COOPER KASPIAN

Systems Developer • Security Researcher • Reverse Engineer
Salem, OR | ckaspian@pm.me | linkedin.com/in/ckaspian | github.com/ckaspian

SUMMARY

Systems developer and security researcher with hands-on experience in C/C++ reverse engineering, Windows internals, driver-level interaction, and game engine exploitation. Experienced in building both internal and external tooling that interfaces with running processes from both user-land and kernel. Demonstrated understanding of anti-tamper evasion techniques, DMA-based memory access, signature scanning, runtime hooking, and binary analysis across multiple game engines (Source, Unity, Unreal). Currently developing WASM VM solutions with advanced encryption routines for commercial DRM at Seyra Online.

TECHNICAL SKILLS

Languages	C++ (primary), C, Rust, Go, C#, ARMv9, x86/x64 Assembly, JavaScript, Python
RE & Security	Binary analysis, signature scanning, DLL injection, process hollowing, runtime hooking, DMA memory access, anti-tamper analysis, obfuscation/encryption
Windows Internals	Kernel-mode driver development, NT API, processor intrinsics, PEB/TEB traversal, syscall hooking, memory-mapped I/O
Toolchain	Clang/LLVM, GCC, MSVC, CMake, vcpkg, x64dbg, IDA, Binary Ninja, Ghidra, Git, GitHub CI/CD
Systems	Async runtimes (Tokio), gRPC/Protobuf, WebAssembly VMs, service-oriented architecture, Linux systems administration

EXPERIENCE

Back End Developer | Seyra Online

November 2025 – Present | North America (Remote)

- Developed WASM VM in Rust and JavaScript interface for commercial DRM, focusing on advanced encryption routines with minimized binary footprint
- Architected secure runtime environment prioritizing tamper-resistance and efficient code execution for digital rights management
- Implemented WebAssembly-based obfuscation layer to protect proprietary content delivery pipelines

SECURITY RESEARCH & REVERSE ENGINEERING PROJECTS

Apex Legends DMA Solution — *C++, CMake, DMA Hardware*

- Built a DMA-based memory interaction framework for Apex Legends, demonstrating deep understanding of how cheats bypass kernel-level anti-cheat protections
- Implemented frontend/backend architecture with shared headers for structured cross-process communication over dependencies
- Analyzed EasyAntiCheat (EAC) detection vectors and kernel-level memory protection mechanisms

CS:GO Internal & External Tooling — *C++, C, Win32 API*

- Developed both internal (DLL-injected) and external (ReadProcessMemory) cheat frameworks for CS:GO (Source engine), with 348+ commits across the internal project
- Built custom CS:GO SDK from reverse-engineered game structures including entity lists, bone matrices, and ConVar systems
- Implemented kernel-mode driver for external memory access, demonstrating NT driver development and user/kernel boundary exploitation

TF2 Internal Framework — *C++, Source Engine SDK*

- Created an internal cheat framework for Team Fortress 2 targeting the Source engine runtime, with hook-based function interception and entity enumeration

CigFinder — Signature Detection Tool — *C (Pure), YARA-style Patterns*

- Authored a high-performance binary signature scanner in pure C with zero external dependencies, using direct byte-sequence pattern matching
- Designed for cross-platform POSIX compliance with minimal memory footprint — directly applicable to anti-cheat signature scanning workflows

Runtime Modification Tooling — *C#, BepInEx, MelonLoader, Protobuf*

- Developed runtime hook-based mods for Unity and IL2CPP titles (Muse Dash, Death Must Die, Lethal Company), patching compiled assemblies at load time
- Muse Dash Unlocker garnered 22+ GitHub stars; demonstrated IL decompilation analysis and runtime state manipulation

Open-source Contributions — *Rust*

- Contributed upstream bug fixes to Protocol Buffers serialization and Dioxus rendering pipeline

ADDITIONAL PROJECTS

Godin — Production Backend Service — *Go, PostgreSQL, Discord API*

- Designed service-oriented backend with dynamic handler dispatch, custom PostgreSQL abstraction layer with transaction support, and structured observability via zap logging

Wednesday Oxidize — *Rust, Tokio, Async Runtime*

- Rust rewrite of core infrastructure emphasizing memory safety, async/await patterns at scale, and compile-time correctness via the type system

EDUCATION

Associate of Arts and Sciences — Computer Science
Chemeketa Community College | *December 2025 – August 2027 (In Progress)*

High School Diploma — Computer Software Engineering
South Salem Senior High School | *August 2022 – June 2025*

AWARDS & ACHIEVEMENTS

- SkillsUSA Bronze Medalist & Nationalist Nominee
- George Fox D1 Silver Medalist & Top 5 Finisher
- TSA Coding Gold Medalist
- GitHub Developer Program Member