Workshop Self-Evaluation: "How CTF Works" by Antranig Vartanian
ARMSec Conference, 16:00-17:00

I'll admit, walking into the workshop was at first intimidating, as I knew nothing about CTFs and had found out about it upon reading the name of the workshop. But the session turned out to be fascinating. Watching how they solved multiple puzzles, all the twists and turns, made it feel more comfortable to understand. It was overwhelming at times, and I of course didn't catch everything, but I also learned a lot of new things and got a good understanding of what CTF culture is like:

1) What is a CTF? Capture the flag is a competition where teams try to solve challenges and gain points by finding hidden "flags". There is also an updating live scoreboard, and statistics that show the performance of each team: number of fails, wins, taken time ... Seeing the stats was reassuring: failure isn't something to avoid as long as it leads to the endpoint; even seniors had trouble solving problems, but over countless attempts they always figured it out. The challenges were like riddles, with the answers hidden in websites, source codes, network data, even songs.

2) How is it organised? The speaker explained how they had 3 different servers, special Wi-Fi networks with passwords and bandwidth limits. They used a platform called CTFd to run everything. Antranig also mentioned an issue with a "worker class" in Python causing crashes; funny how even pros have tech-problems mid-event.

3) Variety of challenges: Antranig demonstrated a number of past CTF challenges:

- "Oh My BSD" — was a video game, and the goal was to get the character's head to a flag in the sky. The solution was to open the game files, find where the character's head size is defined (the radius value), and make it bigger. Or to use a cheat engine to reduce the gravity, allowing the player to jump higher and reach the flag. It was a simple reverse-engineering, tweaking something apart to change how it works.
- "Lilt4ever" — no one's favorite, but definitely memorable. It was a program that'd run a lilt Hawaiian song in a loop and print pixels one line. Once opened, it wouldn't allow to lower the volume or to be closed. Some thought of changing the audio file, but the pixels being linked to the original file made that ineffective.

Some noted how once they put their laptop to sleep overnight, upon opening it, they saw a sudden print of pixels that had added up over the hours. This showed that it was caused a timer in the code (written in Pascal). Turned out only way to shut it was to kill it through task manager. This was like a lesson of persistence and being attentive to how a program behaves and not just what it does.

- "Don't believe your eyes" — was a lesson to always check the source code. It showed how even if a website looks normal, the flag could be hidden in the HTML, or in the data my browser sends and receives (network traffic).
- "Bad House 21st Century" — required you to find an hidden server. Just playing a website wouldn't work. The solution was to use command-line tools (e.g. dig, traceroute) to look up DNS records and to map the path to the server. This one was just diving into the networks.
- "Ai, Ai, Ai" — required the player to convince ChatGPT that they are sysadmin to receive the key, so-called prompt injection.
- "Candy Crush" — was a simple game of moving the character to collect dropping items. The problem was, to win, you needed to collect 10, but only 9 would drop. Thus you had to exploit something to the origin, run the same, change things in the code to get the 10th item to spawn.
- "Postararakan Eraistaorgh" — would play a mashup of popular songs, but the hidden hint was in a blended gibberline (Ai-to-Ai communication system). The creator had mixed a certain sound into the music. And to find the key, one had to identify it, extract it, and use tools to translate from "Ai" back to "Human". The main skill was curiosity: to look up things we don't know and learn knowledge that leads to a solution.

4) What I learnt: Despite my limited background knowledge, I managed to take away some ideas, first of which is that CTF isn't just coding; its about curiosity, patience, persistence, and creativity in order to come to the solution. Second, everything is layered; a game is also its code, a website is its hidden source, a network has traffic we can't see. CTFs ask you to peel those layers and dig deeper. Third, tools (curl, dig, cheat engines) aren't magic, but things that can move the work smooth and simple, this highlights the importance of mastering your tools to know when and how to use them. And lastly, that infrastructure matters; CTF was an engineering project as much as a game — it required serious planning of servers, networks...

5) Challenges: The pace was relatively fast for me, mostly because the speaker assumed we already knew what CTF categories were, or what game styles or Jeopardy! Tournament. I tried my best to write down new terms and look into them after. Furthermore, I noticed a lot of solutions required experience; I never would've thought to check a "well-known" address for a calendar... So I understood the "what", but occasionally missed the "how" or "why".

6) Conclusion: Despite the fear, I'm glad to have attended and gained more knowledge. CTFs seem to cultivate a clever and flexible intelligence, and it showed how cybersecurity isn't just firewalls and antivirus, but also a mindset of how to question how everything works and what hidden corners may exist.

In terms of my systems engineering course, it showed the importance of system design and logical problem-solving. As a first exposure, it was simulating, challenging, and fun. The best thing is that it made me want to learn more about it.