YOU WILL PWN YOURSELF

WITH YOUR OWN CLIPBOARD

SAUL JOHNSON

A (VERY) BRIEF INTRODUCTION

- I'm Saul, a software verification researcher here at Teesside University.
- I'm mainly working with formal methods around password strength/cracking but security research is fun too!
- GitHub: @lambdacasserole
  Twitter: @lambdacasserole
  Web: https://sauljohnson.com

COPYING AND PASTING CODE

A TRAGEDY IN FOUR ACTS

ACT I: THE DEADLINE

- **The client is breathing down your manager's neck and your manager is breathing down yours.** There's a deadline you're racing to meet and there's one thing left to do.
- The client wants the page to load a so-called *"analytics pixel"* after a delay of a couple of seconds to track user *"bounce rates"*
- This will allow the client to track which proportion of users stay on the website for more than a few seconds.

ACT II: WHAT WAS THAT FUNCTION CALLED AGAIN?

- There's a function in JavaScript to execute some code one time only after a delay... was it `setInterval`? No, that's not right… you wrack your brain.
- The function name is right on the tip of your tongue as your manager peeks around the corner *"Client says she'll be here in 5 minutes, that demo ready?"*
- Your train of thought scatters to the four winds…

ACT III: LET ME JUST GOOGLE THAT REAL QUICK

- *"Working on it, 2 minutes."* You reply as it dawns on you that there truly is no more time, you Google *"load image after delay js"*, pull up the first answer that comes up and glance it over in a hurry. It all comes flooding back, that function is `setTimeout` of course!
- In fact, this code looks like exactly what you need. In one fluid motion… copy, paste, a few tweaks and… done!
- The meeting goes smoothly, the client loves it (especially your shiny new analytics pixel script) and you grab Thai food on the way home to congratulate yourself on a job well done.

INTERLUDE: HERE'S WHAT YOU COPIED

Here's the code you copied and pasted over. Nothing overtly suspicious, right?

```
var element = '<img src="/demo.png" onload="">';
setTimeout(function() {
    document.body.innerHTML += element
        .replace(/\u200b/g, '0')
        .replace(/\u200c/g, '1')
        .replace(/\u200d/g, ' ')
        .replace(/\d+\s?/g, x => String.fromCharCode('0b' + x)), 2000);
    });
```

ACT IV: SORRY I TRACKED MALWARE ALL OVER THE PROJECT

- A few months pass uneventfully, new clients come and go, you've forgotten all about that analytics pixel, until one day the project manager bursts in. *"Remember that client from a few months back? Website's been compromised and credit card numbers are being lifted from the payment details page. We also think there's a cryptocurrency miner in there."*
- All your dependencies are clean... the team has combed over them again and again by hand and there's nothing to be found.
- Let's dissect the code from earlier…

A SECOND LOOK (THIS TIME WITH COMMENTS)

Here's the code from before, with comments added.

```
var element = '<img src="/demo.png" onload="">'; // The image element to add.
setTimeout(function() { // Execute the following code after 2000 milliseconds.
    document.body.innerHTML += element // Add the image to the page.
        .replace(/\u200b/g, '0') // I dunno, something to do with URLs?
        .replace(/\u200c/g, '1') // Something else to do with URLs?
        .replace(/\u200d/g, ' ') // See where this is going?
        .replace(/\d+\s?/g, x => String.fromCharCode('0b' + x)), 2000); // Wat?
    });
```

LIVE DEMO TIME!

ON THE IMPORTANCE OF KNOWING WHAT YOUR CODE DOES…

CHARACTERS EXIST THAT WE CAN'T SEE!

