COMP1531

📚 Full-Stack - Auth

Lecture 9.1

Author(s): Hayden Smith

(Download as PDF)

In This Lecture

- **Why? 🤔**
  - Basic security is an important step in building good software
- **What? 📰**
  - Authentication
  - Authorisation

🤨 Auth

It's a nickname COMP1531 gives two important concepts:

Authentication: Process of verifying the identity of a user

Authorisation: Process of determining an authenticated user's access privileges

Note: UNSW has more cybersecurity courses on these topics.

🤨 Auth

Examples:

Authentication: Checking if you have a username/password for a valid MS Teams account

Authorisation: For your valid MS Teams account, checking if you are an admin or not

🔒 Authentication

What is the the most basic approach?

1. User registers, we store their username and password
2. When user logs in, we compare their input password to their stored password
3. If it matches, they entered the right password

🔒 Authentication

What's wrong with this?

```
type Data = {
  users: { [email: string]: string };
};

const data: Data = {
  users: {},
};

function register(email: string, pw: string) {
  if (email in data.users) {
    return false;
  } else {
    data.users[email] = pw;
    return true;
  }
}

function login(email: string, pw: string) {
  if (email in data.users) {
    if (pw === data.users[email]) {
      return true;
```

```
    }
  }
  return false;
}
```

9.1_auth_simple.ts

🔒 Authentication

What's wrong with this?

🔒 Authentication

What's wrong with this?

We're storing peoples' passwords!

🔒 Authentication

What's wrong with this?

We're storing peoples' passwords!

In this example, yes, it's just being stored in a variable in RAM, which is OK. But in reality, our "data" would be stored on a hard drive long term! Which is scary. How do we avoid this??

🔒 Authentication

What's wrong with this?

We're storing peoples' passwords!

In this example, yes, it's just being stored in a variable in RAM, which is OK. But in reality, our "data" would be stored on a hard drive long term! Which is scary. How do we avoid this??

We need to 🦄 hide 🦄 the password

🔒 Authentication

What's wrong with this?

We're storing peoples' passwords!

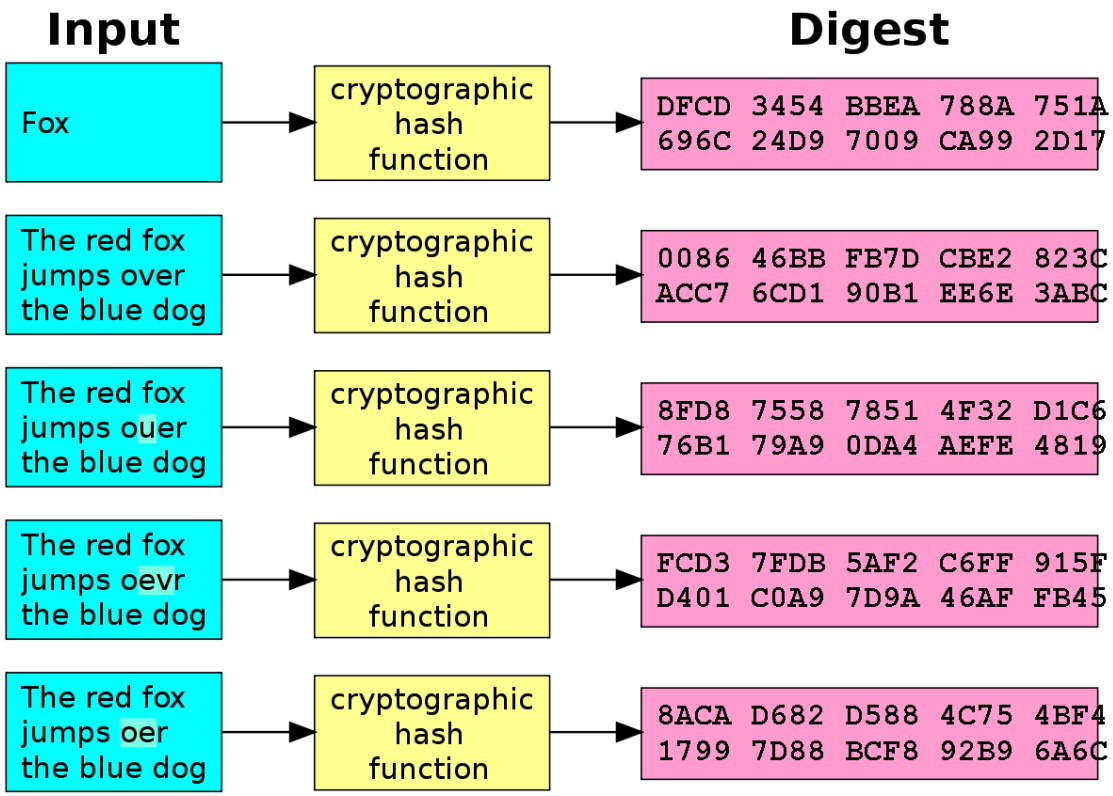
In this example, yes, it's just being stored in a variable in RAM, which is OK. But in reality, our "data" would be stored on a hard drive long term! Which is scary. How do we avoid this??

