Mistakes We Make: SDLC Implementation

Artsem Kadushko

Application Security Engineer

Moscow, August 25-26, 2022

Who am I?

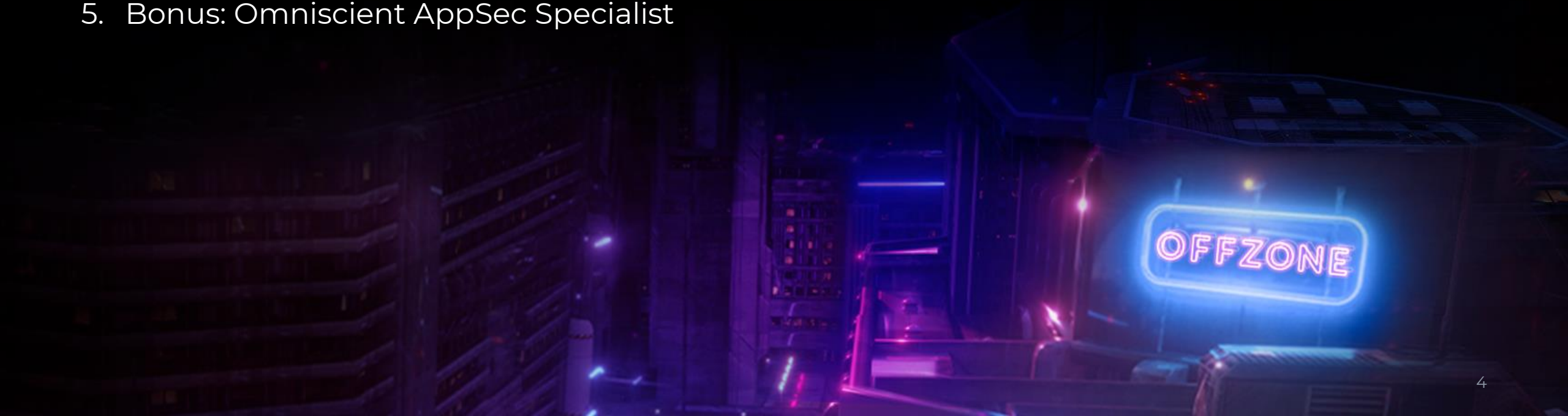
- Application Security Engineer
- Head of Bulba Hackers
- CTF Player
- Penetration Tester
- Cyber Security Lecturer
- Belarussian infosec influencer

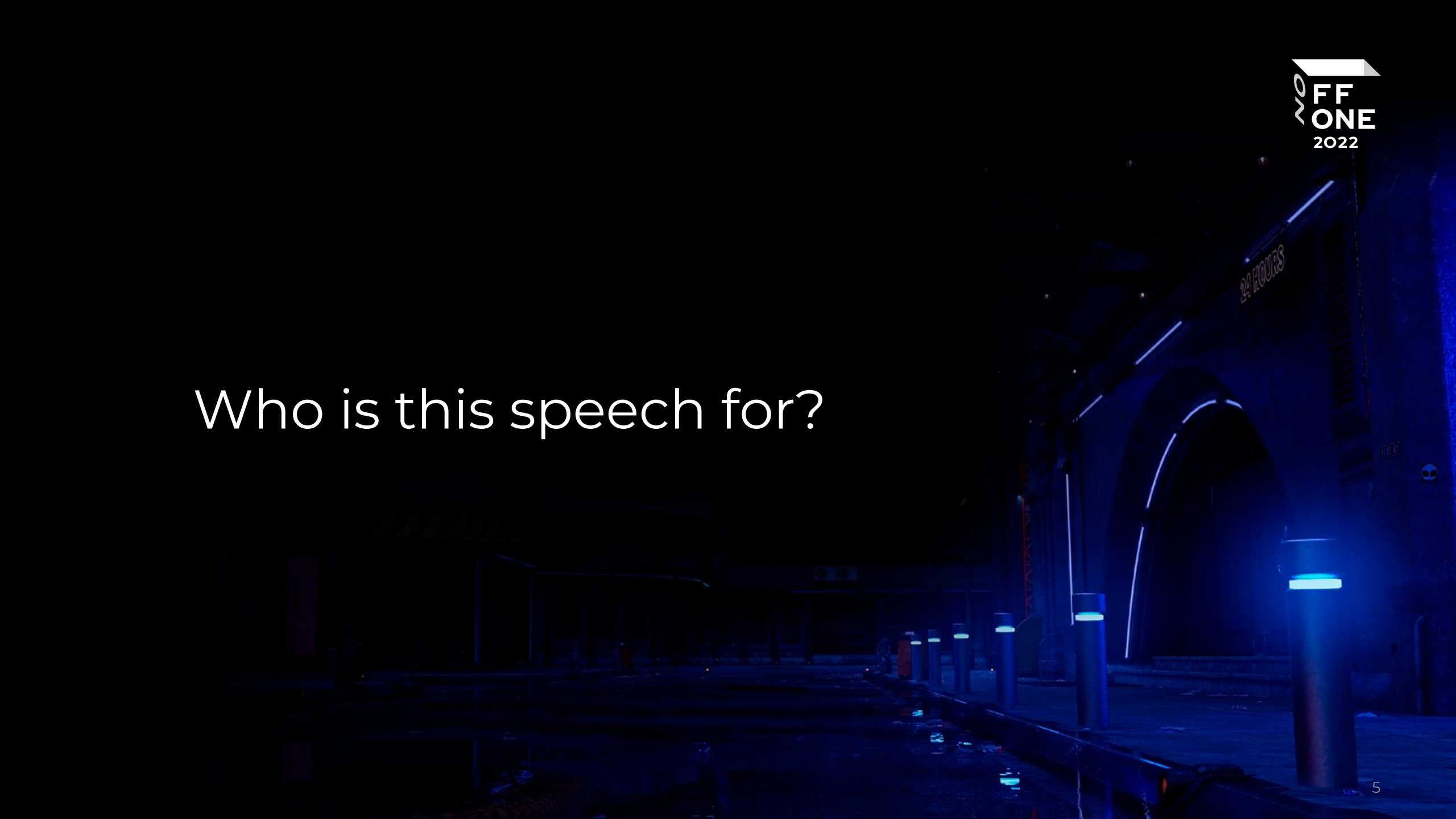
Who are Bulba Hackers?