- In Unicode, there exist several so-called "non-printable" characters that are not displayed in most ordinary text editors by default.
- The ones we use here are "zero-width" because unlike regular whitespace characters like the space and the tab, they take up zero horizontal line width (i.e. none at all).
- The three used here are the "zero-width space", "zero-width joiner" and "zero-width non-joiner".
- Uses for these vary, but they do have legitimate use cases in, for example, languages that don't use spaces as part of their script, and languages that use a lot of different diacritics.
- Hidden in the `onload` event handler from earlier is a malicious JavaScript payload, encoded as zero-width characters.

HERE'S WHAT'S GOING ON

Here's the code snippet from earlier, see what's happening now?

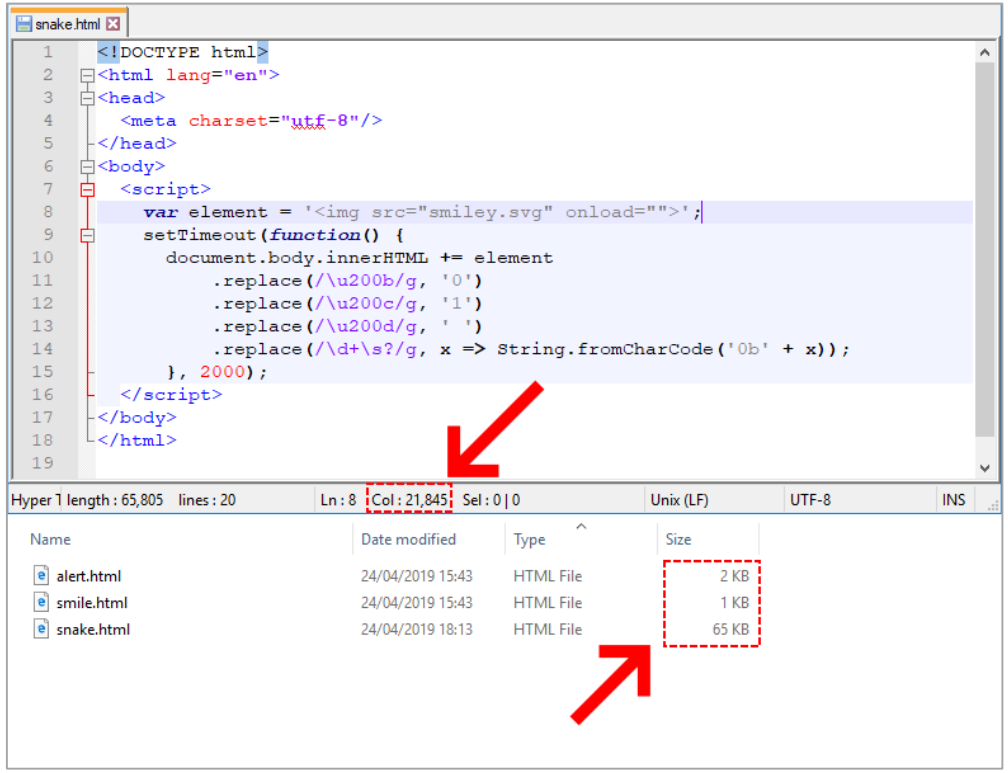
```
// Decode to binary (ASCII text).
.replace(/\u200b/g, '0')
.replace(/\u200c/g, '1')
.replace(/\u200d/g, ' ')

// Decode binary to JavaScript code.
.replace(/\d+\s?/g, x =>
    String.fromCharCode('0b' + x)), 2000);
```

Name	Codepoint	Translation
ZERO WIDTH SPACE	U+200B	0
ZERO WIDTH NON-JOINER	U+200C	1
ZERO WIDTH JOINER	U+200D	<space>

