PROBA TEKNIKOA: Cryptopals-eko ariketak

Olatz Madinabeitia Etxeberria

Aurkibidea

1 Nire aurkezpena

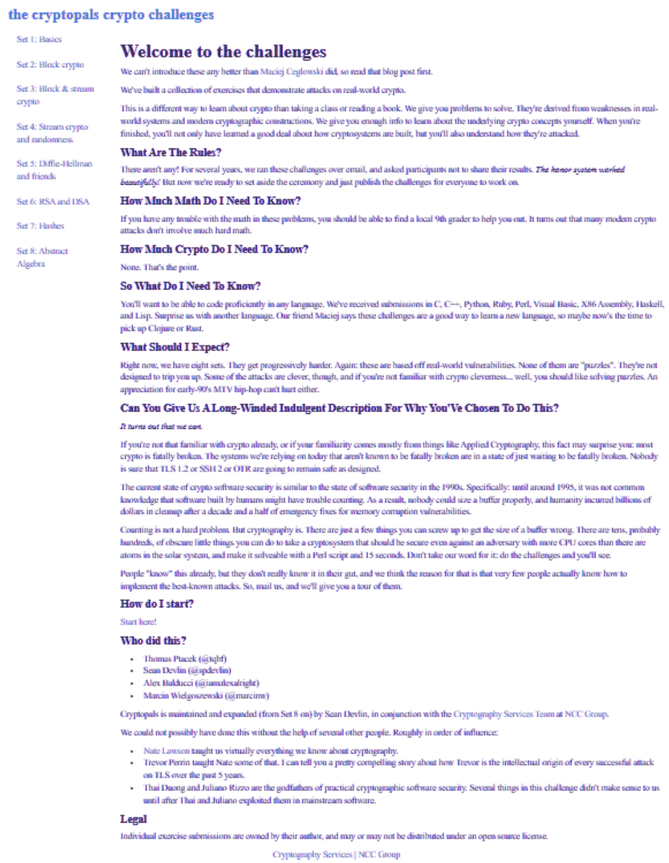
Ikasketak

- Telekomunikazio masterreko ikasketak
  - MLSecOps-en inguruko MaL

Helburu profesionalak

- Profesionalki formatu
- Ingurune kolaboratibo eta atsegin batean lan egitea

2 Lanaren aurkezpena



the cryptopals crypto challenges

Set 1: Basics

Set 2: Block crypto

Set 3: Block & stream crypto

Set 4: Stream crypto and randomness

Set 5: Diffie-Hellman and friends

Set 6: RSA and DSA

Set 7: Hashes

Set 8: Abstract Algebra

Welcome to the challenges

We can't introduce these any better than Maciej Ceglowski did, so read that blog post first.

We've built a collection of exercises that demonstrate attacks on real-world crypto.

This is a different way to learn about crypto than taking a class or reading a book. We give you problems to solve. They're derived from weaknesses in real-world systems and modern cryptographic constructions. We give you enough info to learn about the underlying crypto concepts yourself. When you're finished, you'll not only have learned a good deal about how cryptosystems are built, but you'll also understand how they're attacked.

What Are The Rules?

There aren't any! For several years, we ran these challenges over email, and asked participants not to share their results. *The honor system worked beautifully!* But now we're ready to set aside the ceremony and just publish the challenges for everyone to work on.

How Much Math Do I Need To Know?

If you have any trouble with the math in these problems, you should be able to find a local 9th grader to help you out. It turns out that many modern crypto attacks don't involve much hard math.

How Much Crypto Do I Need To Know?

None. That's the point.

So What Do I Need To Know?

You'll want to be able to code proficiently in any language. We've received submissions in C, C++, Python, Ruby, Perl, Visual Basic, X86 Assembly, Haskell, and Lisp. Surprise us with another language. Our friend Maciej says these challenges are a good way to learn a new language, so maybe now's the time to pick up Clojure or Rust.

What Should I Expect?

Right now, we have eight sets. They get progressively harder. Again: these are based off real-world vulnerabilities. None of them are "puzzles". They're not designed to trip you up. Some of the attacks are clever, though, and if you're not familiar with crypto cleverness... well, you should like solving puzzles. An appreciation for early-90's MTV hip-hop can't hurt either.

Can You Give Us A Long-Winded Indulgent Description For Why You've Chosen To Do This?

It turns out that we can.

If you're not that familiar with crypto already, or if your familiarity comes mostly from things like Applied Cryptography, this fact may surprise you: most crypto is fatally broken. The systems we're relying on today that aren't known to be fatally broken are in a state of just waiting to be fatally broken. Nobody is sure that TLS 1.2 or SSH 2 or OTR are going to remain safe as designed.

The current state of crypto software security is similar to the state of software security in the 1990s. Specifically: until around 1995, it was not common knowledge that software built by humans might have trouble counting. As a result, nobody could size a buffer properly, and humanity incurred billions of dollars in cleanup after a decade and a half of emergency fixes for memory corruption vulnerabilities.

Counting is not a hard problem. But cryptography is. There are just a few things you can screw up to get the size of a buffer wrong. There are tens, probably hundreds, of obscure little things you can do to take a cryptosystem that should be secure even against an adversary with more CPU cores than there are atoms in the solar system, and make it solveable with a Perl script and 15 seconds. Don't take our word for it: do the challenges and you'll see.

People "know" this already, but they don't really know it in their gut, and we think the reason for that is that very few people actually know how to implement the best-known attacks. So, mail us, and we'll give you a tour of them.

How do I start?

Start here!

Who did this?