- Information and Cyber Security community in Belarus
- Top-1 CTF Team in Belarus (Regarding CTFTime ratings in 2019-2022)
- Top-5 The Standoff team in 2021 (Spring & Autumn)
- Improving education in the cybersecurity field in Belarus
- Authors of Computer Security courses (CTF & Pentest) for students

Agenda

1. Who is this speech for?
2. Lack of planning
3. Lack of prioritization and understanding of assets
4. Money = Quality
5. Bonus: Omniscient AppSec Specialist

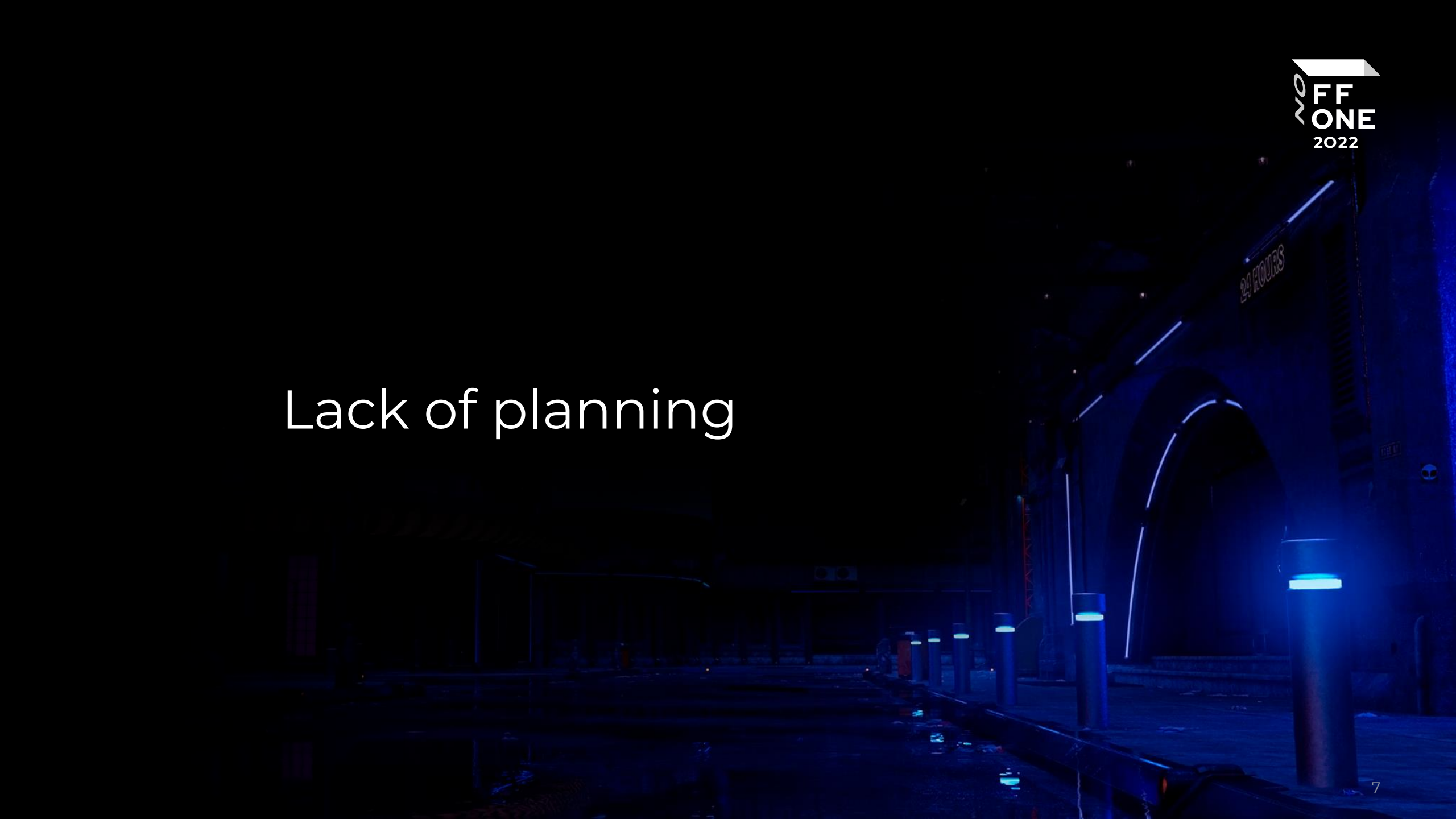




Who is this speech for?

A LONG
TIME AGO
IN A GALAXY
FAR, FAR AWAY...

