작은 헤더

제목: 블랙 햇 러스트
적용된 공격 보안과 러스트 프로그래밍 언어

저자: 실뱅 케르쿠르 page -1-

작은 제목

블랙 햇 러스트
러스트 프로그래밍 언어를 활용한 공격 보안

Sylvain Kerkour

v2022.56

page -2-

작은 제목
목차

저작권
귀하의 조기 액세스 보너스
연락처
서문

1. 소개
   1.1 공격의 유형 14
   1.2 공격의 단계 16
   1.3 공격자의 프로필 17
   1.4 귀속 18
   1.5 러스트 프로그래밍 언어 19
   1.6 러스트의 역사 19
   1.7 러스트는 멋지다 20
   1.8 설정 24
   1.9 우리의 첫 번째 러스트 프로그램: SHA-1 해시 크래커 25
   1.10 러스트 접근을 위한 정신 모델 32
   1.11 내가 배운 몇 가지 34
   1.12 요약 42

2. 다중 스레드 공격 표면 발견
   2.1 수동 정찰 43
   2.2 능동 정찰 44
   2.3 자산 발견 44
   2.4 우리의 첫 번째 스캐너 46

2.5 오류 처리 46
2.6 서브도메인 열거 47
2.7 포트 스캔 48
2.8 다중 스레딩 50
2.9 러스트의 두려움 없는 동시성 51
2.10 데이터 경합의 세 가지 원인 54
2.11 소유권의 세 가지 규칙 54
2.12 참조의 두 가지 규칙 54
2.13 기타 동시성 문제 54

page -3-

2.14 Adding multithreading to our scanner 54

2.15 Alternatives 57

2.16 Going further 58

2.17 Summary 58

3 Going full speed with async 60
3.1 Why 60
3.2 Cooperative vs Preemptive scheduling 61
3.3 Future 62
3.4 Streams 62
3.5 What is a runtime 63
3.6 Introducing tokio 63
3.7 Avoid blocking the event loops 66
3.8 Sharing data 67
3.9 Combinators 71
3.10 Porting our scanner to async 85
3.11 How to defend 90
3.12 Summary 90

4 Adding modules with trait objects 92
4.1 Generics 93
4.2 Traits 95
4.3 Traits objects 97
4.4 Command line argument parsing 105
4.5 Logging 106
4.6 Adding modules to our scanner 107
4.7 Tests 115

4.8 Other scanners 119
4.9 Summary 119

5 Crawling the web for OSINT 120
5.1 OSINT 120
5.2 Tools 120
5.3 Search engines 121
5.4 IoT & network Search engines 123
5.5 Social media 123
5.6 Maps 124
5.7 Videos 124
5.8 Government records 124
5.9 Crawling the web 125
5.10 Why Rust for crawling 126
5.11 Associated types 127
5.12 Atomic types 128
5.13 Barrier 130
5.14 Implementing a crawler in Rust 131
5.15 The spider trait 131
5.16 Implementing the crawler 131

5.17 간단한 HTML 웹사이트 크롤링
5.17 간단한 HTML 웹사이트 크롤링 ……. 136

5.18 JSON API 크롤링
5.18 JSON API 크롤링 …….. 138

5.19 JavaScript 웹 애플리케이션 크롤링
5.19 JavaScript 웹 애플리케이션 크롤링 …….. 141

5.20 방어 방법
5.20 방어 방법 …….. 143

5.21 더 나아가기
5.21 더 나아가기 …… 145

5.22 요약
5.22 요약 …… 146

6 취약점 찾기
6.1 취약점이란 무엇인가 ……. 147
6.2 약점 vs 취약점 (CWE vs CVE) …….. 147

6.3 취약점 vs 익스플로잇 148
6.4 0 Day vs CVE 148
6.5 웹 취약점 148
6.6 주입 공격 149
6.7 HTML 주입 149
6.8 SQL 주입 150
6.9 XSS 152
6.10 서버 측 요청 위조 (SSRF) 156
6.11 교차 사이트 요청 위조 (CSRF) 157
6.12 오픈 리다이렉트 159
6.13 (서브) 도메인 탈취 160
6.14 안의 파일 쓰기 161
6.15 서비스 거부 (DoS) 163
6.16 안의 파일 쓰기 164
6.17 메모리 취약점 165
6.18 버퍼 오버플로우 166
6.19 사용 후 해제 167
6.20 이중 해제 168
6.21 기타 취약점 169
6.22 원격 코드 실행 (RCE) 169
6.23 정수 오버플로우 (및 언더플로우) 170
6.24 논리 오류 171
6.25 경쟁 조건 172
6.26 추가 자료 172
6.27 버그 헌팅 172
6.28 도구들 174
6.29 자동 감사 176
6.30 요약 181

