XSS/SQL injection
- Don't use external dependencies on pages with sensitive info, such as a npm package you didn't write



What is
web
security?

Your web browser

Application server

Application client

Databases

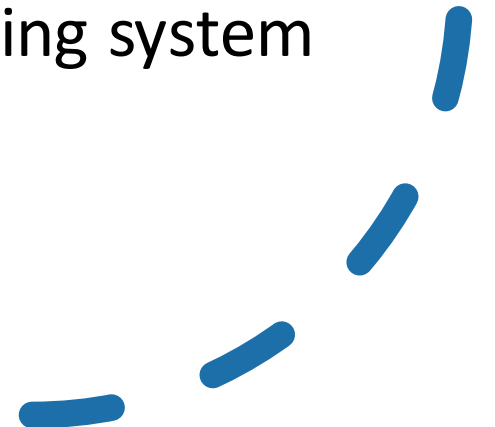
APIs

Networks

The packages you install

Why do we care about web security?

- You don't want your credit card info to be leaked
- You don't want your grandparents' credit card info to be leaked
- You're launching a startup, medical records system, etc. and don't want to face fines
- You don't want users to be unable to access your unemployment filing system
- You don't want your nation's voting system to be hacked





Great, how
can I get
started?

Learn about basic HTML, CSS,
Javascript to create websites with

- Could also learn React for more aesthetic fun!

Learn more about layers of the web
below what the client sees

- Client-server architecture
- HTTP requests
- DNS routing
- Cookies
- Browser architecture



A quote on security at Nikola...

"The entire infotainment system is a HTML 5 super computer. That's the standard language for computer programmers around the world, so using it lets us build our own chips. And HTML 5 is very secure. Every component is linked on the data network, all speaking the same language. It's not a bunch of separate systems that somehow manage to communicate."

- Trevor Milton, CEO of Nikola

Hello world!

I am a basic HTML page.

- List element
- List element 2

HTML - Hypertext Markup Language

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8"/>
    <title>My HTML site</title>
  </head>
  <body>
    <h1>Hello world!</h1>
    <p>I am a basic HTML page.</p>
    <ul>
      <li>List element</li>
      <li>List element 2</li>
    </ul>
  </body>
</html>
```


Hello world!

I am a basic HTML page.

- List element
- List element 2

CSS - Cascading Stylesheet

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8"/>
  <title>My HTML site</title>
  <link rel='stylesheet' href='test.css'/>
</head>
<body>
  <h1 id="unique_element">Hello world!</h1>
  <p>I am a basic HTML page.</p>
  <ul>
    <li class="my_class">List element</li>
    <li class="my_class">List element 2</li>
  </ul>
</body>
</html>
```

Hello world!

I am a basic HTML page.

- List element
- List element 2

CSS - Cascading Stylesheet

Contents of test.css:

```
p { /* specify style for all <p> elements */  
    font-size: 30px;  
}  
p:hover { /* specify style for all <p> elements when hovered on */  
    font-size: 20px;  
}  
#unique_element { /* specify style for a tag with this id */  
    color: #00ffff;  
}  
.my_class { /* specify style for elements with this class */  
    color: #000fff;  
}
```


JavaScript

- Created by UIUC alumnus Brendan Eich
- Powerful language used across the Web
- Named JavaScript due to past popularity of Sun Microsystems' Java, but no other relationship with Java
- Many Javascript web frameworks and tools available

To learn more outside of this guide about Javascript syntax, try out <https://github.com/stanford-web-security/assign0>



Node.js

- JavaScript... but it's on the command line!
 - Has a few more features
 - Used by many to create web applications
- 



npm

- Package manager used by default for Node.js
- Around 450,000+ packages available
- A security issue? **Potentially**



Things you
should install
right now

- Git, if you don't already have it
- Node.js v12
- After installing Node, run 'npm install sqlite3' to install SQLite
- A code editor of your choice
 - Sublime Text works
 - Visual Studio Code is my preference
 - If using VS Code, download Git extension!!





Setting up a simple server in Node.js

- Goal: listen for TCP connections on a port specified as a command-line argument, and write the current time to the socket
- Step 1: create a file named "time-server.js"

Why? Not necessarily specific to security vulnerabilities, but will introduce you to Javascript and basic networking :)

(Idea credit: Feross Aboukhadijeh)

Setting up a simple server in Node.js

- Next: need to create a TCP server in this file
- Can be done using the net module
- Things we need to make this TCP server:
 - Create a server
 - Create a function that listens for connections, runs each time there is a connection
 - Make the server listen on the desired port



Setting up a simple server in Node.js

- Next: need to create a TCP server in this file
- Can be done using the net module
- Things we need to make this TCP server:
 - Create a server - `net.createServer()`
 - Create a function that listens for connections, runs each time there is a connection - function listener (socket) { /* ..TO DO.. */ }
 - Make the server listen on the desired port – `server.listen(8000)`



Setting up a simple server in Node.js

```
const net = require('net')
```



Setting up a simple server in Node.js

```
const net = require('net')  
const server = net.createServer(function  
  (socket) {  
    // handle connections here  
  });
```



Setting up a simple server in Node.js

```
const net = require('net')  
const server = net.createServer(function  
  (socket) {  
    // handle connections here  
  });  
server.listen(8000)
```



Setting up a simple server in Node.js

```
const net = require('net')
const server = net.createServer(function (socket) {
  // handle connections here
});
server.listen(8000)
```

To run the server: node time-server.js in one terminal

To try connecting to it: nc localhost 8000 in another terminal

What happens when you connect?