Lack of planning



Lack of planning

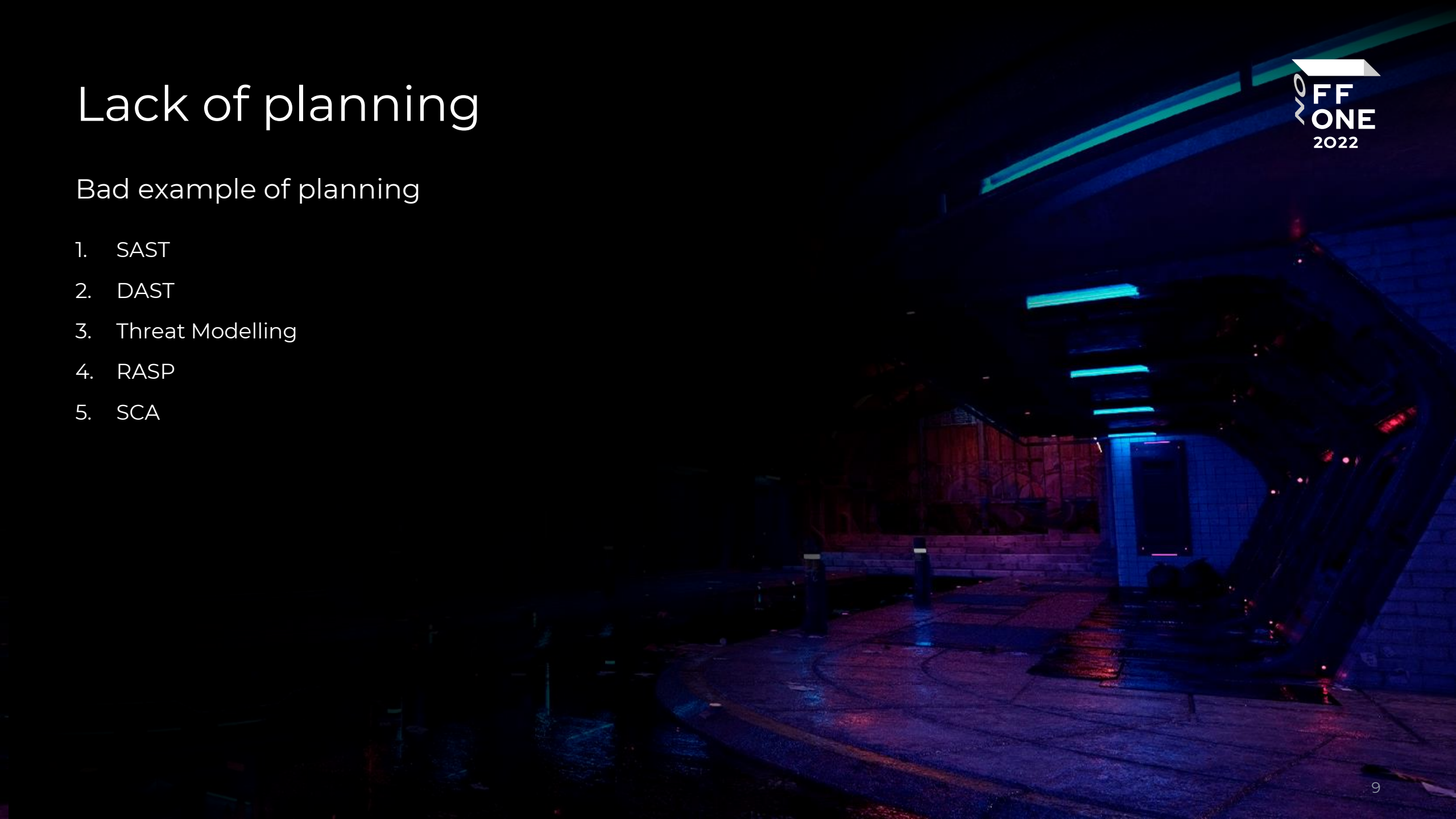
Why do we need to plan our SDLC implementation?

- In SDLC we have too many possibilities to make our development safe
- Understanding requirements for implementation (i.e. time, money, people)
- Solving problems earlier
- We don’t waste time
- We understand our goals for nearest future

Lack of planning

Bad example of planning

1. SAST
2. DAST
3. Threat Modelling
4. RASP
5. SCA

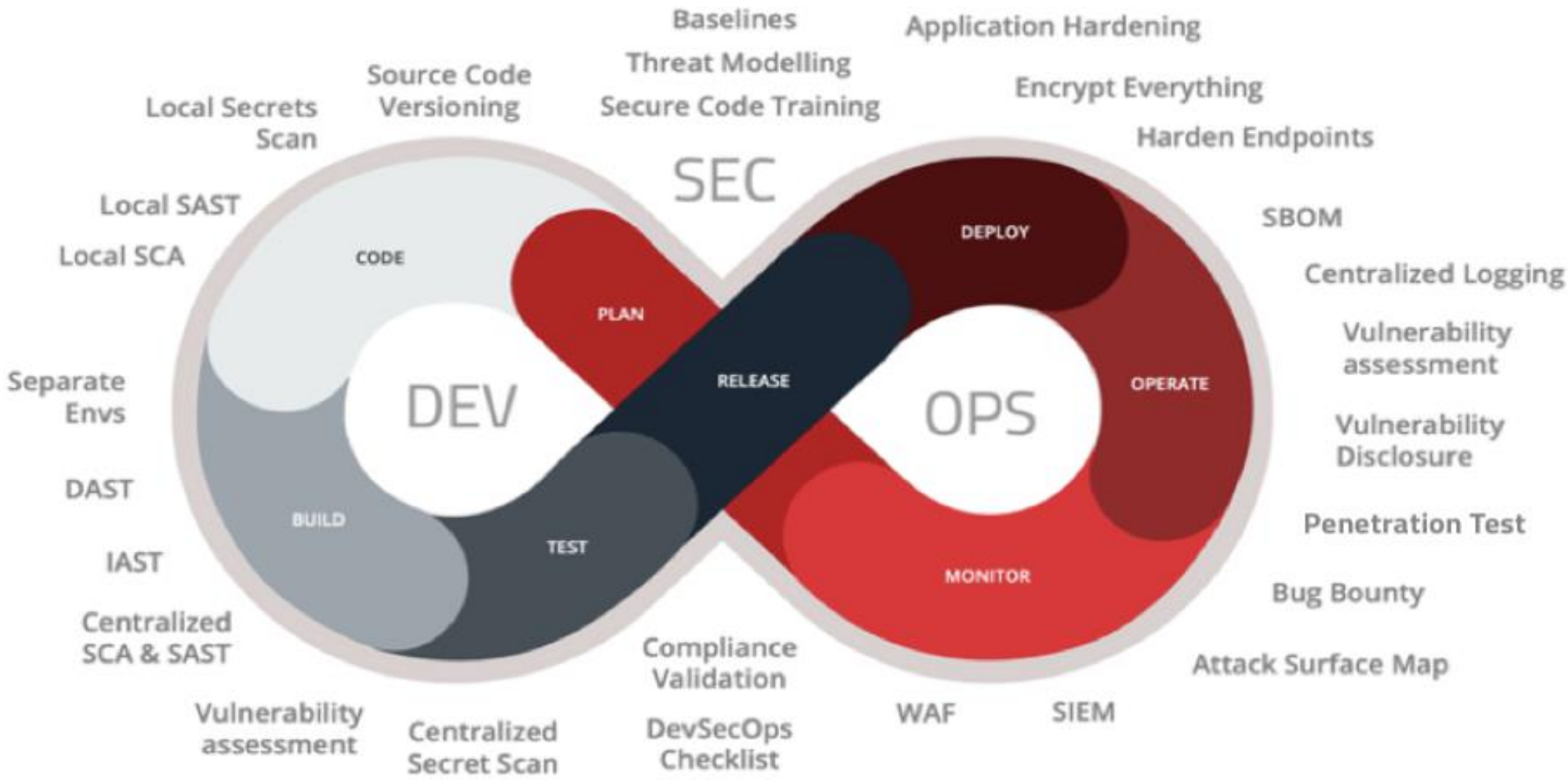


Lack of planning

Good example of planning

1. SAST
    1. Figure out target projects for SAST
    2. Finding the best SAST solution for our cases
    3. Integration in test's CI/CD process without blocking
    4. Fixing False-Positives / Remove or Add new rules for SAST
    5. Defining Quality Gates for process
    6. Integrating into CI/CD without enabled Quality Gates
    7. Collecting Feedback
    8. Reconfiguring our SAST
    9. Enabling Quality Gates for CI/CD
    10. ???
    11. Supporting SAST