7 익스플로잇 개발
7.1 익스플로잇을 찾는 곳 182
7.2 라이브러리이자 바이너리인 크레이트 만들기 183
7.3 libc 184
7.4 익스플로잇 툴킷 구축 185
7.5 CVE-2019-11229 && CVE-2019-89242 185
7.6 CVE-2021-3156 185
7.7 요약 191

8 Writing shellcodes in Rust
8.1 What is a shellcode ……. 192
8.2 Sections of an executable ……. 194
8.3 Rust compilation process …….. 194
8.4 no_std ……. 195
8.5 Using assembly from Rust …….. 197
8.6 The never type …….. 198
8.7 Executing shellcodes …… 198
8.8 Our linker script …… 200
8.9 Hello world shellcode …….. 201
8.10 An actual shellcode …… 204
8.11 Reverse TCP shellcode …… 211
8.12 Summary …….. 215

9 Phishing with WebAssembly
9.1 Social engineering …… 216
9.2 Nontechnical hacks …….. 220
9.3 Phishing …… 221
9.4 Watering holes …….. 222
9.5 Telephone ……. 225
9.6 WebAssembly …….. 225
9.7 Sending emails in Rust …… 226
9.8 Implementing a phishing page in Rust …….. 231
9.9 Architecture …….. 231
9.10 Cargo Workspaces …….. 231
9.11 Deserialization in Rust …….. 232
9.12 A client application with WebAssembly …… 233
9.13 Evil twin attack …… 243
9.14 How to defend …….. 246
9.15 Summary …….. 248

10 A modern RAT
10.1 Architecture of a RAT ……. 249
10.2 C&C channels & methods …….. 251
10.3 Existing RAT …… 253
10.4 Why Rust …….. 254
10.5 Designing the server …….. 255
10.6 Designing the agent …… 264
10.7 Docker for offensive security …… 265
10.8 Let’s code …….. 266
10.9 Optimizing Rust’s binary size …… 287

10.10 Dockerizing the server …….. 288
10.11 Some limitations …….. 290
10.12 Summary …….. 290

11 Securing communications with end-to-end encryption
11.1 The C.I.A triad …….. 291

11. 암호학
11.2 위협 모델링 …… 293
11.3 암호화 …….. 294
11.4 해시 함수 …….. 294
11.5 메시지 인증 코드 …….. 294
11.6 키 파생 함수 …….. 296
11.7 블록 암호 …….. 296
11.8 인증된 암호화 (AEAD) …….. 297
11.9 비대칭 암호화 …….. 299
11.10 디피-헬만 키 교환 …… 300
11.11 서명 …… 300
11.12 종단 간 암호화 …… 301
11.13 누가 암호화를 사용하는가 ……. 310
11.14 암호화의 일반적인 문제와 함정 …….. 311
11.15 TOFU의 작은 부분? …….. 312
11.16 러스트 암호화 생태계 …… 312
11.17 요약 …… 314
11.18 우리의 위협 모델 …….. 314
11.19 프로토콜 설계 ……. 315
11.20 러스트에서 종단 간 암호화 구현 …….. 319
11.21 몇 가지 제한 사항 …… 330
11.22 더 알아보기 …… 331
11.23 요약 …… 332

12. 다중 플랫폼으로 가기
12.1 왜 다중 플랫폼인가 …… 333
12.2 크로스 플랫폼 러스트 ……. 334
12.3 지원되는 플랫폼 …….. 335
12.4 크로스 컴파일 ……. 336
12.5 크로스 …….. 337
12.6 사용자 정의 도커파일 …… 339
12.7 aarch64(arm64)로 크로스 컴파일 …….. 340
12.8 더 많은 러스트 바이너리 최적화 팁 ……. 341

12.9 패커 341
12.10 지속성 342
12.11 단일 인스턴스 347
12.12 더 나아가기 348
12.13 요약 348

13. 우리의 RAT을 벌레로 변환하여 법위을 늘리기
13.1 벌레란 무엇인가 349
13.2 전파 기술 350
13.3 크로스 플랫폼 벌레 352
13.4 SSH을 통한 전파 353
13.5 의존성 공급 354
13.6 러스트에서 크로스 플랫폼 벌레 구현 355
13.7 설치 356
13.8 전파 357

page -7-

작은 제목
13.9 더 발전된 RAT 기술 362
13.10 요약 366

14 결론 367
14.1 우리가 다루지 않은 내용 367
14.2 Rust의 미래 369
14.3 유출된 저장소 369
14.4 나쁜 사람들이 잡히는 방법 369
14.5 당신의 차례 370
14.6 나만의 RAT 만들기 373
14.7 다른 흥미로운 블로그 374
14.8 연락처 374

page -8- # 저작권

실수나 개선할 점이 발견되면, 또는 공격 보안에 대한 아이디어를 공유하고 싶으시면, GitHub에서 논의에 참여해 주세요: https://github.com/skerkour/black-hat-rust page -10-

연락처
이 책과 보안적인 내용을 정기적으로 제 뉴스레터에 게시합니다.

