Для книги **"RUST Очима хакера"** (українською мовою) я пропоную структуру змісту, яка враховує базові концепції Rust, а також їхню інтерпретацію з точки зору безпеки, експлуатації та захисту систем. Книга буде складатися з **15–20 заголовків**, розбитих на логічні розділи. Ось приклад змісту:

Зміст книги "RUST Очима хакера"

Частина 1: Вступ до Rust
1. **Що таке Rust? Чому його обирають хакери?**
   - Огляд мови програмування Rust.
   - Безпека та продуктивність як ключові переваги.
   - [Посилання](https://www.rust-lang.org/what)

2. **Історія Rust та його вплив на безпеку систем**
   - Виникнення Rust у Mozilla.
   - Як Rust допомагає запобігати уразливостям (buffer overflows, memory leaks).
   - [Посилання](https://doc.rust-lang.org/book/ch00-00-introduction.html)

3. **Встановлення Rust та перша програма**
   - Налаштування середовища (`rustup`, `cargo`).
   - Перша програма "Hello, World!" та аналіз її безпеки.
   - [Посилання](https://doc.rust-lang.org/book/ch01-02-hello-world.html)

Частина 2: Безпека та влада над пам'яттю
4. **Концепція Ownership (Володіння)**
   - Як Rust забезпечує безпечну роботу з пам'яттю.
   - Приклади уразливостей, яких Rust уникнутий (порівняння з C/C++).
   - [Посилання](https://doc.rust-lang.org/book/ch04-00-understanding-ownership.html)

5. **Borrowing (Позичання) та References (Посилання)**
   - Правила borrowing та їхня роль у запобіганні data races.
   - Атаки через неправильне управління посиланнями.

6. **Lifetimes (Життєвий цикл)**
   - Як Rust контролює час життя даних.
   - Приклади потенційних проблем без lifetime checks.

7. **Safe vs Unsafe Rust**
   - Коли варто використовувати `unsafe`.
   - Ризики та можливість експлуатації.

Частина 3: Паралельність та конкурентність
8. **Паралельність в Rust**

   - Захищеність від data races завдяки системі типів.
   - Приклади паралельних програм.

9. **Threads та Message Passing**
   - Безпечна взаємодія між потоками.
   - Аналіз атак на паралельні системи.

10. **Асинхронне програмування**
    - Async/await в Rust.
    - Безпека асинхронного коду.

Частина 4: Хакерський підхід до Rust
11. **Аналіз уразливостей в Rust**
    - Як Rust захищає від типових уразливостей (buffer overflows, UAF).
    - Реальні приклади багів в Rust-проектах.

12. **Fuzzing та тестування безпеки**
    - Використання fuzzing для знаходження уразливостей.
    - Інструменти: `cargo-fuzz`, `afl.rs`.

13. **Експлуатація уразливостей в Rust**
    - Можливість експлуатації `unsafe` блоків.
    - Приклади реальних CVE в Rust-бібліотеках.

14. **Реверс-інжиніринг Rust-бінарників**
    - Особливості дизасемблювання Rust-коду.
    - Інструменти: Ghidra, Radare2.

Частина 5: Практичні аспекти
15. **Створення безпечного веб-сервера на Rust**
    - Використання фреймворків (`actix-web`, `rocket`).
    - Безпека веб-додатків.

16. **Робота з криптографією в Rust**
    - Бібліотеки: `ring`, `sodiumoxide`.
    - Безпечна генерація ключів та шифрування.

17. **Написання експлоїтів для Rust-систем**
    - Аналіз захисту Rust перед експлоїтами.
    - Практичні приклади.

18. **Аудит безпеки Rust-коду**
    - Інструменти для аналізу коду: `clippy`, `rust-analyzer`.
    - Методики ручного аудиту.

Частина 6: Додаткові теми
19. **Rust в системному програмуванні**
    - Написання операційних систем та драйверів.
    - Безпека низькорівневого коду.

20. **Майбутнє Rust та безпека**
    - Розвиток Rust в контексті кібербезпеки.
    - Прогнози та нові функції.

Розрахунок кількості сторінок
- **Кількість заголовків**: 20.
- **Середня кількість сторінок на розділ**: 20–25 сторінок.
- **Загальна кількість сторінок**: **400–500 сторінок**.

Додаткові ресурси
1. [The Rust Programming Language Book](https://doc.rust-lang.org/book/)
2. [Rust by Example](https://doc.rust-lang.org/rust-by-example/)
3. [Exercism.io (Rust Track)](https://exercism.org/tracks/rust)
4. [Unsafe Rust Guidelines](https://doc.rust-lang.org/nomicon/)
5. [Fuzzing in Rust](https://github.com/rust-fuzz/cargo-fuzz)

Цей зміст охоплює широкий спектр тем, які будуть корисні як для початківців, так і для досвідчених програмістів, які хочуть поглянути на Rust з точки зору безпеки.