We need to 🦄 hide 🦄 the password

..what does that even mean..

🕵🏽 Hiding Information

Encryption and **Hashing** are both processes of taking plaintext information and concealing it by turning it into a seemingly random string of characters.



Source

🕵🏽 Hiding Information

Encryption is reversible.

Hashing is irreversible.

Reversibility does make the hiding process finitely less secure (since a method to "unhide" the information exists). But it provides the convenience of being able to reverse it!

Hiding Information

Let's explore a **hashing** example together.

#️⃣ Hashing

Hashing is our method of irreversibly hiding information. One way to generate a hash is to use the `crypto` library. This is **built-in** to NodeJS so you don't need to `npm install` anything.

```ts
import crypto from 'crypto';

function getHashOf(plaintext: string) {
  return crypto.createHash('sha256').update(plaintext).digest('hex
}

const msg = 'BigMacSecretSauce';
const hash = getHashOf(msg);

console.log(hash);

export { getHashOf }; // ignore this line
```

9.1_hash.ts

`getHashOf` converted `BigMacSecretSauce` to `8c64be3db244091660a3b69ef548e3d43f9f945aaae78e1ff2de939cac1116ba`

There is no way to convert `hash` to `msg`! Even if you know the entire method of how this conversion happens, it's not reasonably possible to reverse it.

#️⃣ Hashing

How would we apply this to our authentication issue?

```ts
type Data = {
  users: { [email: string]: string };
};

const data: Data = {
  users: {},
};

function register(email: string, pw: string) {
  if (email in data.users) {
    return false;
  } else {
    data.users[email] = pw;
    return true;
  }
}

function login(email: string, pw: string) {
  if (email in data.users) {
    if (pw === data.users[email]) {
      return true;
    }
  }
  return false;
}
```

9.1_auth_simple.ts

#️⃣ Hashing

How would we apply this to our authentication issue?

```ts
type Data = {
  users: { [email: string]: string };
};

import { getHashOf } from './9.1_hash';

const data: Data = {
  users: {},
};

function register(email: string, pw: string) {
  if (email in data.users) {
    return false;
  } else {
    data.users[email] = getHashOf(pw);
    return true;
  }
}

function login(email: string, pw: string) {
  if (email in data.users) {
    if (getHashOf(pw) === data.users[email]) {
      return true;
    }
  }
  return false;
}
```

9.1_auth_fixed.ts

Authorisation

Authorisation is a much simpler topic. It's really just about you having appropriate logic to decide what permissions a given authenticated user does or doesn't have.

An example is in your project: Is the user a member or an admin?

We will not explore this topic further in COMP1531.

👂 Feedback

Or go to the form here.