매주 제 프로젝트와 재머와 이익을 위해 기술을 (악)용하는 방법에 대해 배우는 모든 내용을 공유합니다: 프로그래밍, 해킹 및 기업가 정신. 이메일 또는 RSS로 구독할 수 있습니다: https://kerkour.com/subscribe.

page -11- # 서문

고등학교를 졸업한 후, 내 인생 계획은 사실 탐정이 되는 것이었습니다. 아마도 셜록 홈즈 책을 너무 많이 읽었기 때문일 것입니다. 프랑스에서 사법대학에 가는 가장 쉬운 방법은 전문 학교에 진학하는 것이었습니다.

나는 준비가 되어 있지 않았습니다.

법학을 공부하는 것은 나에게 맞지 않다는 것을 금방 깨달았습니다. 현실은 교수들이 멀게 하려는 내러티브 정치에 왜곡되어 있었습니다. 여기서는 깊은 지식이 가르쳐지지 않으며, 단지 숫자, 날짜, 멋져 보이고 똑똑하게 들리는 방법만 배웁니다. 나는 세상이 어떻게 작동하는지 이해하고 싶었습니다. 예를 들어, 우리가 하루 종일 열심히 타이핑하는 이 기계들은 어떻게 작동하는 걸까요?

그래서 나는 리눅스를 설치하기 시작했습니다. (아니, GNU/Linux 전쟁에 들어가지는 않을 것입니다.) 내 Asus EeePC에 1GB RAM만 있는 작은 넷북에 리눅스를 설치했습니다. 윈도우가 너무 느려서 C++ 프로그램을 Qt로 개발하기 시작했습니다. 온라인 튜토리얼 덕분에 내 텍스트와 나만의 채팅 시스템을 코딩했습니다. 하지만 내 호기심은 충족되지 않았습니다.

어느 날, 내 인생을 바꾼 책을 우연히 발견했습니다: “Hacking: The Art of Exploitation, 2nd Edition” 저자: Jon Erickson.

이 책은 내가 사물을 만드는 것에 대한 호기심을 불러일으켰을 뿐만 아니라, 더 중요한 것은 사물을 부수는 방법에 대한 호기심도 불러일으켰습니다. 신뢰할 수 있는 것을 구축하려면 그것을 부수는 방법을 이해해야 한다는 것을 깨달았습니다.

이 책은 저수준 프로그래밍과 메모리 안전 버그를 이용하는 방법을 배우기에 훌륭하지만, 오늘날에는 웹 취약점, 네트워크 및 시스템 프로그래밍, 그리고 현대 프로그래밍 언어로 코딩하는 새로운 기술이 필요합니다.

러스트와 공격 보안의 매혹적인 세계에 오신 것을 환영합니다.

러스트 책은 러스트가 무엇인지 가르치는 데 훌륭한 작업을 수행하지만, 러스트에 대한 이유와 방법을 다룬 책이 부족하다고 느꼈습니다. 이는 일부 개념이 누락되었음을 의미합니다. page -12- # 효과적인 사용법

이 책에서는 선입견을 깨고, Rust가 현실 세계에서 너무 복잡하다는 생각이나 비생산적이라는 오해를 불식시키며, 공격 보안에 적용된 실제 Rust 프로젝트를 설계하고 만드는 방법을 살펴볼 것입니다. 우리는 다재다능한 Rust가 어떻게 사용자에게 고수준 추상화, 높은 성능, 필요할 때 저수준 제어를 제공하는 독특한 언어로서 여러 프로그래밍 언어(Python, Ruby, C, C++ 등)를 대체할 수 있는지를 알아볼 것입니다.

우리는 항상 이론, 즉 시대를 초월한 깊은 지식을 바탕으로 시작할 것입니다. 이 지식은 특정 프로그래밍 언어와 무관하며, 공격 보안에 필요한 올바른 사고방식을 갖추는 데 도움이 될 것입니다.

이 책은 공격자의 사고 방식을 이해하고자 하거나 공격 보안 세계에 진입하고자 하는 사람들을 위해 설계되었습니다. 궁극적으로는 이를 통해 생계를 유지할 수 있기를 바랍니다.

이 책의 목표는 여러분이 행동으로 나아가는 데 필요한 시간을 절약하고, 지식을 정제하여 적용된 코드 프로젝트로 제시하는 것입니다.