Setting up a simple server in Node.js

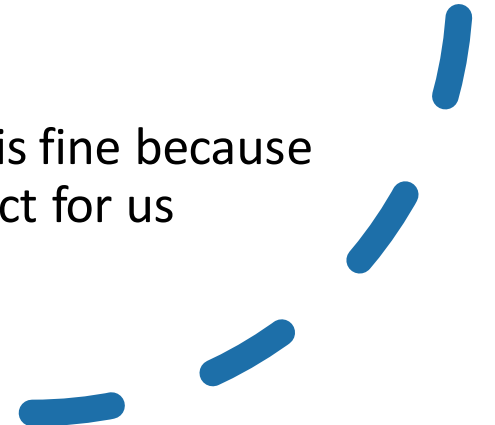
```
const net = require('net')
const server = net.createServer(function (socket) {
  // handle connections here
});
server.listen(8000)
```

To run the server: node time-server.js in one terminal

To try connecting to it: nc localhost 8000 in another terminal

What happens when you connect?

Note: Windows will not have nc installed, this is fine because we will build a client in Node.js that can connect for us



Setting up a simple server in Node.js

```
const net = require('net')
const server = net.createServer(function (socket) {
  // handle connections here
  console.log("somebody connected with me!!!")
});
server.listen(8000)
```

To run the server: node time-server.js in one terminal

To try connecting to it: nc localhost 8000 in another terminal

Now, we can tell if someone connects to us.



Setting up a simple server in Node.js

```
const net = require('net')
const server = net.createServer(function
(socket) {
  // handle connections here
  // how can we actually send data to the
  connection?
});
server.listen(8000)
```



Setting up a simple server in Node.js

```
const net = require('net')
const server = net.createServer(function
(socket) {
  // handle connections here
  socket.write("hello!!!")
});
server.listen(8000)
```

What happens now?



Setting up a simple server in Node.js

```
const net = require('net')
const server = net.createServer(function
(socket) {
  // handle connections here
  socket.end("hello!!!")
});
server.listen(8000)
```



Testing your server, by building a client

You need something that can send your server a TCP request.

If you're on Mac/Linux, you should have "nc" on your command-line. But, in case you don't, or are using Windows, we can also build a Node.js program that will create a TCP client.



Testing
your server,
by building
a client

Things we want our client to do:

- Connect to the server
- Receive data from the server
- Close once we receive data



Testing your server, by building a client

```
var net = require('net');

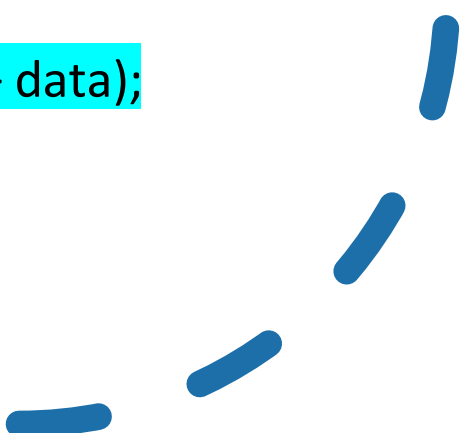
var client = new net.Socket(); // create client
// connect, specifying port, IP address to connect to
client.connect(process.argv[2], '127.0.0.1', function() {
  console.log('Connected to server at port ' + process.argv[2]);
  client.write('Hello server!!');
});
```



Testing your server, by building a client

```
var net = require('net');


var client = new net.Socket(); // create client
// connect, specifying port, IP address to connect to
client.connect(process.argv[2], '127.0.0.1', function() {
  console.log('Connected to server at port ' + process.argv[2]);
  client.write('Hello server!!');
});
// print data when we receive it
client.on('data', function(data) {
  console.log('Data received from the server: ' + data);
});
```



Testing your server, by building a client

```
var net = require('net');

var client = new net.Socket(); // create client
// connect, specifying port, IP address to connect to
client.connect(process.argv[2], '127.0.0.1', function() {
  console.log('Connected to server at port 8000');
  client.write('Hello server!!');
});
// print data when we receive it
client.on('data', function(data) {
  console.log('Data received from the server: ' + data);
});
// print when server connection closed
client.on('close', function() {
  console.log('Connection closed');
});
```



Testing
your server,
by building
a client

Now, to test your server:

- In one terminal: `node my-server.js`
- In another terminal: `node my-client.js 8000`