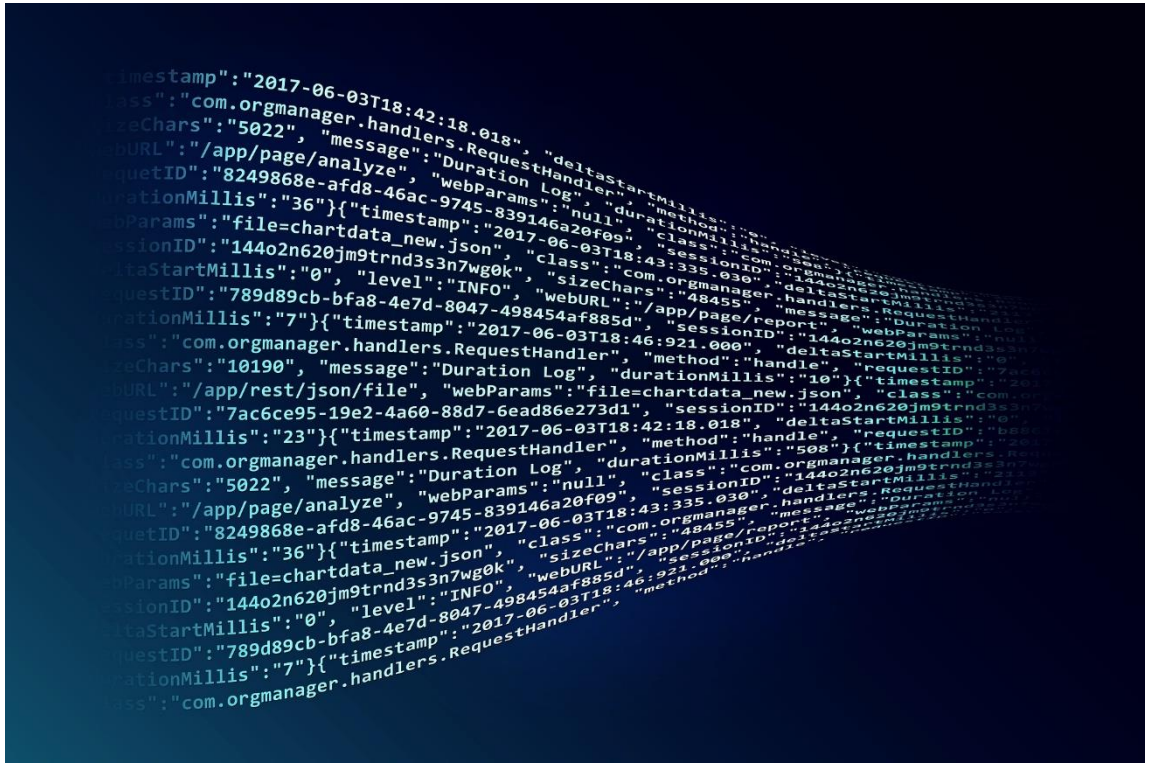
SO WHAT ARE THE LIMITATIONS HERE?

- This technique allows us to embed arbitrary scripts into webpages with no visible changes to the source code at all. It's (almost) free real estate for whatever payload we like, so long as the decoding logic is in there somewhere.
- File size will be a dead giveaway, though. As will the UI of a text editor if it shows the current column number, for example.
- The cursor will also get "stuck" if we step through the text with our arrow keys.



HOW CAN I MESS WITH THIS MYSELF?

- I make a toolkit available on GitHub for encoding scripts as zero-width characters, as well as detecting the presence of the three zero-width characters used here in text files.
- Every file shown during this presentation was created with this toolkit, which is written in Python and Bash.
- Get your hands on it here: https://github.com/lambdacasserole/zero-width-js
- Be responsible with this code. It doesn't do anything you couldn't do manually with Notepad++ and a free online text to binary converter, but be ethical and legal please.



THANK YOU FOR YOUR ATTENTION!

I'LL BE AROUND AFTERWARDS IF YOU'D LIKE TO TALK OFFLINE!

ACKNOWLEDGEMENTS

- Jordan Eleredge wrote the Snake implementation in JavaScript:
  - Jordan's GitHub: https://github.com/captbaritone
  - The Snake repository itself: https://github.com/captbaritone/snake.js