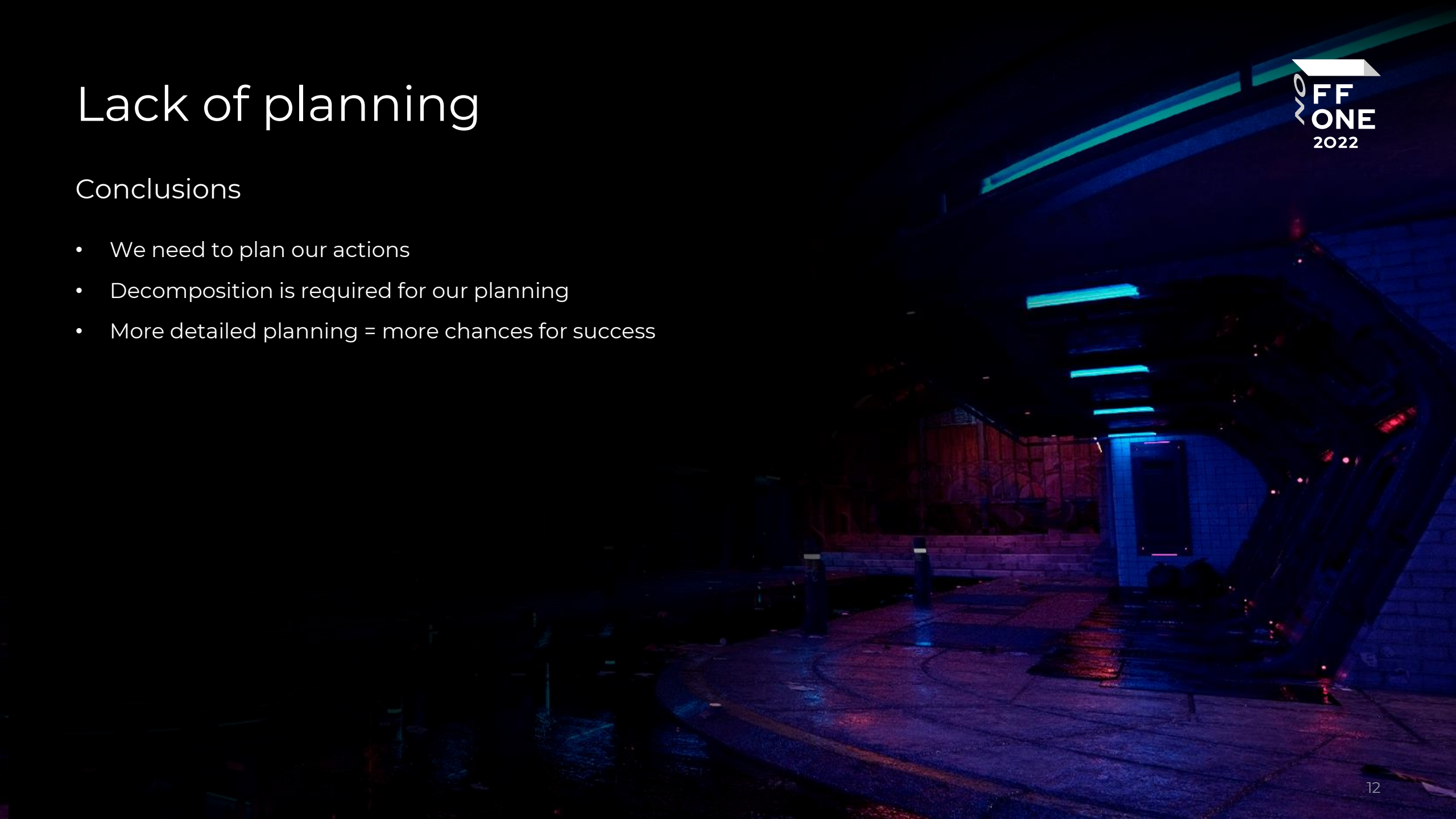
DevSecOps Playbook



Lack of planning

Conclusions

- We need to plan our actions
- Decomposition is required for our planning
- More detailed planning = more chances for success



Lack of prioritization and understanding of assets

Lack of prioritization and understanding of assets

What is prioritization?

Sorting our plan by smth (compliances, needs, capabilities, etc.)

Lack of prioritization and understanding of assets

Why do we need to know our assets?

“If you know the enemy and know yourself, you need not fear the result of a hundred battles.

If you know yourself but not the enemy, for every victory gained you will also suffer a defeat.

