

**LLSOFTSECBOOK: LOW-LEVEL
SOFTWARE SECURITY FOR
COMPILER DEVELOPERS**

FOSDEM, FEBRUARY 2022

KRISTOF BEYLS

SUMMARY

What?

An open source book to educate on low-level software security (for compiler developers)

Why?

As compiler developers, we feel we could use more expertise.

There is no good broad educational material. So why not create it ourselves?

How?

Open source: enable more experts to contribute.

Make it freely available: multiply impact.

Status?

Just started public development. First content written. Help (reviews, contributions, ...) very welcome!

Measures of success?







End result: Security features are implemented better in code generation frameworks.

Fewer attacks succeed in breaking hardening features.

MOTIVATION: LACK OF EDUCATIONAL MATERIAL ON LOW-LEVEL SOFTWARE SECURITY

- Many compiler engineers that from time to time work on security features feel they sometimes lack deep expertise. It makes them somewhat unsure about design and trade-off decisions when implementing security features.
 - Results in slower, poorer-quality deployment of support for security features and mitigations for vulnerabilities.
- This seems to be true across the industry.
- We did not find good educational material to help us overcome the lack of expertise.
 - Security researchers tend to publish specific exploits; but there is little material giving an overview of “this is what compiler engineers should know about security related to compilers.”

MOTIVATION: IS IT WORTH IT TO CREATE A BOOK?

- There is no better way to learn about a topic than to try and explain it in simple language to someone else? (see [Feynman Technique](#)).
-  Compiler engineers tend to work more often on security features.
-   Making the book public: more compiler engineers can profit from the book.
-    Making the book open source:
 - We can get input from experts on topics we lack expertise in.
 - It may help strengthen the network of low-level software security experts.
- Book may also help other low-level software developers such as firmware and kernel developers?

WHICH LICENSE?


- Aim to make it easy both to contribute and to consume the book. We chose for the [Creative Commons Attribution \(CC-BY-4.0\) license](#).











CURRENT STATUS: PROJECT IS LIVE AT GITHUB.COM/LLSOFTSEC/LLSOFTSECBOOK/

llsoftsec / llsoftsecbook Public Unwatch 6 Fork 6 Starred 18

<> Code Issues 15 Pull requests 1 Discussions Actions Projects Wiki Security Insights Settings

main 2 branches 1 tag Go to file Add file Code

 **kbeyls** Side-channel and covert channel chapter introduction (#75) ... ✓ 0dff9d7 10 hours ago 🕒 58 commits






 .github/workflows	[ci] Check links in markdown files. (#53)	2 months ago
 docker	Add (empty) bibliography.	3 months ago
 .all-contributorsrc	docs: add jacobbramley as a contributor for ideas (#73)	10 days ago
 .gitignore	Git ignore .vscode directory (#76)	2 days ago
 LICENSE	Add CC-BY-4.0 LICENSE file	3 months ago
 Makefile	Store default pandoc templates in source control	3 months ago
 README.md	docs: add jacobbramley as a contributor for ideas (#73)	10 days ago
 book.bib	#45: Added initial section on supply chain attacks. (#57)	last month
 book.md	Side-channel and covert channel chapter introduction (#75)	10 hours ago
 build_with_docker.sh	Produce a basic HTML version too.	3 months ago

About


Low-Level Software Security for Compiler Developers

llsoftsec.github.io/llsoftsecbook/

security compiler book
compiler-development

 Readme
 View license
 18 stars
 6 watching
 6 forks

Contributors 6



A LIVE COPY OF THE BOOK IS AVAILABLE AT
[LLSOFTSEC.GITHUB.IO/LLSOFTSECBOOK/](https://llsoftsec.github.io/llsoftsecbook/) (PDF)

[1 Introduction](#)

[1.1 Why an open source book?](#)

[2 Memory vulnerability based attacks and mitigations](#)

[2.1 A bit of background on memory vulnerabilities](#)

[2.2 Exploitation primitives](#)

[2.3 Stack overflows](#)

[2.4 Code reuse attacks](#)

[2.5 Non-control data exploits](#)

[2.6 Hardware support for protection against memory vulnerabilities](#)

[2.7 Other issues](#)

[2.8 JIT compiler vulnerabilities](#)

[3 Covert channels and side-channels](#)

[3.1 Cache covert channels](#)

[3.1.1 Typical cache architecture](#)

[3.1.2 Flush + Reload](#)

[3.1.3 Prime + Probe](#)

[3.2 Timing covert channels](#)

[3.3 Resource contention channels](#)

[3.4 Channels making use of aliasing in branch predictors and other predictors](#)

[4 Physical access side-channel attacks](#)

[5 Remote access side-channel attacks](#)

[5.1 Timing attacks](#)

[5.2 Cache side-channel attacks](#)

[6 Supply chain attacks](#)

[6.1 History of supply chain attacks](#)

[7 Other security topics relevant for compiler developers](#)

[Appendix: contribution guidelines](#)

[References](#)

Low-Level Software Security for Compiler Developers



This work is licensed under a [Creative Commons Attribution 4.0 International License](#).

© 2021 Arm Limited kristof.beyls@arm.com

1 Introduction

Compilers, assemblers and similar tools generate all the binary code that processors execute. It is no surprise then that for security analysis and hardening relevant for binary code, these tools have a major role to play. Often the only practical way to protect all binaries with a particular security hardening method is to let the compiler adapt its automatic code generation.

With software security becoming even more important in recent years, it is no surprise to see an ever increasing variety of security hardening features and mitigations against vulnerabilities implemented in compilers.

Indeed, compared to a few decades ago, today's compiler developer is much more likely to work on security features, at least some of their time.

Furthermore, with the ever-expanding range of techniques implemented, it has become very hard to gain a basic understanding of all security features implemented in typical compilers.

This poses a practical problem: compiler developers must be able to work on security hardening features, yet it is hard to gain a good basic understanding of such compiler features.

This book aims to help developers of code generation tools such as JITs, compilers, linkers and assemblers to overcome this.

There is a lot of material that can be found explaining individual vulnerabilities or attack vectors. There are also lots of presentations explaining specific exploits. But there seems to be a limited set of material that gives a structured overview of all vulnerabilities and exploits for which a code generator could play a role in protecting against them.

This book aims to provide such a structured, broad overview. It does not necessarily go into full details. Instead it aims to give a thorough description of all relevant high-level aspects of attacks.

CURRENT STRUCTURE OF THE BOOK

1. Introduction
2. Memory vulnerability based attacks and mitigations
3. Covert channels and side-channels
4. Physical access side-channel attacks
5. Remote access side-channel attacks
6. Supply chain attacks
7. Other security topics relevant for compiler developers

LOOKING FOR HELP

The project is at its start - maybe 5% of the content is present. We can use a lot of help and contributions:

- Ideas for new content.
- Any other ideas on how to improve the book.
- Review of existing content and newly written content in pull requests.
- Contributions of new sections or improvements to the text.
- Introduce your friends to this project.

How to reach out:

- Raise a github issue or start a discussion thread at github.com/llsoftsec/llsoftsecbook/
- Reach out to me in any way, e.g. by email at kristof.beyls at gmail.com