- Thomas Pacek (@tqbf)
- Sean Devlin (@spdevlin)
- Alex Balducci (@iamalexalright)
- Marcin Wielgoszewski (@marcinw)

Cryptopals is maintained and expanded (from Set 8 on) by Sean Devlin, in conjunction with the Cryptography Services Team at NCC Group.

We could not possibly have done this without the help of several other people. Roughly in order of influence:

- Nate Lawson taught us virtually everything we know about cryptography.
- Trevor Perrin taught Nate some of that. I can tell you a pretty compelling story about how Trevor is the intellectual origin of every successful attack on TLS over the past 5 years.
- Thai Duong and Juliano Rizzo are the godfathers of practical cryptographic software security. Several things in this challenge didn't make sense to us until after Thai and Juliano exploited them in mainstream software.

Legal

Individual exercise submissions are owned by their author, and may or may not be distributed under an open source license.

Cryptography Services | NCC Group

https://cryptopals.com/-eko lehenengo 3 Set-ak egitea:

- Set 1: Basikoa:
  - Hex ↔ Base64 bihurketak
  - XOR eragiketak
  - ECB zifratuen detekzioa
- Set 2: Crypto blokea
  - AES CBC moduan inplementatzea
  - PKCS#7 padding eta padding orakuluak
  - ECB vs CBC detekzioa
- Set 3: Crypto eta stream blokea
  - CTR modua eta bere erabilera
  - Estatistika bidezko erasoak
  - MT19937 PRNG: egoeraren klonazioa eta berreskuratzea
  - Stream cipher bat MT19937 erabiliz eta bere aurkako erasoak
  - Token faltsutzea denboran oinarritutako seed erabiliz

3 Erabilitako teknologia

- **Programazio lengoaia:** C++
- **Build system:** CMake
- **Testing:** GoogleTest
- **Criptografía:** OpenSSL

4 Proiektuaren antolaketa

Proiektua **3 Set-etan** banatu da eta barnean hurrengo antolaketa:

Set/

- **include/** → Header fitxategiak (.hpp): interfazeak eta definizioak
- **src/** → Inplementazio nagusiak (.cpp): algoritmoak eta logika
- **tests/** → Unit test-ak GoogleTest erabiliz
- **data/** → Sarrerako fitxategiak (.txt)
- **CMakeLists.txt** → Eraikuntza konfigurazioa CMake bidez

5 Datu interesgarriak

XOR	Kriptografian oinarrizko eragiketa binarioa da. Datuak modu simetrikoan zifratzeko eta deszifratzeko erabiltzen da: mezu XOR gakoa=zifratua. Gakoa ezagutuz gero, eragiketa alderantziz egin daiteke.
ECB vs CBC	ECB (Electronik Codebook) bloke bakoitza banaka zifratzen du eta horrek ereduak detektatzea ahalbidetzen du. CBC (Cipher Block Chaining) blokeak kateatzen ditu, ereduak ezkutatuz eta segurtasuna hobetuz. ECB ikusizko analiaren aurrean ahula da.
Padding oracle	Padding okerra duten mezu zifratuak ustiatzen dituen erasotzeko teknika da. Gakoa ezagutu gabe datuak deszifratzeko aukera ematen du, sistemaren erantzunak "orakulu" gisa erabiliz.
CTR mode	Bloke bidezko zifratzailea fluxu bidezko bihurtzen du. Zenbatzaile zifratua keystream gisa erabiltzen du, mezuarekin XOR bidez konbinatzen du. Zifratze paraleloa eta simetrikoa ahalbidetzen du.
MT19937	Oso azkarra den zenbaki pseudo-aleatorioen sortzailea da, sistema askotan erabiltzen dena. Ez da kriptografikoki segurua, bere barne egoera nahikoa irteera behatuz berreskuratu daitekeelako.
Token Forgery	MT19937 aurreikus daitekeen hazi batekin (adibidez, uneko ordua) erabiltzen da, token faltsuak sr daitezke benetako diruditenak. Honek PRNG (Pseudorandom Number Generator) seguruak erabiltzeko garrantzia erakusten du.
CMake & GTest	CMake C++ proiektuak modu modularrean eta eramangarrian konpilatzeko erabiltzen da. GoogleTest erronka bakoitza automatikoki bidaltzeko erabiltzen da, test unitarioen bidez, kodea ondo funtzionatzen duela ziurtatuz.

5 Praktikoa

SET 1: hex to base64

```
student@vbox:~/Ikerlan/Set1/build$ ./hex_to_base64
SSdtIGtpbGxpbmcgeW91ciBicmFpbiBsaWtlIGEgcG9pc29ub3VzIG11c2hyb29t
student@vbox:~/Ikerlan/Set1/build$ ./tests/run_tests
Running main() from ./googletest/src/gtest_main.cc
[==========] Running 1 test from 1 test suite.
[----------] Global test environment set-up.
[----------] 1 test from HexToBase64Test
[ RUN      ] HexToBase64Test.BasicConversion
[       OK ] HexToBase64Test.BasicConversion (0 ms)
[----------] 1 test from HexToBase64Test (0 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test suite ran. (0 ms total)
[  PASSED  ] 1 test.
```

Biltegi honetan aurkitu daitezke: https://github.com/omadinabeitia001/Ikerlan

6 Ondorioak

- Kriptografiaren oinarriak ulertu ditut, eta sistema errealak nola apurtzen diren ikasi dut
- IRITZIA: Gaia interesgarriak iruditu zait baina oraindik familiarizatu egin beharko nintzateke eta teoria gehiago jakin.

Mila esker

Galderarik?