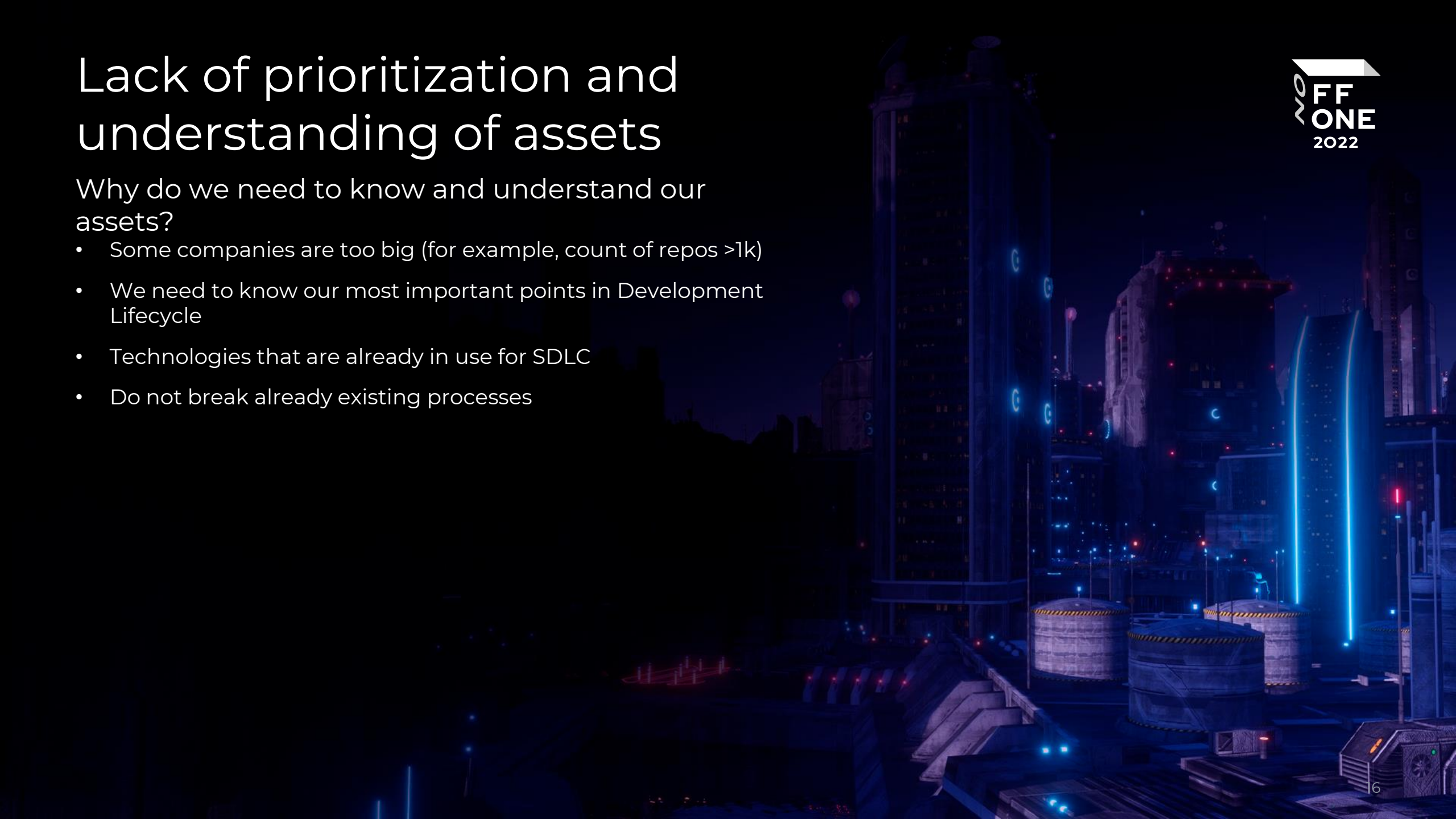
If you know neither the enemy nor yourself, you will succumb in every battle.”

© Sun Tzu, the Art of War

Lack of prioritization and understanding of assets

Why do we need to know and understand our assets?

- Some companies are too big (for example, count of repos >1k)
- We need to know our most important points in Development Lifecycle
- Technologies that are already in use for SDLC
- Do not break already existing processes



Lack of prioritization and understanding of assets

How can we find our assets?

- Developers
- DevOps
- System Admins
- Project Owners / Project Management

Lack of prioritization and understanding of assets

Different cases to show different approaches to implementing SDLC

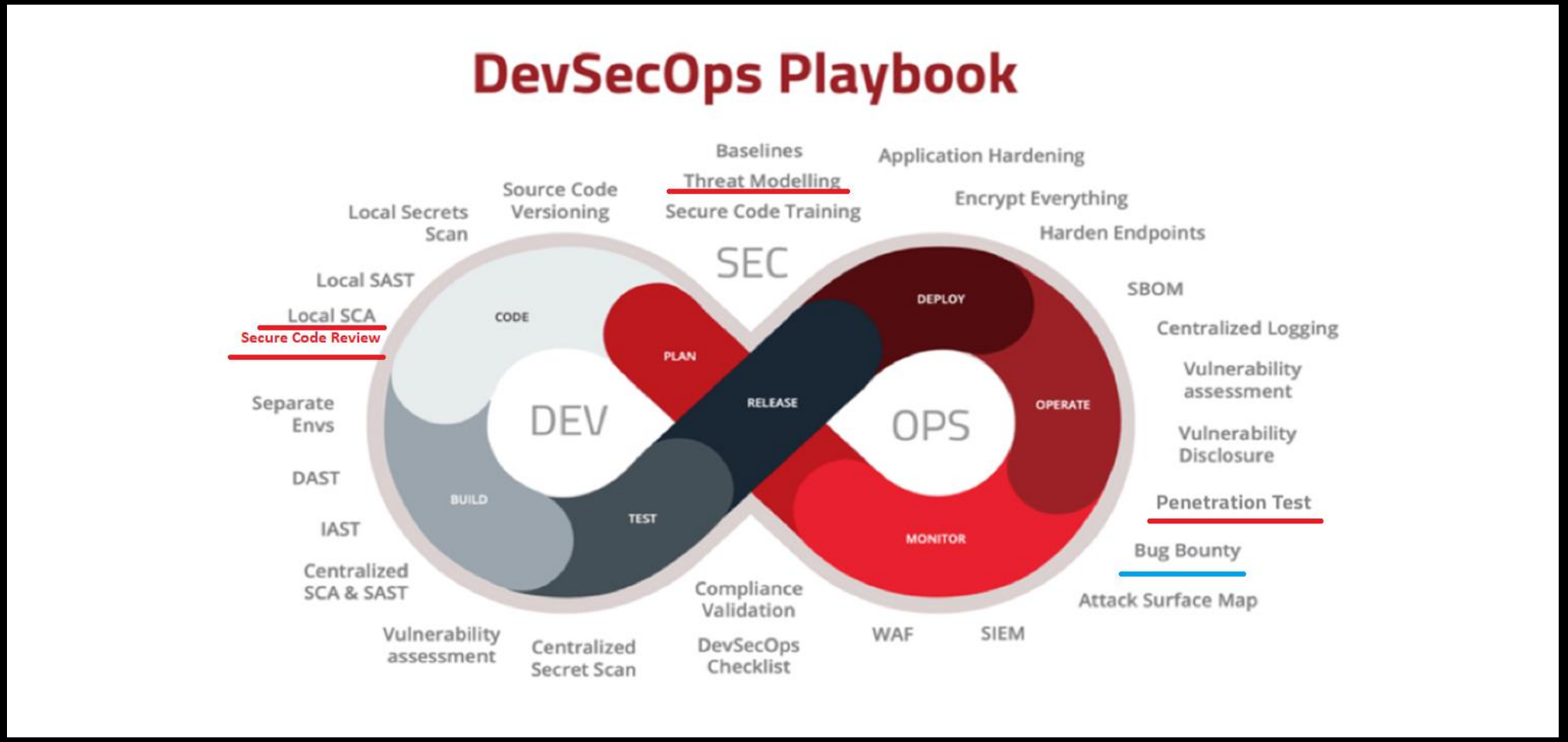
- Your company has 1 product
- Your company creating a big amount of new projects weekly (and smth like this)
- Your developers use different CI/CD systems (Jenkins, TeamCity, Gitlab, Github Actions, etc.)
- Your developers do not have any CI/CD processes

