# Charlotte Som

Systems programmer, videogame cheat developer, reverse engineer, multispecialist.

charlotte@som.codes
@char
char.lt
hackery.site

## Writing

### "Circumventing the JVM Classfile Verifier"

*2019* *som.codes/jvm-force-no-verify*

### "Game Science: GTA V's Stunt Jumps"

*2021* *som.codes/gta-v-stunt-jumps*

### "Circumventing Cisco Duo's Authenticator App"

*2021* *som.codes/cisco-duo-bypass*

### "Extracting API Keys from a Minecraft Launcher"

*2022* *som.codes/mmc-msaclientid*

## Competitions

### Google Hash Code 2020

***Google***

*February 2020*

Worldwide intractable problem optimization competition. Competed as part of a globally-distributed team of three.

- **UK**: 6th place out of 406 teams.
- **USA**: 13th place out of 619 teams.
- **Canada**: 3rd place out of 135 teams.
- **Worldwide**: 252nd place out of 10724 teams.

### Deloitte UK CTF

***Deloitte***

*December 2020* London

UK-wide infosec competition, organised by Deloitte. As a team of six, we participated as "Loughborough University".

- **Qualifiers**: 10th place.
- **London finals**: 7th place.

### BLÅHAJ CTF Team

*2021 – 2022*

- **Hack-A-Sat 2 CTF**: Organised by the *US Air Force & US Space Force*. Placed 15th out of 697 teams.
- **corCTF 2021**: Placed 11th out of 904 teams.

## Languages

- **English** (native, primary)
- **French** (maternal)
- **Spanish** (since 2013)
- **Korean** (since 2016)

## Experience

### Multiplayer game technology

***WorldQL Corporation***

*April 2022 – Present*

I currently work at WorldQL, where the mission is to reduce the barrier between singleplayer and multiplayer game development.

Here, I've worked on EVM-based transaction processing / smart contracts (writing a hot wallet transaction scheduler in Rust), written a multiplayer web game engine in TypeScript (with several interesting netcode paradigms) & backend web services in Rust (at one point embedding `deno_core` in a Rust application for sandboxed multitenant user code exec on the server-side) as well as a (Git-based) beginner-oriented version control system.

To facilitate prototype art we also experimented with Blender automation and machine learning in production, writing & deploying Python + CUDA / ONNX services.

### Game modding / game-hacking

***Self-employed***

*2016 – 2025*

Game development & game modding is a very effective entrypoint into programming at large. I started in 2007(!!) developing ROBLOX games.

- *GTA Online*: C++ (primarily), Rust
- *Minecraft*: Java, Kotlin (primarily), Scala, JVM Bytecode
- *Counter-Strike: Global Offensive*: C++ (primarily), Rust
- Unity3D (various): C# (primarily), .NET MSIL, and C++ (for `il2cpp` games)

In my teens, I used to sell a custom client for Minecraft (5-figure USD revenue from 2016 – 2019), and a subscription trainer menu for GTA Online. I find the resulting end-to-end product experience & reverse engineering skills valuable to this day.

### Commercial obfuscator for JVM programs

***paramorphism.dev (Self-employed)***

*2017 – 2022*

This solo project involved writing the product itself, internal tooling around JVM bytecode to aid debugging, and the sales site in SvelteKit (then-*Sapper*, actually).

- *Paramorphism*: Bytecode obfuscator for JVM programs written in Kotlin.
- *libparamorphism*: Optional native runtime opaque library for programs obfuscated by Paramorphism written in Rust first and then Zig later.
- *Koffee*: Domain-specific language for Kotlin for Java classfile generation.
- *Aksara* (internal): Bytecode assembly language and assembly/disassembly toolchain written in Kotlin.
- *Katon* (internal): Bytecode viewer and editor with a GUI written in Rust and interfacing via Java Native Interface to Kotlin (*Aksara*) and Java (*Fernflower*).
- *Citadel*: An experimental custom-instruction-set VM with a *libparamorphism*-backed native executor (+ some compilers), focusing on self-modifying code.

### E-commerce automation

***Force Software LLC***

*November 2020 – October 2021*

The flagship "Splashforce" product was an end-user e-commerce automation suite for various sneaker sites. My duties included:

- `cronet`: Custom patches for & isolation of Chromium's HTTP stack. (for use as a library to evade TLS fingerprinting in the application.)
- Pairing an Electron (node.js / web JavaScript) frontend to a Go backend.
- Anti-piracy, reverse-engineer-deterrent releases for the Go backend. (Code virtualization, `cgo` FFI bindings, etc.).
- Reverse engineering JS & WASM bot protection measures on websites.
- Reverse engineering & circumventing bot prevention measures on Android apps. (Java, Smali, etc).