Sharpening my InfoSec tools

Justus Perlwitz

2017-06-30

Contents

After a long time of being complacent with my skills, I thought I should up my InfoSec game. So far I've been mainly busy with figuring out how to enhance application security in my work. That means I learned how to

- Implement role and permission systems,
- Create fail-secure software that is least permissible by default,
- Implement the newest recommendations on enumeration, XSS, CSRF and untrustworthy user input in general.

But from time to time I notice that my house is built on a weak fundament. What I particularly lack is deep knowledge of what happens down there in OSI 1-4. While my work has allowed to twiddle with UDP and discover the merits and dangers of UDP-Lite, I have not really had a chance to try to break something on purpose by interacting with applications on such a low level.

Not only that, I think that the more we start laying brick upon brick in this evermore growing information society, the more we keep building leaky abstractions and start engaging in meaningless cargo cult rituals. By this I mean we start implementing certain 'secure ways' of implementing application, not based on first-hand knowledge, but tertiary sources that tell us to do so.

I think a great example of leaky abstractions is a writeup by CloudFlare on DDoS amplification through SSDP. You can be certain that there was more than one crappy WiFi light bulb involved in that incident.

So let's be concrete here. I've identified the following areas where I want to improve my skills and I would like to share them with you. I've sorted them by how important they are to me both short as well as long term.

1 Bit-Twiddling

Having an understanding of CPU-level handling of information means understanding the foundation of what your computer does. Now, I know how general CPU architectures work and I've certainly worked my way through one FPGA course at university and a brilliant book called The Elements of Computing Systems. Nothing says "I understand machines" like dreaming in opcodes.

So here's what can be learned:

- FPGA Design. Nothing says "I SPEAK BINARY" like pulling off a CPU design. I'm thinking about recreating the processor from Charles Petzold's book *Code*. (~100h)
- Do all 64 xorpd challenges. (~30h)

2 Reverse Engineering

I've always been interested in challenges like MicroCorruption but have never gotten around to actually finishing them. Furthermore, the Reverse Engineering Challenges by Dennis Yurichev seem very interesting.

So here's how this one goes:

- MicroCorruption (~100h)
- Reverse Engineering Challenges (~200h)

3 Application Security

I've already done two different tracks on OverTheWire and I've learned a lot about securing application servers and operating systems. It is necessary nowadays to understand the **full** stack, and not only one's own comfortable territory. The challenges always follow the same format: You have to find the SSH keys or the password for the next stage's server.

For this one I will estimate ~100h of work.

4 Applied Cryptography

While I have already started the Matasano Crypto Challenge, I've never really finished it. To me, it is just the right amount of cryptography mixed with application engineering.

For this one the work will be another 100h.

5 Crypto Puzzles

This is for anything that does not fit in the other categories. Combinatorics, theoretical CompSci and number theory will always play a big role in InfoSec. Therefore, I want to allocate some time to getting my hands dirty with algos and what not. A few resources are interesting to me:

- WeChall has a lot of crypto puzzles (~50h).
- Project Euler is impossible to solve completely, for there is an incredible amount of puzzles. But getting somewhere to solving at least 40 seems like a formidable goal (~100h).

6 Open Source Contributions

Too many open source projects do not receive regular contributions to improve security. Too many web applications (I'm looking at you, PHP) are still vulnerable to the OWASP top 10. So, a valuable contribution can be made easily. I want to look for an open source project on GitHub, try installing and operating it. Then I will look for weaknesses and contribute back patches.

I estimate this will amount to around 100 hours of works over 20 weeks.

7 My Method

I will pick a topic every week and continue working on it. In order to implement a puzzle solution I will pick a new language that I'm not yet comfortable with. Haskell comes to my mind here. Furthermore, I shall regularly post on this blog and track my progress. I will not of course disclose the actual solutions I find for puzzles. That would destroy the fun. But writeups with nothing more than vague hints should be acceptable.

8 Summary

How do we continue from here? I suppose I should keep my dear readers updated on this very blog. I shall speak to you again in a week from now and let you know about my progress.