AppSec Low-Coster: how to build security program within tough constraints?

Michael Markevich
Cybersecurity expert and mentor

AppSecFest 2026

Who Am I

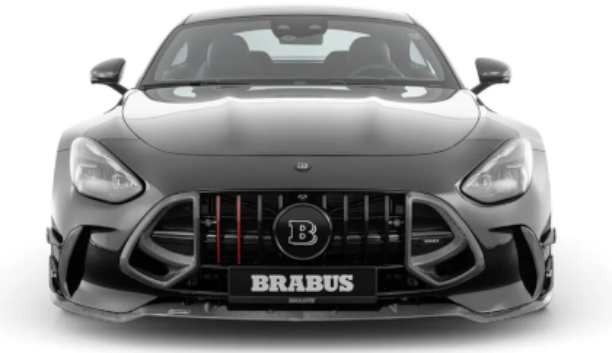
- Cybersecurity practitioner with over 25 years of experience
- 3 x CISO, startup advisor and founder
- Career mentor and academic lecturer
- Open-source software engineer and privacy enthusiast

Why This Talk?

Security is often seen as an unnecessary expense. Is it possible to build a usable AppSec program within budget and resource constraints?

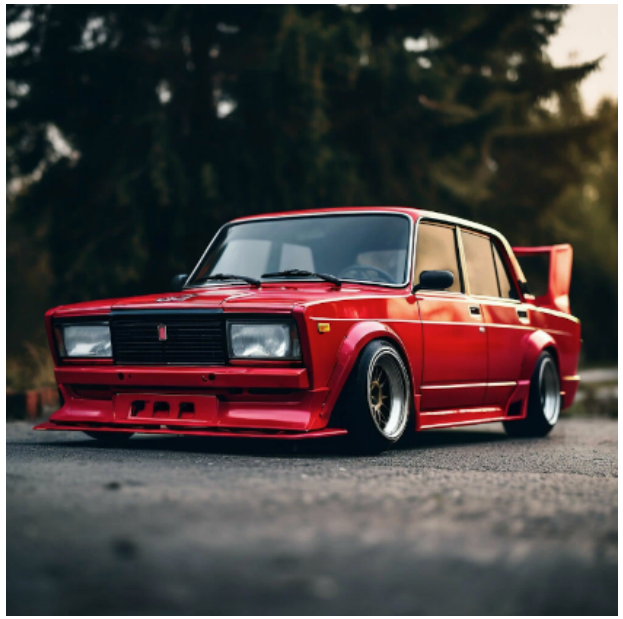
Some people claim that...

Expensive



Good

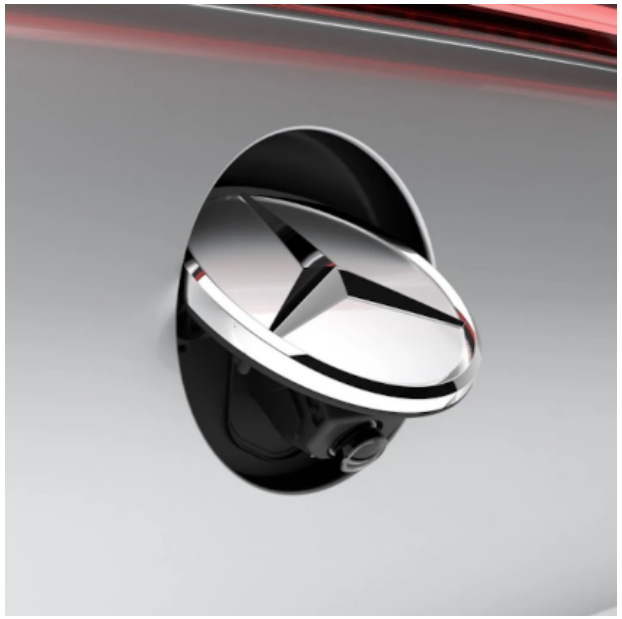
Cheap



Bad

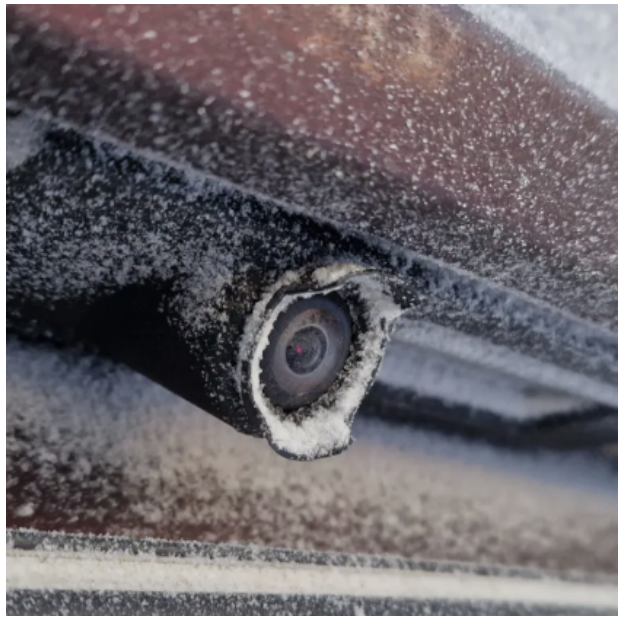
Can it be different?