Lack of prioritization and understanding of assets

Example priorities for cases

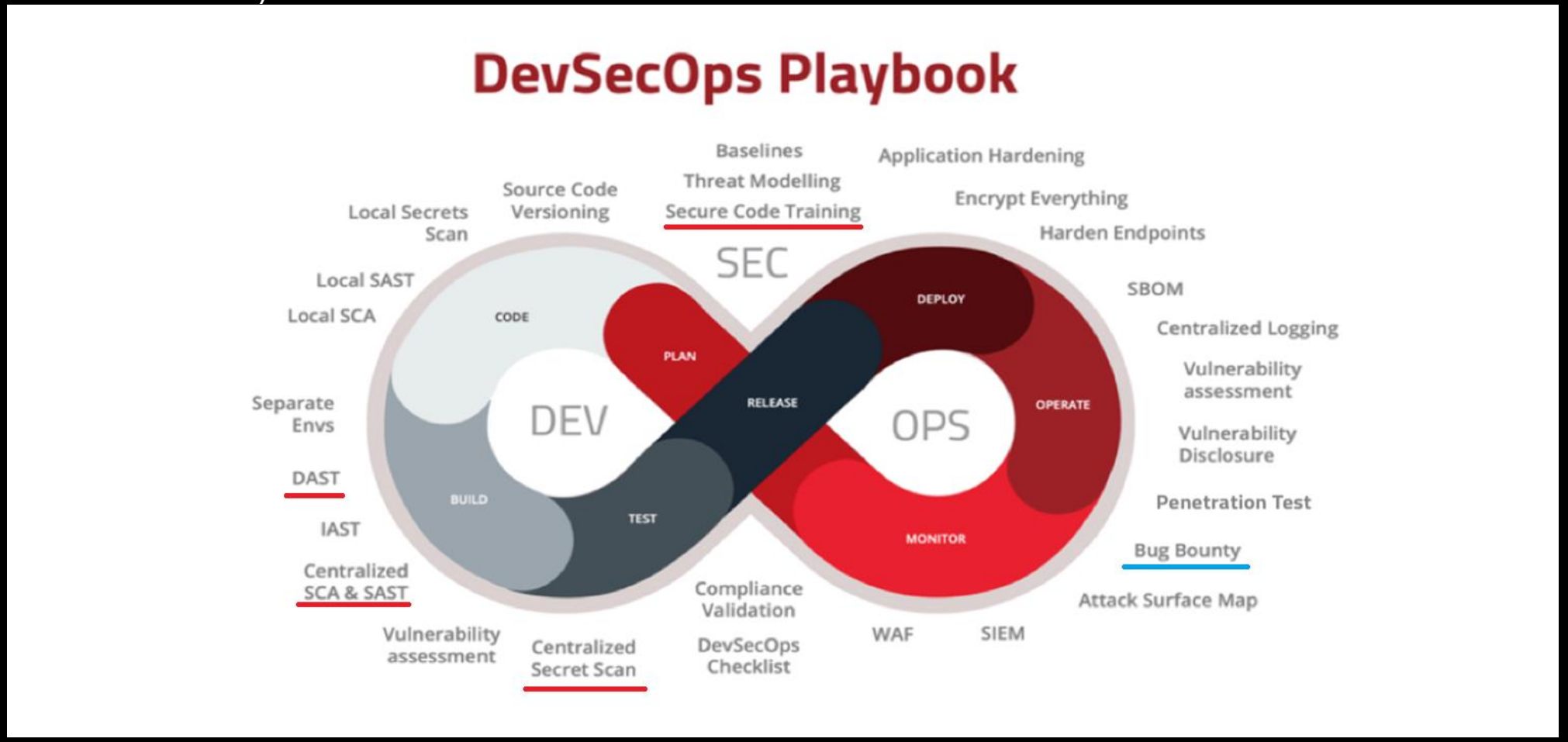
Lack of prioritization and understanding of assets

