BLACKHAT ZIG

```
const std = @import("std");

pub fn main() !void {
    var arena = std.heap.ArenaAllocator.init
    defer arena.deinit();

    var buf = try arena.allocator().alloc(u8

    std.debug.print("pwn with zig\n", .{});
}
```

```
0x0000   90 90 90 90  55  48  89  e5  48  83  ec  20
0x0010   48 8b 05 78  56  34  12  48  89  c7  e8  23
0x0020   45 67 89 c3  48  83  c4  20  5d  c3  90  90
0x0030   90 90 90 90  90  90  90  90  90  90  90  90
```

OFFENSIVE SECURITY, EXPLOIT DEVELOPMENT, AND TOOLING WITH THE ZIG PROGRAMMING LANGUAGE

EXPLOIT DEVELOPMENT

REVERSE ENGINEERING

NETWORK SECURITY TOOLING

RED TEAM TRADECRAFT

KERNEL INSTRUMENTATION

DEFENSIVE SECURITY ENGINEERING

```
push  rbp
mov  rbp, rsp
sub  rsp, 0x20
mov  rax, qword ptr [rip+0x12345678]
mov  rdi, rax
call rax
add  rsp, 0x20
pop  rbp
ret
```

STEVE T.

BlackHat Zig

Offensive Security, Exploit Development, and Tooling with the Zig Programming Language

Steve T. Team Publications

This book is available at https://leanpub.com/blackhatzig

This version was published on 2026-07-03

Contents

Offensive Security, Exploit Development, and Tooling with the Zig Programming Language

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

About this book

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Introduction: The Right Tool for the Job

In cybersecurity, your toolchain is an extension of your methodology. A buffer overflow exploit demands precise memory layout control. A packet fuzzer needs raw socket access with zero-copy construction. A reverse engineering utility must parse binary formats without depending on libc. A shellcode loader requires position-independent code with no hidden allocations. Each domain has its own requirements, and the tools that serve them well share a common profile: they give the programmer direct control over memory, execution, and output, while minimizing abstraction overhead.

For decades, that profile has been served by C. C's simplicity, its predictable compilation model, and its C ABI compatibility made it the lingua franca of security tooling. Every exploit framework, every disassembler, every packet crafting library traces its lineage back to a C foundation. But C comes with baggage: undefined behavior, manual memory management, and a compiler that will happily optimize away the very code you need to control.

Enter Zig.

Zig is not a replacement for C. It is a reimagining of what a systems language can be while retaining the properties that make C valuable in security work. It compiles to native code with LLVM, targets the same architectures as C, and provides a C-compatible ABI that lets it call C functions directly. But where Zig diverges is in its design philosophy: every allocation is explicit, every control flow is visible, and every optimization is under the programmer's control. There are no hidden allocations, no preprocessor macros, no operator overloading that can hide function calls, and no garbage collector that might run at an unpredictable moment.

This matters for security because security tools operate in hostile environments where predictability is a feature, not a bug. When you write a packet fuzzer, you need to know exactly what bytes go on the wire. When you write an exploit PoC, you need to know exactly how memory is laid out. When you write a reverse engineering utility, you need to know exactly which symbols are resolved and which are not. Zig's design ensures that the code you write is the code that runs, with minimal surprises from the compiler or runtime.

This book is for security researchers, exploit developers, reverse engineers, and red teamers who want to understand how Zig can be used across the cybersecurity spectrum. It assumes familiarity with C/C++ and basic cybersecurity concepts, but no prior Zig experience is required. Each chapter builds on the last, moving from language fundamentals through offensive applications to defensive countermeasures. We will write production-quality code, examine real-world case studies, and explore both the opportunities and limitations of using a young language in a field that demands reliability above all else.

The chapters that follow are organized to take you from foundation to mastery:

Chapter 1 establishes the case for Zig in security, comparing it directly with C/C++ and Rust across dimensions that matter to security tooling. Chapter 2 dives deep into Zig's memory model — allocators, stack vs. heap, explicit allocation — and why this matters for exploit development. Chapter 3 covers comptime metaprogramming, the feature that lets you generate parsers, encoders, and protocol handlers at compile time. Chapter 4 explains C interop, how to bridge to the world of existing exploits and libraries, and where the integration breaks down.

Chapters 5 through 10 cover the offensive security spectrum: reverse engineering tools, exploit development, network security, red teaming and stealth, operating system internals, and detection evasion. Each chapter includes hands-on code examples and real-world case studies drawn from actual research and tooling projects. Chapter 11 flips the perspective to defensive countermeasures, showing how Zig's safety features bring value to blue-team work. Chapter 12 looks forward at the ecosystem, community, and roadmap, and provides a hands-on walkthrough of building your first Zig security tool.

By the end of this book, you will understand how Zig's design makes it uniquely suited for building offensive security tools, how to write production-quality code in the language, and what trade-offs to expect as you adopt it into your workflow. More importantly, you will understand why a language that prioritizes explicit control, predictable compilation, and zero-cost abstractions is a natural fit for the cybersecurity spectrum.