Black Hat Rust는 세상의 모든 지식을 포함하는 대백과사전이 아니라, 여러분이 시작할 수 있도록 돕고 행동으로 나아가는 길을 열어주기 위해 설계되었습니다. 지식은 종종 전제 조건이지만, 행동이 세상을 형성하며 때로는 지식이 행동의 장애물이 될 수 있습니다. 우리는 가장 원시적인 공격 기법이 여전히 가장 효과적이라는 것을 알게 될 것입니다. 따라서 현대 OS의 보호 메커니즘을 우회하는 방법과 같은 매우 구체적인 주제는 이미 방대한 문헌이 존재하므로 다루지 않을 것입니다. 그럼에도 불구하고, 여러분의 학습 여정을 돕기 위해 최선의 자료를 나열했습니다.

Rust에 능숙해지기까지 약 1년이 걸렸지만, 내가 많은 코드를 작성(및 재작성)하기 시작했을 때 비로소 진전을 이룰 수 있었습니다. Rust는 매우 강력한 언어이지만, 실제로는 여러분이 사용할 수 있는 기능의 하위 집합만 필요합니다. 모든 것을 미리 배울 필요는 없으며, 이 책에서 다룰 기본적인 것들이 있습니다. 다른 것들은 그렇지 않으며, 코드의 가독성과 유지 관리의 어려움에 부정적인 영향을 미칠 수 있습니다.

이 책의 의도는 여러분이 공격 보안의 매혹적인 세계를 발견하도록 돕고, Rust가 공격 보안의 모든 요구를 충족하는 오랜 기다림의 만능 프로그래밍 언어임을 확신시키는 것입니다. 또한 여러분의 시간을 절약하는 데 도움을 주고자 합니다. page-13- # 작은 헤더

제목 강조: 학습의 중요성

시간을 절약할 수 있도록 Rust와 공격 보안 학습에서 중요한 것에 집중하게 해줍니다. 하지만 기억하세요, 지식만으로는 충분하지 않습니다. 지식은 산을 움직이지 않습니다. 행동이 필요합니다.

따라서 이 책은 이야기의 절반에 불과합니다. 나머지 절반은 동반 코드 저장소입니다: https://github.com/skerkour/black-hat-rust. 실습 없이는 배울 수 없으니, 코드를 읽고 수정하여 여러분의 것으로 만들어 보세요!

Rust 코드의 일부를 이해하지 못하거나 길을 잃었다면, Rust 언어 요약표, Rust 책, Rust 언어 참조를 참조하세요.

또한, 이 책은 코드가 많습니다. 웹 브라우저를 옆에 두고 읽는 것을 추천합니다. GitHub에서 코드를 탐색하고 실험해 보세요: https://github.com/skerkour/black-hat-rust. page -14- # Chapter 1
Introduction

"충분히 발전된 사이버 공격은 마법과 구별할 수 없다", 미상

영화나 주류 미디어에서 해커는 종종 로맨틱하게 묘사됩니다: 그들은 검은 마법사, 악랄한 범죄자, 혹은 최악의 경우 후드와 어지렛대를 가진 도둑으로 그려집니다.

실제로 공격자의 프로필 스펙트럼은 매우 넓습니다. 지루한 십대가 주권 국가의 군대와 불행한 전직 직원들을 탐색하는 것부터 시작합니다. 사이버 공격은 그렇게 어렵지 않습니다. 지식은 단순히 기존 행위자들에 의해 불균형하게 분배되고 직투 어린 비밀로 유지됩니다. 호기심과 본능을 따를 용기가 주요 요소입니다.

디지털이 우리의 삶에서 점점 더 중요한 위치를 차지함에 따라 사이버 공격의 영향과 규모는 증가할 것입니다. 우리는 현재 COVID-19 팬데믹 동안 병원에 대한 공격을 목격하고 있습니다. 이는 실제적이고 극적인 결과를 초래합니다.

이제 우리는 오늘날의 전쟁과 전투에 대비할 때입니다(내일이 아니라). 방어하기 위해서는 공격자의 입장에서 생각해야 한다는 것을 이해해야 합니다. 그들은 어떻게 그렇게 쉽게 시스템에 침입할 수 있을까요? 그들의 희생자에게 무엇을 할까요? 이론에서 실천으로, 우리는 공격 보안의 비밀을 탐구하고 Rust 프로그래밍 언어로 우리의 공격 도구를 구축할 것입니다.

왜 Rust인가?
보안(그리고 더 일반적으로 소프트웨어)의 세계는 너무 많은 프로그래밍 언어로 인해 혼란스러워합니다. 빠르고 안전하지 않은 언어(C, C++ 등)와 느리지만 대체로 안전한 언어(Python, Java 등) 중에서 선택해야 합니다. page -15-