Your company has 1 product



Lack of prioritization and understanding of assets

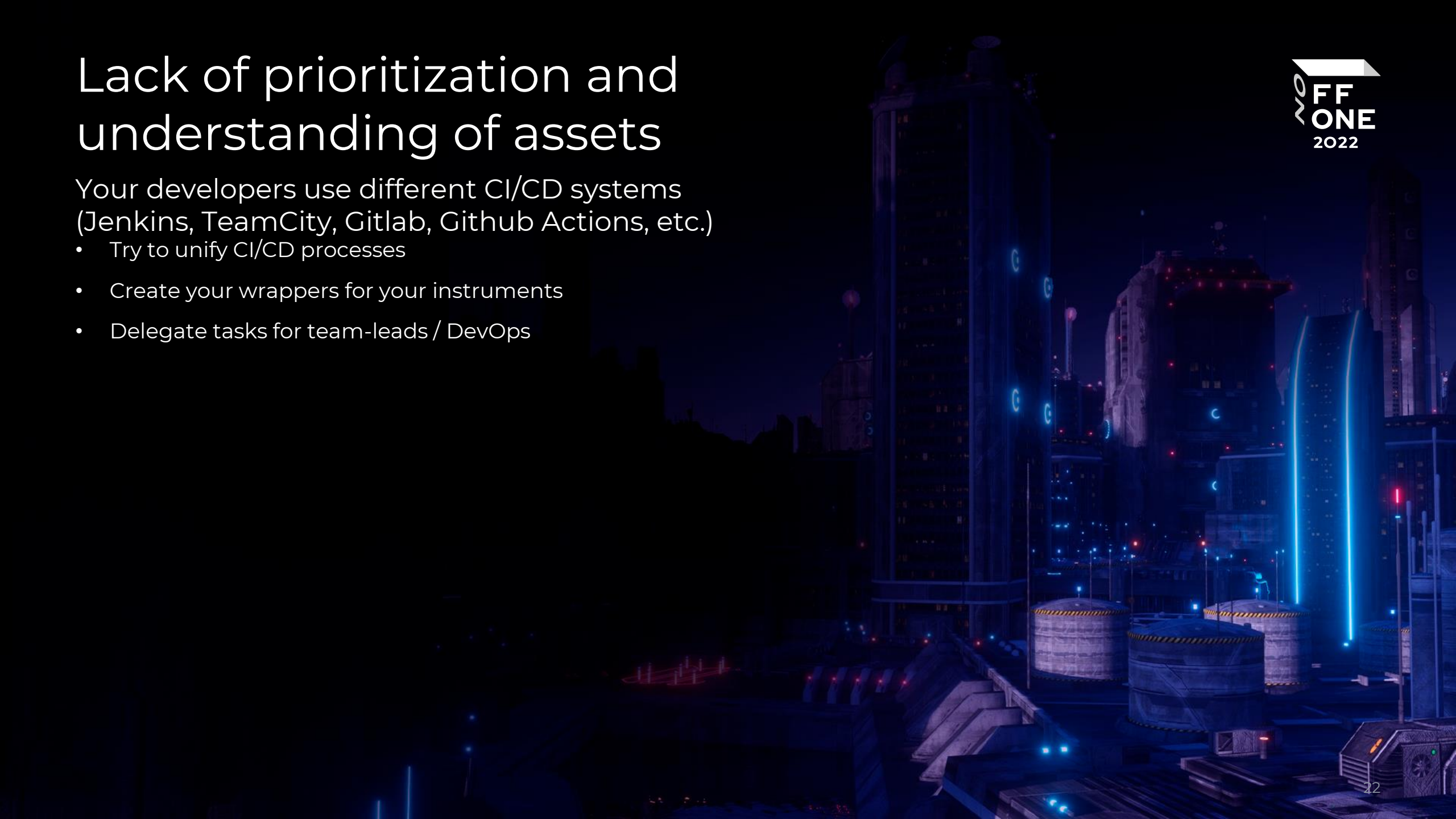
Your company creating a big amount of new projects weekly (and smth like this)



Lack of prioritization and understanding of assets

Your developers use different CI/CD systems (Jenkins, TeamCity, Gitlab, Github Actions, etc.)

- Try to unify CI/CD processes
- Create your wrappers for your instruments
- Delegate tasks for team-leads / DevOps



Lack of prioritization and understanding of assets

Your developers do not have any CI/CD processes

Lack of prioritization and understanding of assets

Conclusions

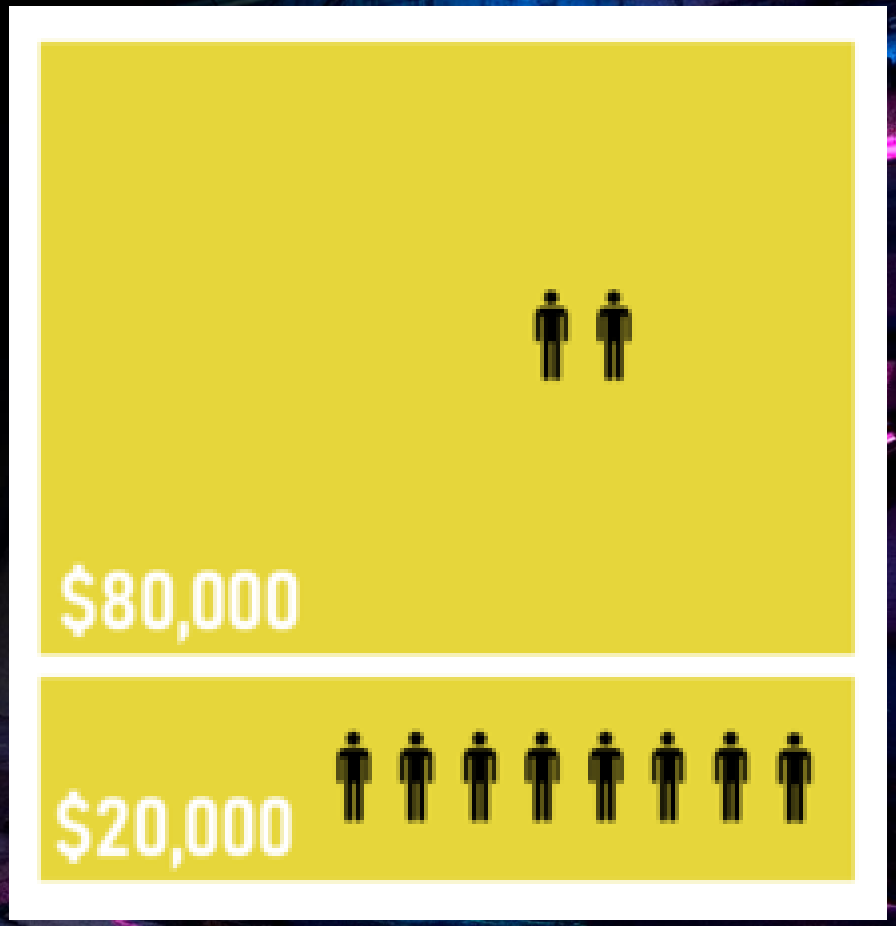
- Planning without prioritization = failure
- Prioritization without understanding assets = failure
- Before implementing SDLC implement the development lifecycle

Money = Quality

Money = Quality

Pareto Principle

20% effort = 80% success



Money = Quality

Why should we use Pareto's principle in terms of SDLC implementation?

- Most Enterprise tools are based on open-source analogs
- We spend a lot of time testing a product
- It's more important to use smth than nothing
- Implementation budget may be small or non-existent

Money = Quality

SAST Vendors

Checkmarx

Solared APPscreener

PT Application Inspector

MICRO FOCUS Fortify

coverity

VERACODE

PVS-Studio

sonarqube

Money = Quality

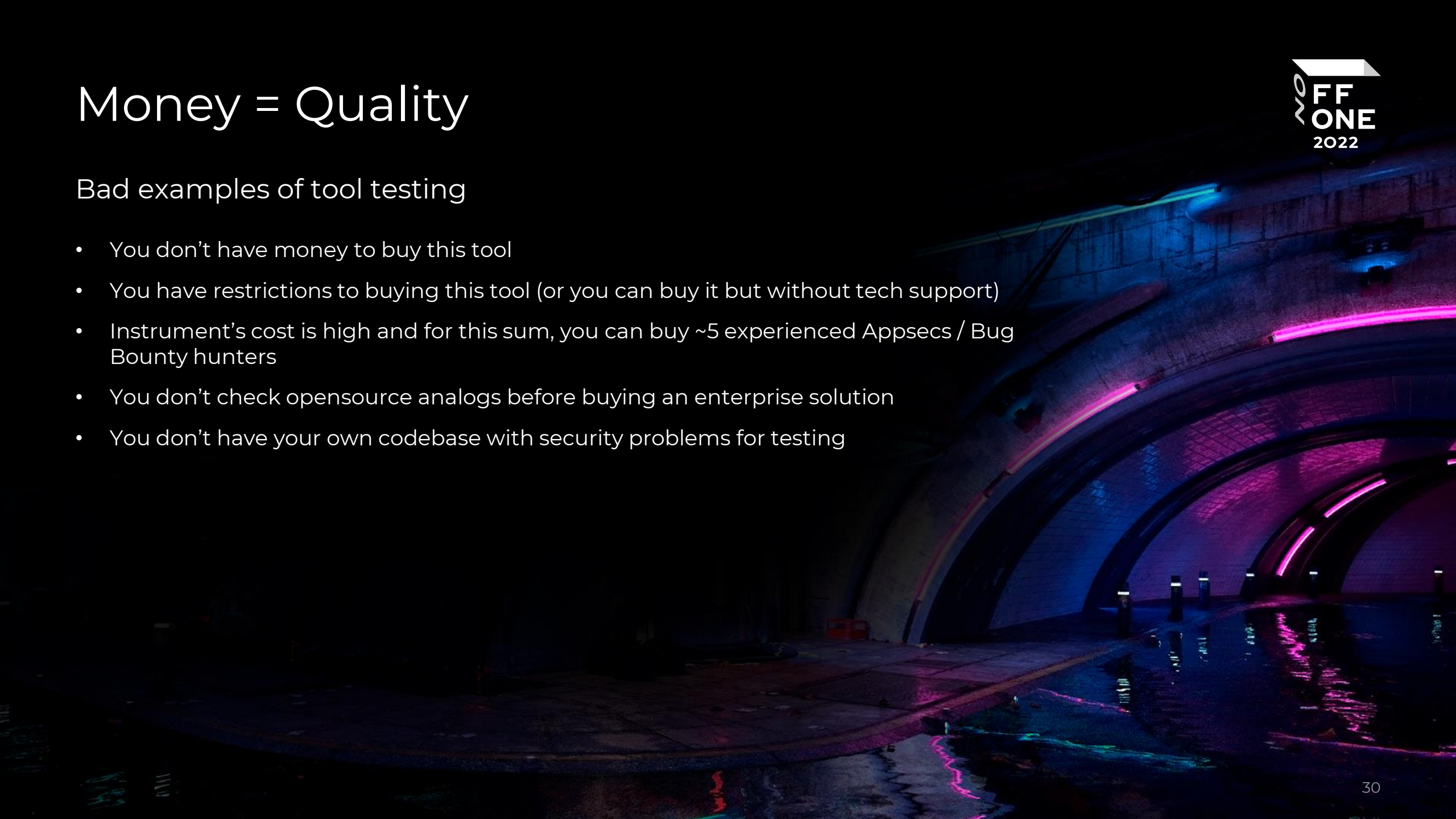
Opensource SAST

- MobSF
- SpotBugs
- CodeQL
- NodeJSScan
- SemGrep
- GoSec
- Bandit
- Security Code Scan

Money = Quality

Bad examples of tool testing

- You don’t have money to buy this tool
- You have restrictions to buying this tool (or you can buy it but without tech support)
- Instrument’s cost is high and for this sum, you can buy ~5 experienced Appsecs / Bug Bounty hunters
- You don’t check opensource analogs before buying an enterprise solution
- You don’t have your own codebase with security problems for testing



Money = Quality

Conclusions

- Pareto principal is fundamental for SDLC
- Opensource can save your money
- Wasting time on testing instruments that you can’t buy = failure
- Experienced specialist is better (in most cases) than a tool

Bonus: Omniscient AppSec Specialist

Omniscient AppSec Specialist

Mistakes Companies Make

- 1 AppSec can solve all problems
- SAST = complete SDLC
- AppSec = DevSecOps = DevOps = Developer = Pentester = …
- Paper security > Practical security
- Bureaucracy
- No budget

Omniscient AppSec Specialist

Types of application security

- Static Application Security (SAST & Code)
- Dynamic Application Security (DAST & Pentest)
- DevSecOps/DevOps (Implementing new tools in pipelines)
- Developers (Creating new tools for AppSecs)
- Architects (Threat Modelling)

Mistakes We Make: SDLC Implementation

Summary

- Planning is important
- Prioritization Is also important
- Pareto’s principle is very useful
- In most cases, you can find opensource analogs
- In SDLC we have many ways to make our development safe


My boss congrats me with SDLC implementation
Me, using Sonarqube Community as SAST
meme-arsenal.ru


ON
FF
ONE
2022