Chapter 1: Why Zig for Security?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

The Language: Zero-Cost Abstractions, Comptime, and the "C-Compatible ABI" Promise

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Zig vs. C/C++: Memory Control, Undefined Behavior, and Compilation Model

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Zig vs. Rust: Safety Guarantees, Ecosystem Maturity, and Learning Curve

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

The Security Tooling Gap: Why Security Tools Need a Different Language Profile

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Ethical Framework: Educational and Research Focus

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Chapter 2: Zig's Memory Model — The Foundation of Exploit Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Allocators in Zig: std.heap, Custom Allocators, and Arena Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Stack-Allocation-First Design: Why It Matters for Performance and Predictability

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Explicit Memory Control: No Garbage Collector, No Hidden Allocations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Buffer Overflows and Memory Safety: How Zig's Model Helps (and Where It Doesn't)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Case Study: Building a Stack-Smashing Detector in Zig

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Chapter 3: Comptime Metaprogramming — Code Generation for Security Tools

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Comptime Fundamentals: Compile-Time Execution and Type-Level Programming

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Code Generation Patterns: Generating Parsers, Encoders, and Protocol Handlers at Compile Time

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Type Introspection: Building Reflective Security Utilities

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Case Study: A Comptime-Driven Packet Parser Generator

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Trade-offs: Readability vs. Metaprogramming Complexity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Packed Structs and Binary Format Mapping

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Chapter 4: C Interop — Bridging to the World of Exploits

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

The C ABI Bridge: How Zig Compiles C Headers and Calls C Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Calling Conventions and Pointer Types: Mapping C Types to Zig

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Replacing C in Exploit Code: When to Use Zig's FFI vs. Inline Assembly

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Case Study: Porting a C Exploit PoC to Zig

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Limitations: What Doesn't Work Seamlessly

Chapter 5: Reverse Engineering with Zig — Tools and Techniques

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Binary Parsing in Zig: Reading ELF, PE, Mach-O Formats

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Building a Minimal Disassembler: x86/x64 Instruction Decoding

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Symbol Resolution and Import Tables: Navigating Binaries Without libc Dependencies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Case Study: A Comptime-Driven Disassembler for x86_64

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Performance: Why Zig's Zero-Cost Model Matters for BE

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Chapter 6: Exploit Development — Shellcode, POCs, and ROP Chains

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Shellcode in Zig: Position-Independent Code, No libc Dependencies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Writing Exploit POCs: Structured Exploit Code vs. Ad-Hoc Scripts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

ROP Chain Construction: Building Chains with Comptime Helpers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

ASLR, PIE, and NX: How Modern Mitigations Affect Exploit Design

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

ASLR, PIE, and NX: How Modern Mitigations Affect Exploit Design

Case Study: A Complete Exploitation Chain from Vulnerability to Shell

Chapter 7: Network Security — Sockets, Protocols, and Packet Crafting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Socket Programming in Zig: TCP, UDP, and Raw Sockets

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Protocol Parsing: TCP/IP, DNS, HTTP, and Custom Protocols

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Packet Crafting and Injection: Building Packets from Scratch

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Network Security Tooling: Scanners, Sniffers, and Protocol Analyzers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Real-World Case Studies: Building Security Tools in Zig

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Case Study 1: TCP SYN Scanner

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Case Study 2: DNS Resolver Fuzzer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Case Study 3: Raw Socket Packet Injector

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Trade-offs: When to Use High-Level vs. Low-Level Networking

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Chapter 8: Red Teaming and Stealth Techniques — Evasion at the Binary Level

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Compilation Artifacts: Controlling Binary Signatures, Symbols, and Metadata

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Anti-Analysis Techniques: Obfuscation Through Comptime, Packing, and Control Flow Manipulation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Control Flow Flattening and Instruction Substitution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Steganography in Zig: Embedding Data in Compiled Binaries

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Case Study: A Stealthy C2 Beacon in Zig

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Ethical Considerations: When Evasion Crosses from Research to Abuse

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Chapter 9: Operating System Internals — Kernel-Level Security Tooling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Syscall Wrappers in Zig: Direct Syscall Invocation Without libc

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Building a Minimal Kernel Module: Using Zig's Object-File Output

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Memory-Mapped I/O and Hardware Access: Pointer Arithmetic and Volatile Access

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Case Study: A Kernel-Mode Driver for Security Monitoring

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Platform Differences: Linux, Windows, macOS Support

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Linux: Direct Syscall Invocation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Windows: NtWriteFile and Kernel32 API

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

macOS: Mach Ports and libsystem

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Chapter 10: Detection Evasion — Bypassing Security Software with Zig

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Static Analysis Evasion: Binary Signature Randomization, Symbol Control

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

AMSI and ETW Bypass Concepts: How Runtime Hooking Interacts with Compiled Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Behavioral Stealth: Minimizing Suspicious API Call Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Case Study: A Multi-Stage Loader with Comptime Obfuscation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

The Cat-and-Mouse Game: Detection vs. Evasion Arms Race

Chapter 11: Defensive Countermeasures — Writing Secure Tools in Zig

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Memory Safety in Blue-Team Tools: Using Zig's Allocator Model to Prevent CVEs in Your Own Tools

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Fuzzing with Zig: Building Fast, Deterministic Fuzzers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Static Analysis and Compile-Time Checks: Catching Bugs Before They Reach Production

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Case Study: A Memory-Safe Network Scanner in Zig

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Trade-offs: When Safety Features Add Overhead

Chapter 12: The Future of Zig in Security — Ecosystem, Community, and Roadmap

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

The Zig Ecosystem Today: Package Manager (Zir), Stdlib Maturity, and Third-Party Libraries

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Security Community Adoption: What Tools Exist, What's Missing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

The Road Ahead: Zig 0.14+, Planned Features, and Security-Relevant Improvements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Building Your First Zig Security Tool: A Hands-On Walkthrough

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Final Assessment: Is Zig Ready for Production Security Work?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.

Conclusion: The Right Tool for the Right Job

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/blackhatzig.