Logging Secrets

In this lesson, we'll study a few scenarios where logging can go awry and how to prevent them.

We'll cover the following

Logging sensitive data

If you develop systems that have to deal with secrets such as passwords, credit card numbers, security tokens or personally identifiable information (PII), you need to be very careful about how you deal with this data within your application, as a simple mistake can lead to a data leak in your infrastructure.

Take a look at this example, where our app fetches user details based on a header.

```
app.get('/users/me', function(req, res){
    try {
        user = db.getUserByToken(req.headers.token)
        res.send(user)
    } catch(err) {
        log("Error in request: ", req)
    }
})
```

Now, this innocuous piece of code is actually dangerous, if an error occurs, the entire request gets logged.

Having the whole request logged is going to be extremely helpful when debugging but will also lead to storing auth tokens (available in the request's headers) in our logs. Anyone who has access to those logs will be able to steal the tokens and

impersonate your users.

Your logs are not top-secret

You might think that since you have tight restrictions on who has access to your logs, you'll be safe. Chances are that your logs are ingested into a cloud service such as GCP's StackDriver or AWS' CloudWatch, meaning that there are more attack vectors, such as the cloud provider's infrastructure itself, the communication between your systems, the provider to transmit logs, and so on.

Whitelisting

The solution is to simply avoid logging sensitive information. Whitelist what you log and be wary of logging nested entities as there might be sensitive information hiding somewhere inside them, such as `req.headers.token`.

Masking

Another solution would be to mask fields, for example, turning a credit card number such as `1111 2222 3333 4444` into `**** **** **** 4444` before logging it.

That can be a dangerous approach. An erroneous deployment or a bug in your software might prevent your code from masking the correct fields, leading to leaking the sensitive information. As I like to say, use it with caution.

Inputting sensitive data into the wrong field and logging it

Last but not least, I want to mention one particular scenario in which any effort we make not to log sensitive information is in vain, when users input sensitive information in the wrong place.

You might have a login form with username and password, and users might actually input their password in the username field (this can generally happen when you auto remember their username so that the input field is not available the next time they log in). Your logs would then look like this:

```
user e0u9f8f484hf94 attempted to login: failure
user lebron@james.com attempted to login: success
...
```

Anyone with access to those logs can figure an interesting pattern out, if a username doesn't follow an email pattern (*email@domain.tld*), chances are the string is actually a password the user had wrongly typed in the username field. Then you would need to look at the successful login attempts made shortly after and try to login with the submitted password against a shortlist of usernames.

What is the point here? Security is hard and, most often, things will work against you, in this context, being paranoid is a virtue.

> ### ℹ Who is silly enough to log a password?
>
> You might think logging sensitive information is an amateur's mistake, but I argue that even experienced programmers and organizations fall under this trap. Facebook, in early 2019, suffered a security incident directly related to this problem. As Brian Krebs put it:
>
> *"Facebook is probing a series of security failures in which employees built applications that logged unencrypted password data for Facebook users and stored it in plain text on internal company servers."*
>
> This is not to say that Facebook should not be held accountable for the incident, but rather that we can probably sympathize with the engineers who forgot the `console.log` somewhere in the code. Security is hard, so making sure we pay extra attention to what we log is an important matter.

In the next lesson, we'll look at why you can never trust clients when building an app.