Expensive



Good

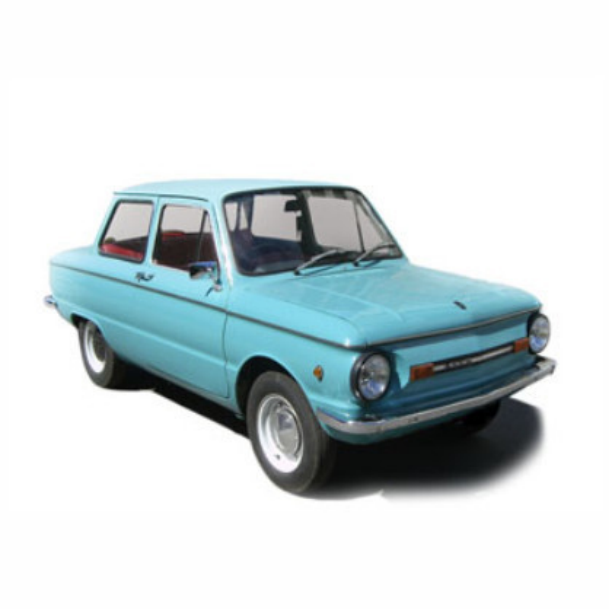
Expensive



Bad

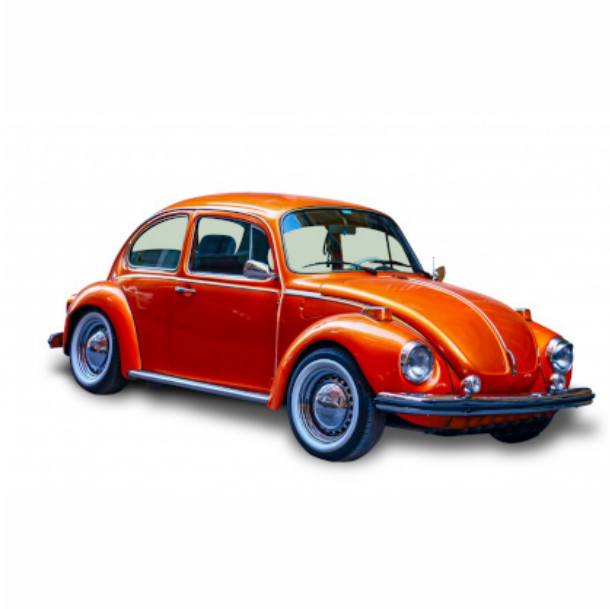
Any other variants?

Cheap



Bad

Cheap



Good

It looks like we have a chance

Let's try to build an AppSec low-coster!

What is a usable AppSec process?

1. Risk-based: we invest time and resources only in what is really needed
2. Frictionless: doesn't cause troubles and delays for other participants
3. Adaptive: capable to respond to its own mistakes

Anything else?

Let's talk about risks

Many of us don't like dealing with risk assessments.

Let me show you something: https://magoo.github.io/Risk-Forecasting

AppSec problems right now

- Roughly 60-80% of code is generated by agents
- Code is written much faster than reviewed
- Developers tend to trust AI more than apply critical thinking
- Shortage of skilled AppSec engineers

So do we really need AppSec engineers?

Frictionless AppSec

AppSec engineers don't block the process if they don't exist.

If we set security requirements from the very start, we have higher chances to generate secure code.

Who can set the requirements?

Can a regular development team lead alone set adequate security requirements?

Is this a real "shift left" we've been talking about for so many years?

AppSec agentic roles

- Threat modelling
- Writing security requirements
- SAST and DAST
- Compliance

What's left for us, humans? Let's discuss.

Adaptive AppSec

We want to document the context and improve it after failures.

- LLM Wiki (https://github.com/nashsu/llm_wiki) and alternatives
- Octopus Code Review (https://github.com/octopusreview)

Where could we save costs?

- Reduce number of traditional (legacy) AppSec ceremonies
- Reduce or avoid dedicated appsec headcount

Where could we get an unpredicted cost?

- Code maintenance costs after one or two years in production
- Token budget (growing all the time)

Conclusions

Agentic AppSec looks as a realistic approximation to an ideal world where insecure code doesn't exist. By maintaining token cost and knowledge quality we can manage AppSec budget efficiently.

Thank you!

Q&A Chat: