

accu
conFerence
2025

Rust Cargo Cult?

Victor Ciura





Rust: Cargo Cult?

ACCU

April 2025

 @ciura_victor

 @ciura_victor@hachyderm.io

 @ciuravictor.bsky.social

Victor Ciura
~~Principal Engineer~~
Rambling Idiot
Rust Tooling @ Microsoft

About me



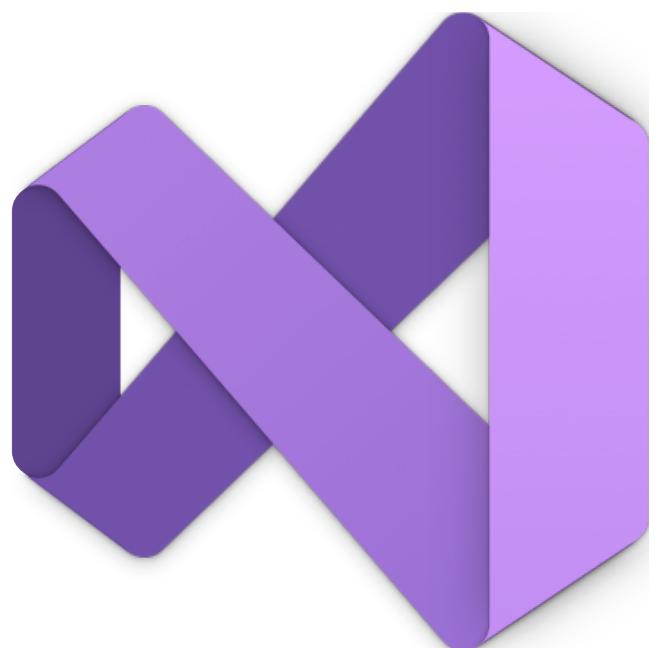
Advanced Installer



Clang Power Tools



Oxidizer SDK



Visual C++



Rust Tooling
Microsoft

 [@ciura_victor](https://twitter.com/ciura_victor)
 @ciura_victor@hachyderm.io
 [@ciuravictor.bsky.social](https://ciuravictor.bsky.social)

How many Rust... curious folks 🤔



How many Rust... curious folks



enthusiasts



How many Rust... curious folks 🤔



enthusiasts 📚🍿

hackers 💻

How many Rust... curious folks 🤔



enthusiasts  

hackers 

professionals 



This is not a Rust 101

* I'm not even going to talk about [language](#) features (much)

I'm not here to:

- convert anyone to 🦀 Rust
- start any language wars
- "*sell the Rust snake oil*"
- tell you to RiiR

So, don't throw 🍅



3 things I like about Rust 😊

3 things I wish* would be better 🙏

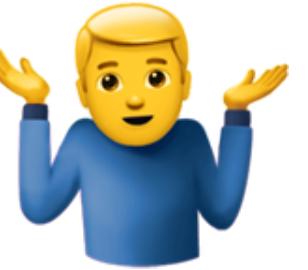
* also [working](#) on improving these (merely *wishing* doesn't accomplish much)

What's so great about Rust anyway?



- Safety by default (spatial, temporal, thread, async)
- Extreme range of operation
- Community & ecosystem

If Rust is so great, why isn't it widely adopted?



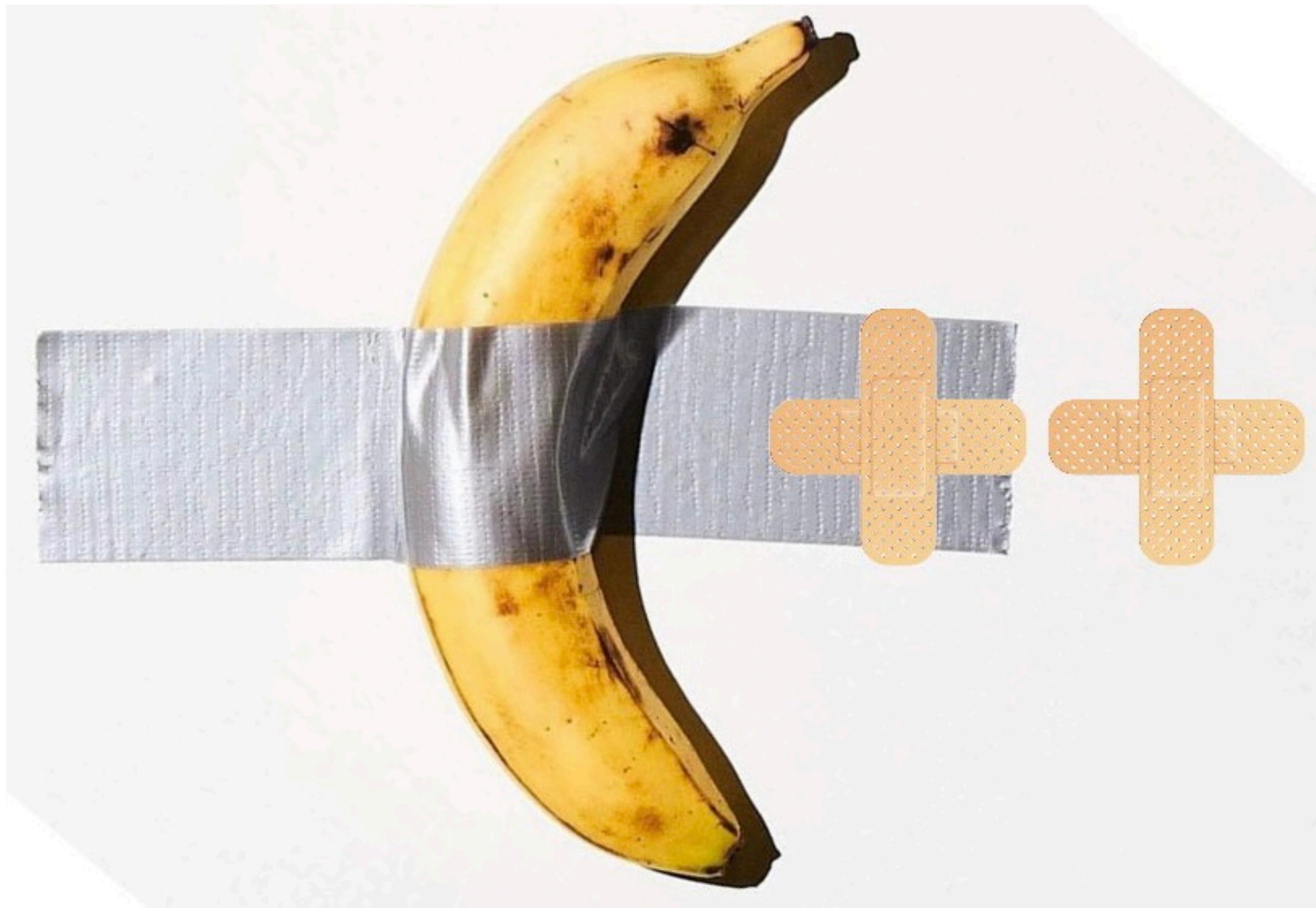
- Enterprise-grade tooling
- Ergonomic interop with C++, C#, Python, Kotlin, etc.
- Maturity of the ecosystem
 - Certifications, ISO standards, audit/assessor companies

Safety by default **(spatial, temporal, thread, async)**

Choices... 😅



C++ Safety Profiles





National Security Agency | Cybersecurity Information Sheet

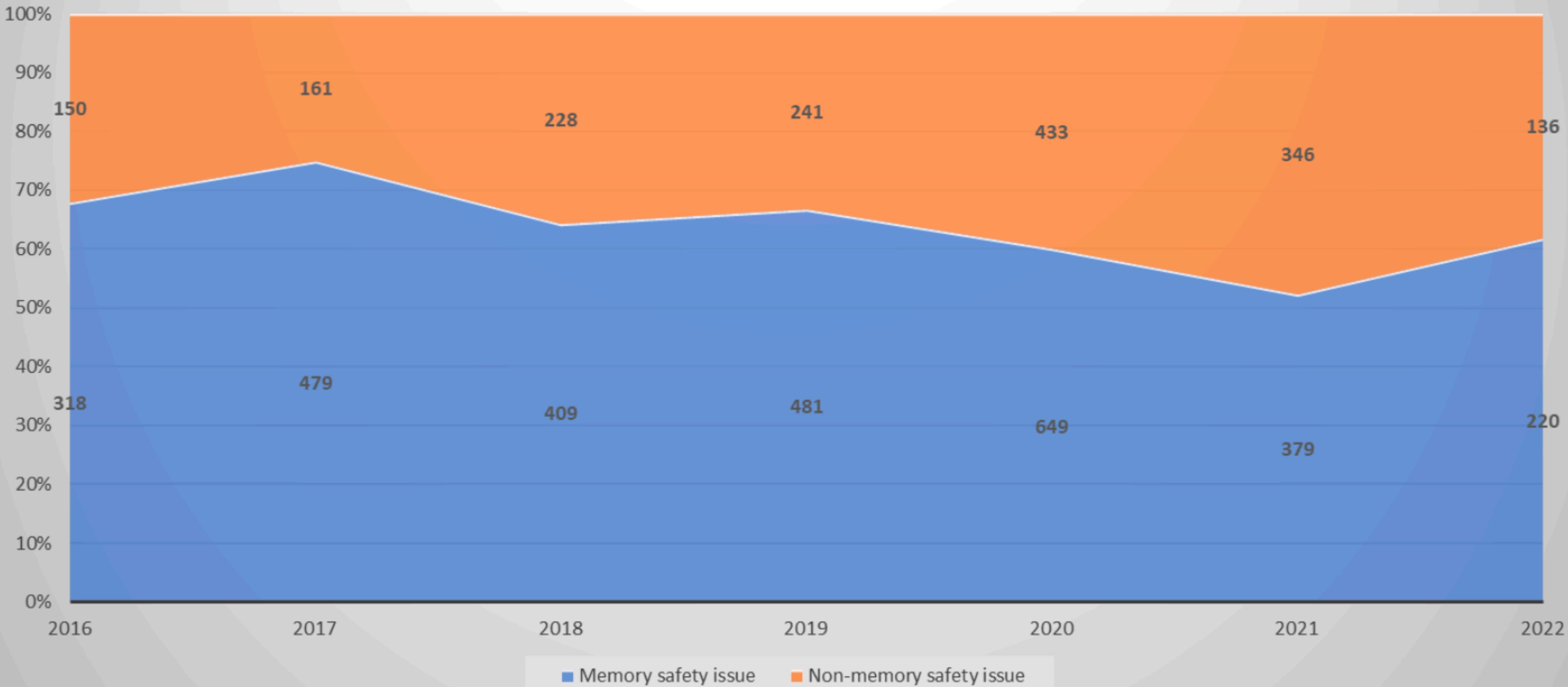
Software Memory Safety

media.defense.gov/2022/Nov/10/2003112742/-1/-1/0/CSI SOFTWARE MEMORY SAFETY.PDF

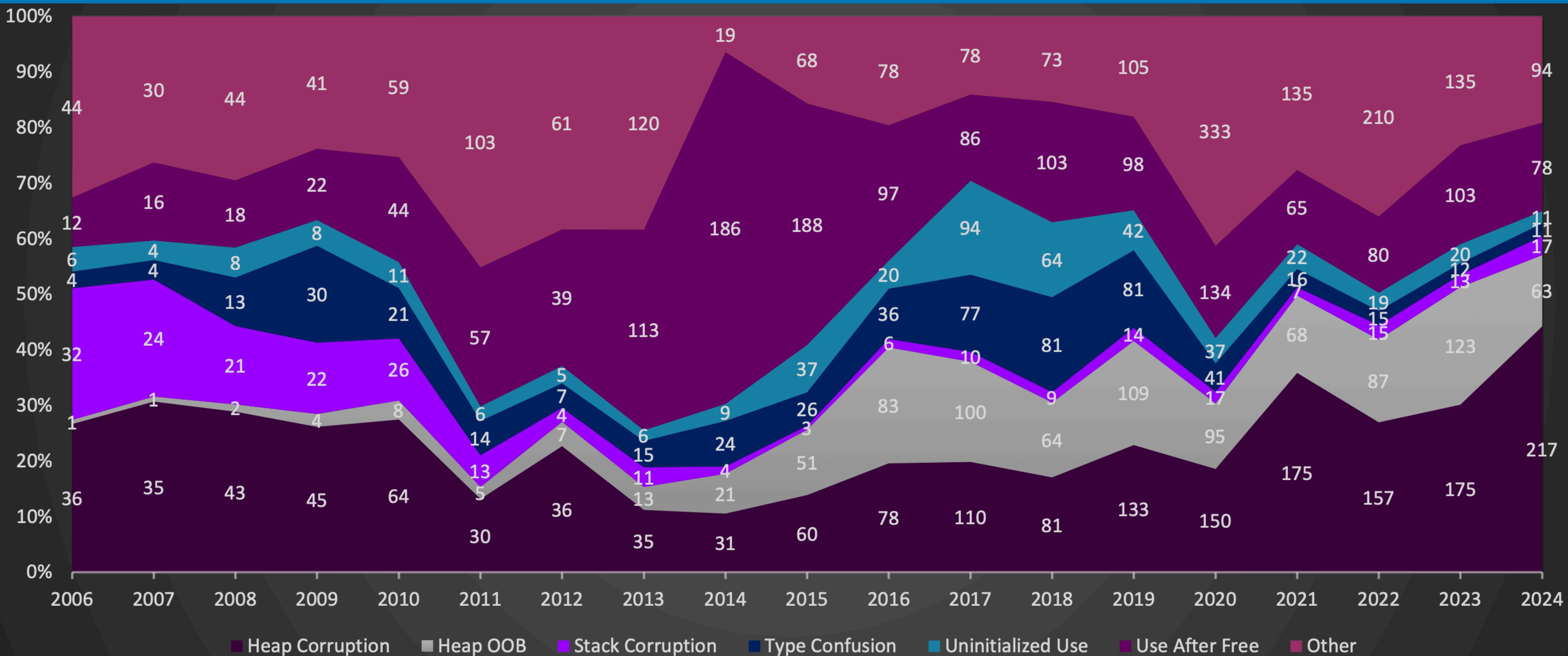


Microsoft CVEs

Is CVE a Memory Safety Issue (RCE, EOP, Info Disclosure)?



Root cause of memory safety CVEs



Embracing an Adversarial Mindset for C++ Security - Amanda Rousseau

youtube.com/watch?v=glkMbNLogZE

Systems Language Overview

	Rust	C++	C
Object Lifetime	Statically Enforced	Not Enforced, unclear path forward.	No hope
Type Safety	Statically Enforced	Not enforced, unclear path forward.	No hope
Bounds Safety	Enforced at runtime when needed	Could be enforced for STL containers.	No hope
Uninitialized Safety	Statically Enforced	Not enforced, could be enforced w/ breaking change.	Stack could be enforced w/ breaking change.

Rust ❤️ C++

They need to play nice together... for a looong time!

Microsoft: Ongoing Efforts

Microsoft: Ongoing Efforts

- Making changes in our **SDL** operations (evolving needs of emerging technologies)

Microsoft: Ongoing Efforts

- Making changes in our **SDL** operations (evolving needs of emerging technologies)
- Completing our deployment of **CodeQL**, integrated with GitHub Copilot learnings

Microsoft: Ongoing Efforts

- Making changes in our **SDL** operations (evolving needs of emerging technologies)
- Completing our deployment of **CodeQL**, integrated with GitHub Copilot learnings
- Continue to invest in **hardening C & C++ code**

Microsoft: Ongoing Efforts

- Making changes in our **SDL** operations (evolving needs of emerging technologies)
- Completing our deployment of **CodeQL**, integrated with GitHub Copilot learnings
- Continue to invest in **hardening C & C++ code**
- Standardizing on **Rust** and other **memory safe languages (MSLs)**

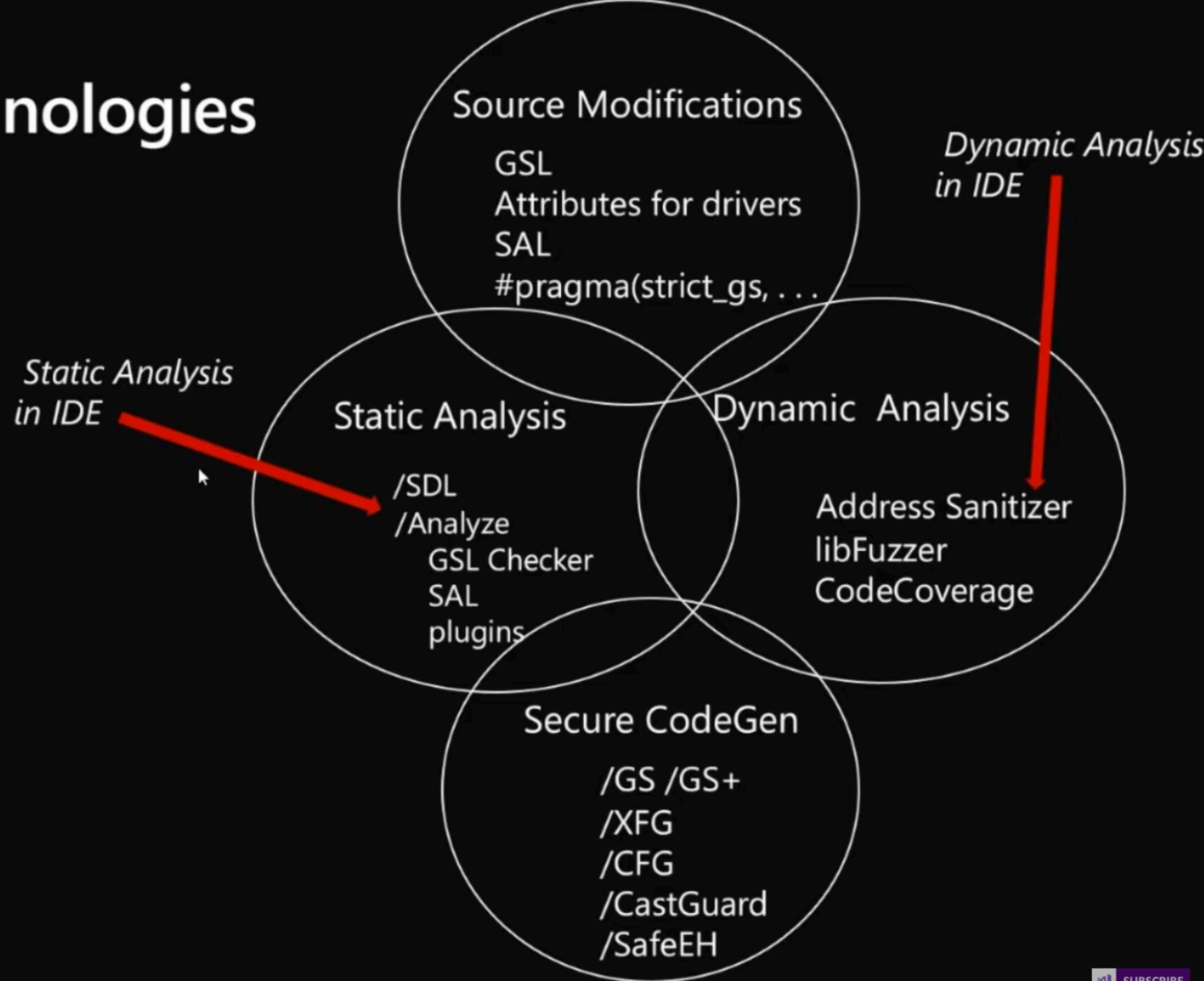
Microsoft: Ongoing Efforts

- Making changes in our **SDL** operations (evolving needs of emerging technologies)
- Completing our deployment of **CodeQL**, integrated with GitHub Copilot learnings
- Continue to invest in **hardening C & C++ code**
- Standardizing on **Rust** and other **memory safe languages (MSLs)**
- Contribute  to support the work of the **Rust Foundation**

Microsoft: Ongoing Efforts

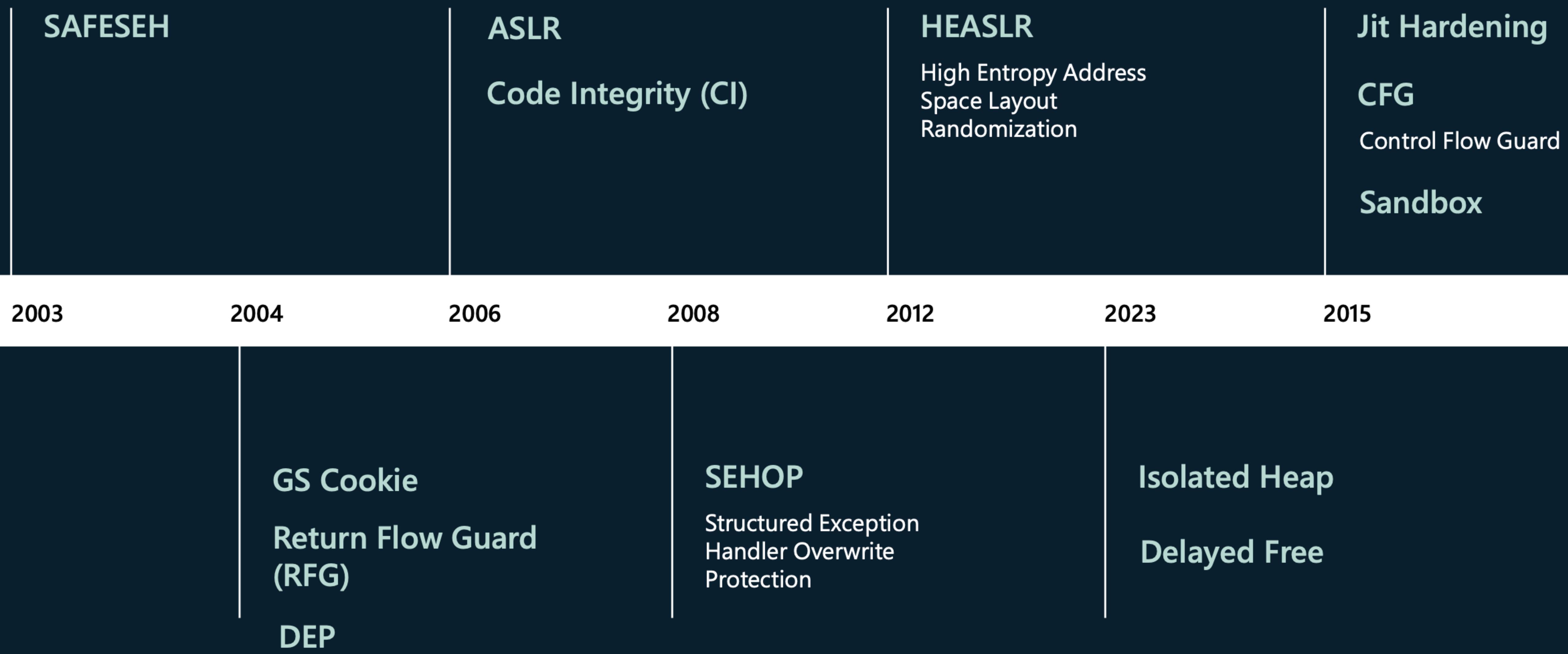
- Making changes in our **SDL** operations (evolving needs of emerging technologies)
- Completing our deployment of **CodeQL**, integrated with GitHub Copilot learnings
- Continue to invest in **hardening C & C++ code**
- Standardizing on **Rust** and other **memory safe languages (MSLs)**
- Contribute  to support the work of the **Rust Foundation**
- Assist developers making the *transition* from C, C++, C# to Rust
 - Investing in Rust **developer tooling**

C++ Security Technologies

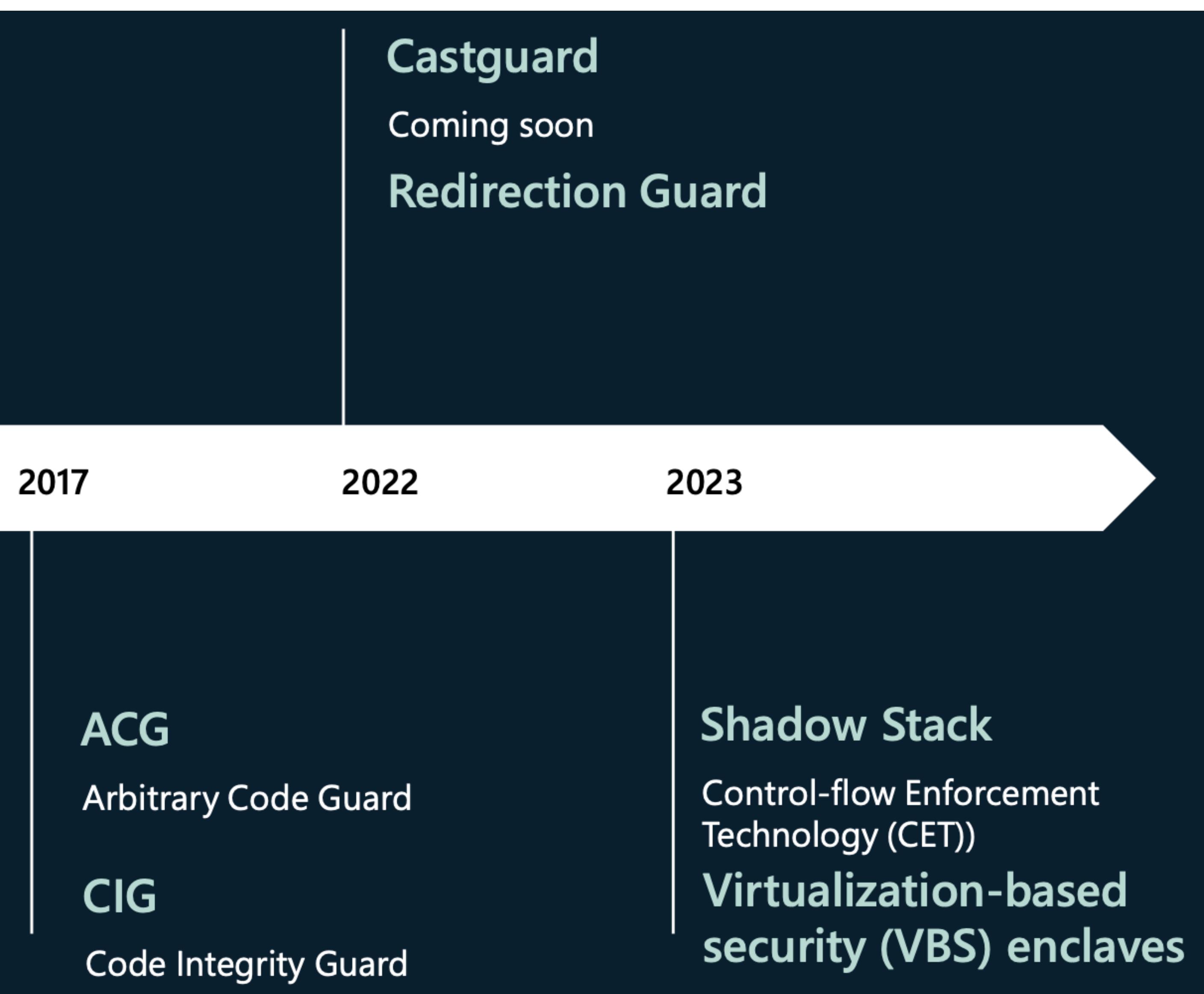


youtube.com/watch?v=i8_RfDAEjMs

Exploit Mitigation Timeline



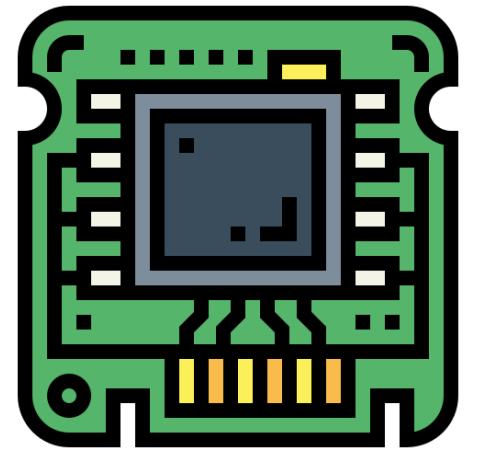
Exploit Mitigation Timeline



Exploit Mitigation Timeline



Extreme range of operation



But Why?



Rusty Windows

Rust already in the Windows kernel (since 2023)

```
C:\Windows\System32>dir win32k*
Volume in drive C has no label.
Volume Serial Number is E60B-9A9E
```

_rs = Rust!

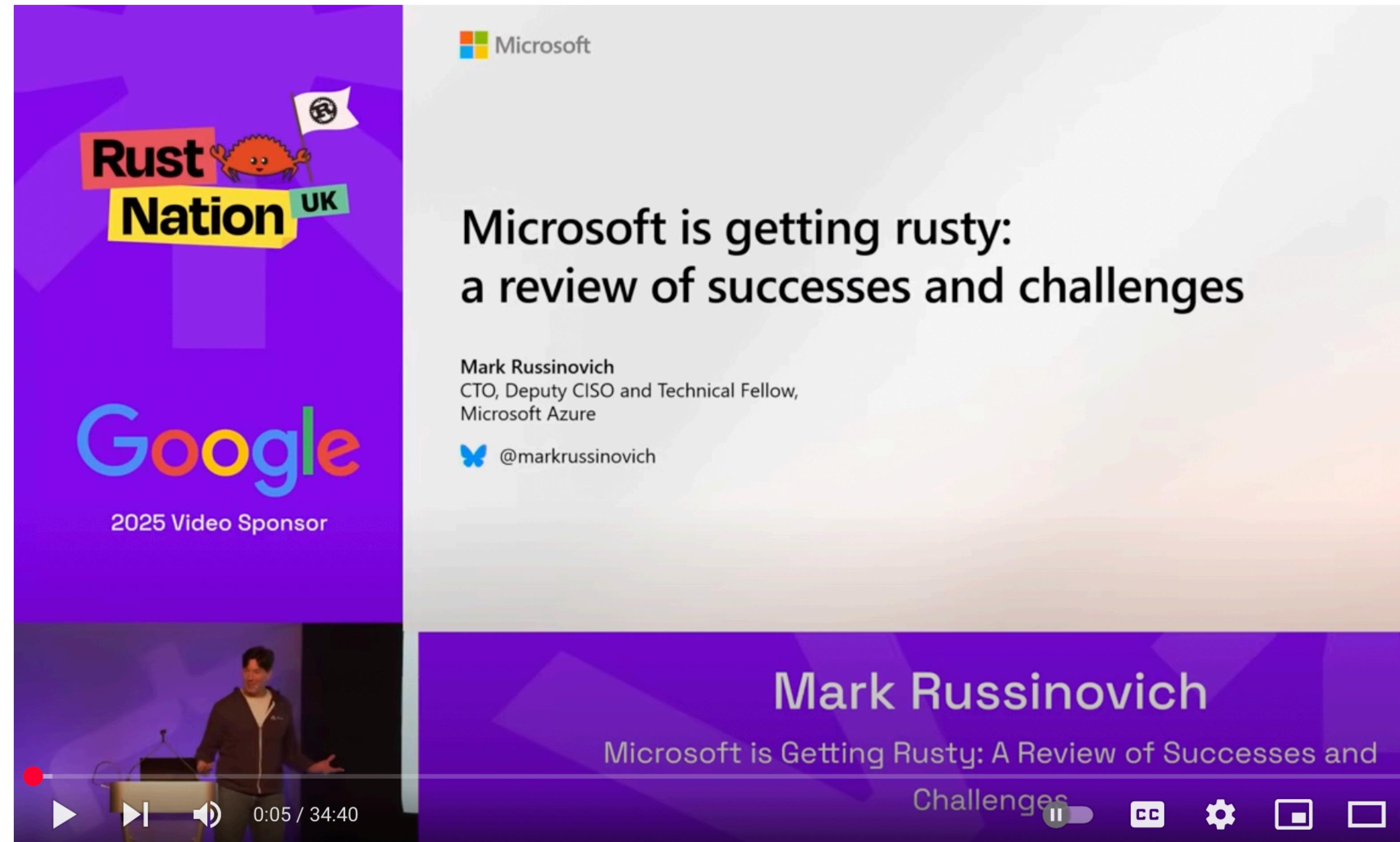
```
Directory of C:\Windows\System32
```

04/15/2023	09:50 PM	708,608	win32k.sys
04/15/2023	09:49 PM	3,424,256	win32kbase.sys
04/15/2023	09:49 PM	110,592	win32kbase_rs.sys
04/15/2023	09:50 PM	4,194,304	win32kfull.sys
04/15/2023	09:49 PM	40,960	win32kfull_rs.sys
04/15/2023	09:49 PM	69,632	win32krnl.sys
04/15/2023	09:49 PM	98,304	win32ksgd.sys
		7 File(s)	8,646,656 bytes
		0 Dir(s)	116,366,049,280 bytes free

Rusty Windows

Ported **Windows 11** core components from C++ to **Rust**

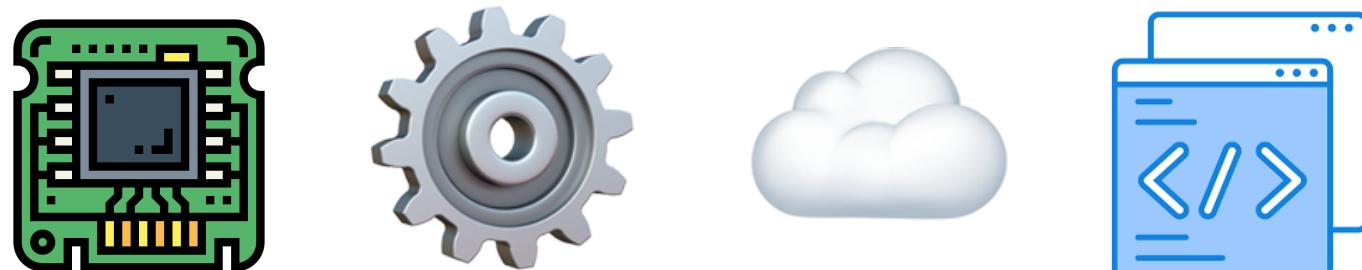
- DirectWrite
- GDI
- ... 🤪



youtube.com/watch?v=1VgptLwP588

Rust @Microsoft

- Project Mu
- Pluto security processor
- SymCrypt - rustls
- Azure Integrated HSM
- Azure Boost Agents
- Open VMM / Open HCL
- Hyper-V
- Azure SDK for Rust
- Azure Data Explorer
- Drasi
- MIMIR
- Caliptra
- Hyperlight / WASM
- ... 🤓



TBD:

- ⚙️ Windows core components
- ☁️ Microservices

Oxidation

More oxidation 🦀 efforts in progress...

C++ → Rust ← C#

TBD 😊

Rust in Production

Learn by doing: Exploration → Flighting → Production

Rust in Production

Learn by doing: Exploration → Flighting → Production

- Direct impact: improve security & reduce operation cost

Rust in Production

Learn by doing: Exploration → Flighting → Production

- Direct impact: **improve security & reduce operation cost**
- **Gain experience** with transitioning to Rust in production

Rust in Production

Learn by doing: Exploration → Flighting → Production

- Direct impact: improve security & reduce operation cost
- Gain experience with transitioning to Rust in production
- Costs of learning Rust?

Rust in Production

Learn by doing: Exploration → Flighting → Production

- Direct impact: improve security & reduce operation cost
- Gain experience with transitioning to Rust in production
- Costs of learning Rust?
- Costs of porting to Rust?

Rust in Production

Learn by doing: Exploration → Flighting → Production

- Direct impact: improve security & reduce operation cost
- Gain experience with transitioning to Rust in production
- Costs of learning Rust?
- Costs of porting to Rust?
- Costs of writing new Rust components?

Rust in Production

Learn by doing: Exploration → Flighting → Production

- Direct impact: improve security & reduce operation cost
- Gain experience with transitioning to Rust in production
- Costs of learning Rust?
- Costs of porting to Rust?
- Costs of writing new Rust components?
- Is the full pipeline of Rust tooling ready?

Rust in Production

Learn by doing: Exploration → Flighting → Production

- Direct impact: improve security & reduce operation cost
- Gain experience with transitioning to Rust in production
- Costs of learning Rust?
- Costs of porting to Rust?
- Costs of writing new Rust components?
- Is the full pipeline of Rust tooling ready?
- Dealing with debugging woes

Rust in Production

Learn by doing: Exploration → Flighting → Production

- Direct impact: improve security & reduce operation cost
- Gain experience with transitioning to Rust in production
- Costs of learning Rust?
- Costs of porting to Rust?
- Costs of writing new Rust components?
- Is the full pipeline of Rust tooling ready?
- Dealing with debugging woes
- Performance targets, POGO, etc.

Rust in Production

Learn by doing: Exploration → Flighting → Production

- Direct impact: improve security & reduce operation cost
- Gain experience with transitioning to Rust in production
- Costs of learning Rust?
- Costs of porting to Rust?
- Costs of writing new Rust components?
- Is the full pipeline of Rust tooling ready?
- Dealing with debugging woes
- Performance targets, POGO, etc.
- Costs of maintaining a hybrid C++/Rust codebase?



Ergonomic & efficient Interop

Choose... none some?

- No need for excessive `unsafe` keyword
- No perf overhead (avoid marshaling costs, eg. copying strings)
- No boilerplate or re-declarations / No C++ annotations
- Broad types support - with safety
- Avoid lowering through C FFI
- Ergonomics - with safety
- Works with dynamic libraries (including the weirdness* of Windows DLLs, CRT)
- Plays well with C++ ABI
- Easily automated
- Hybrid build systems (CMake, cargo, MSBuild, bazel, buck2...)



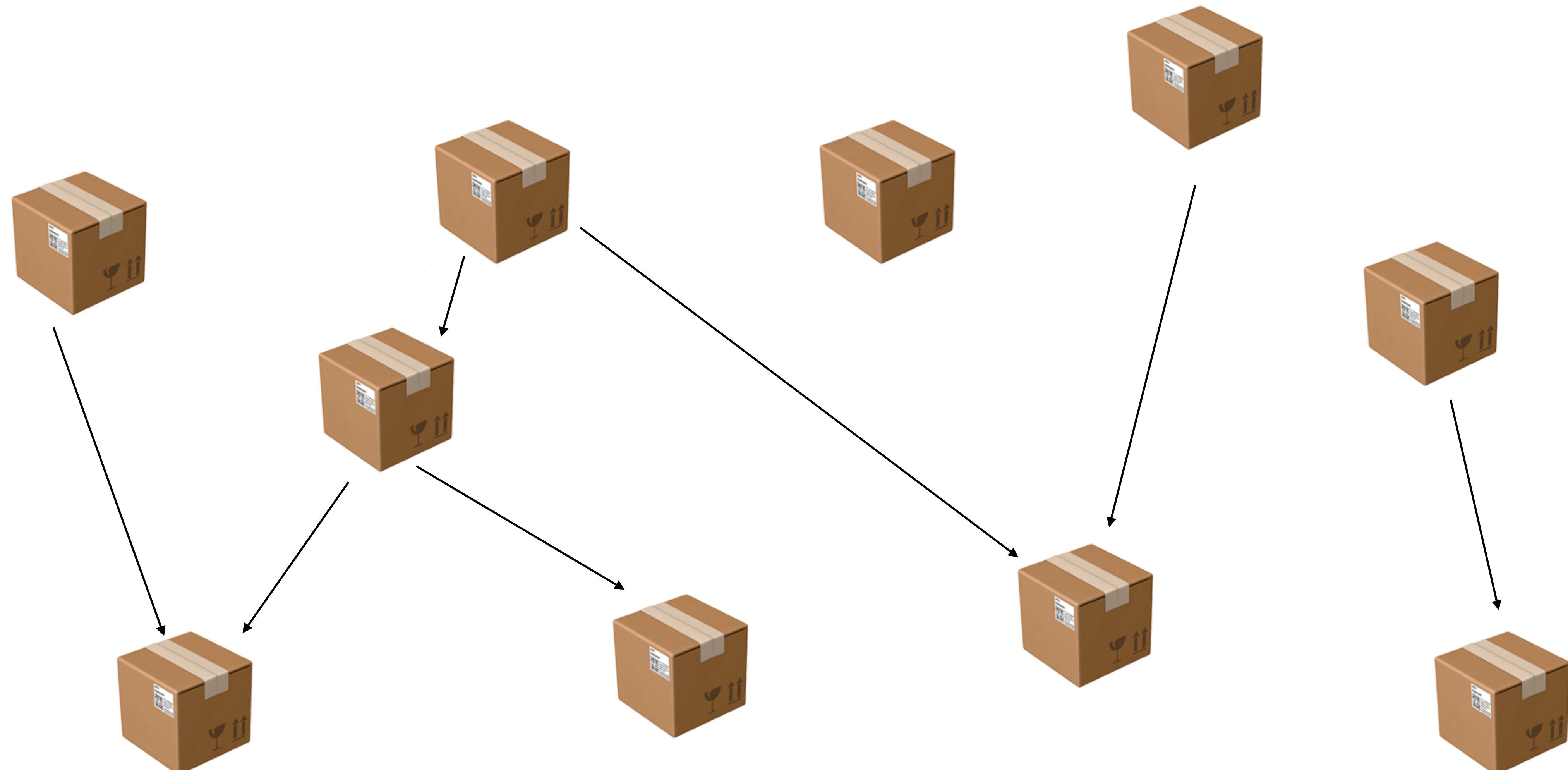
Duck-Tape Chronicles Rust/C++ Interop

Tomorrow - April 4, 09:30 

Ecosystem

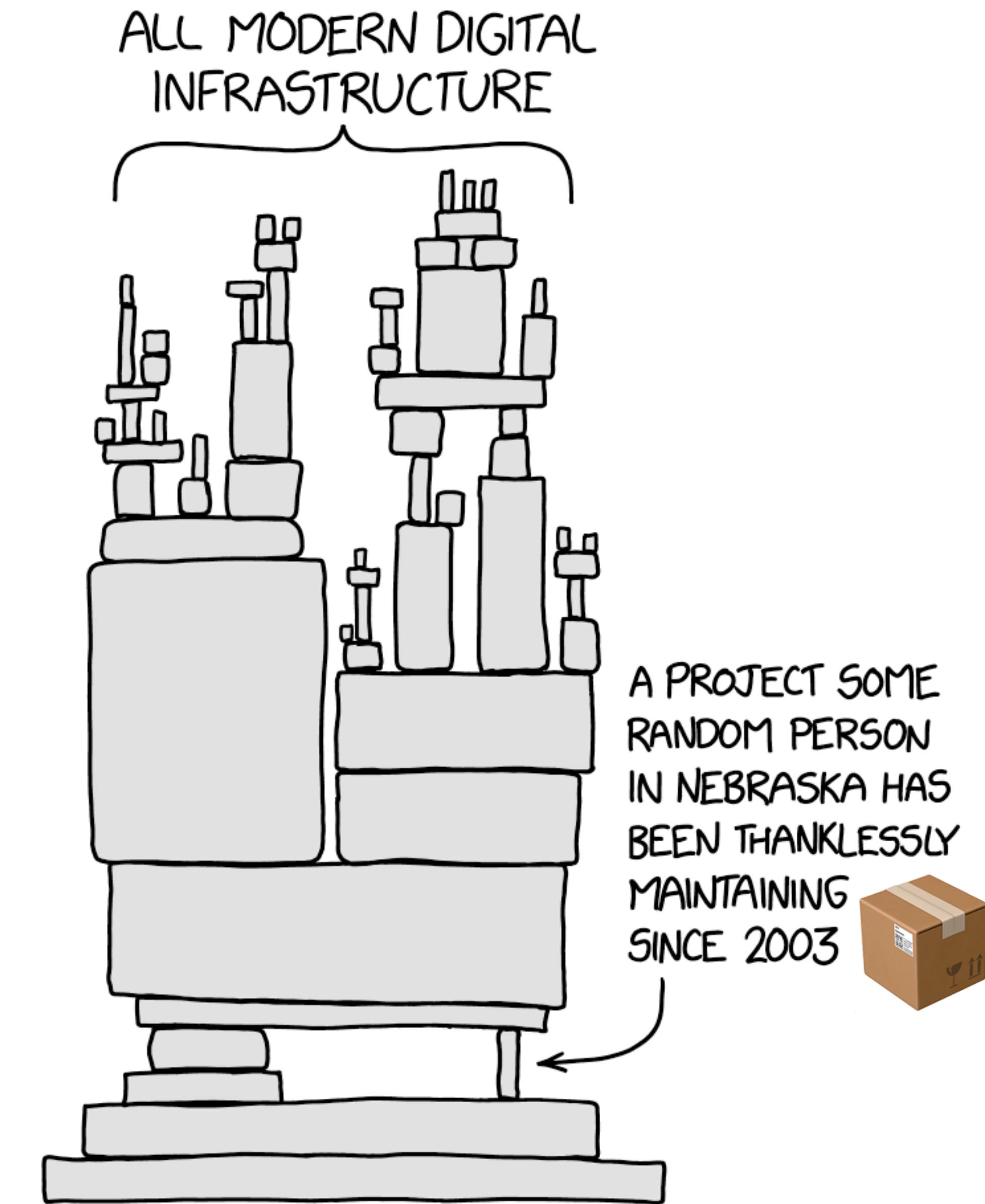
Enterprise-grade tooling

Crate Registry

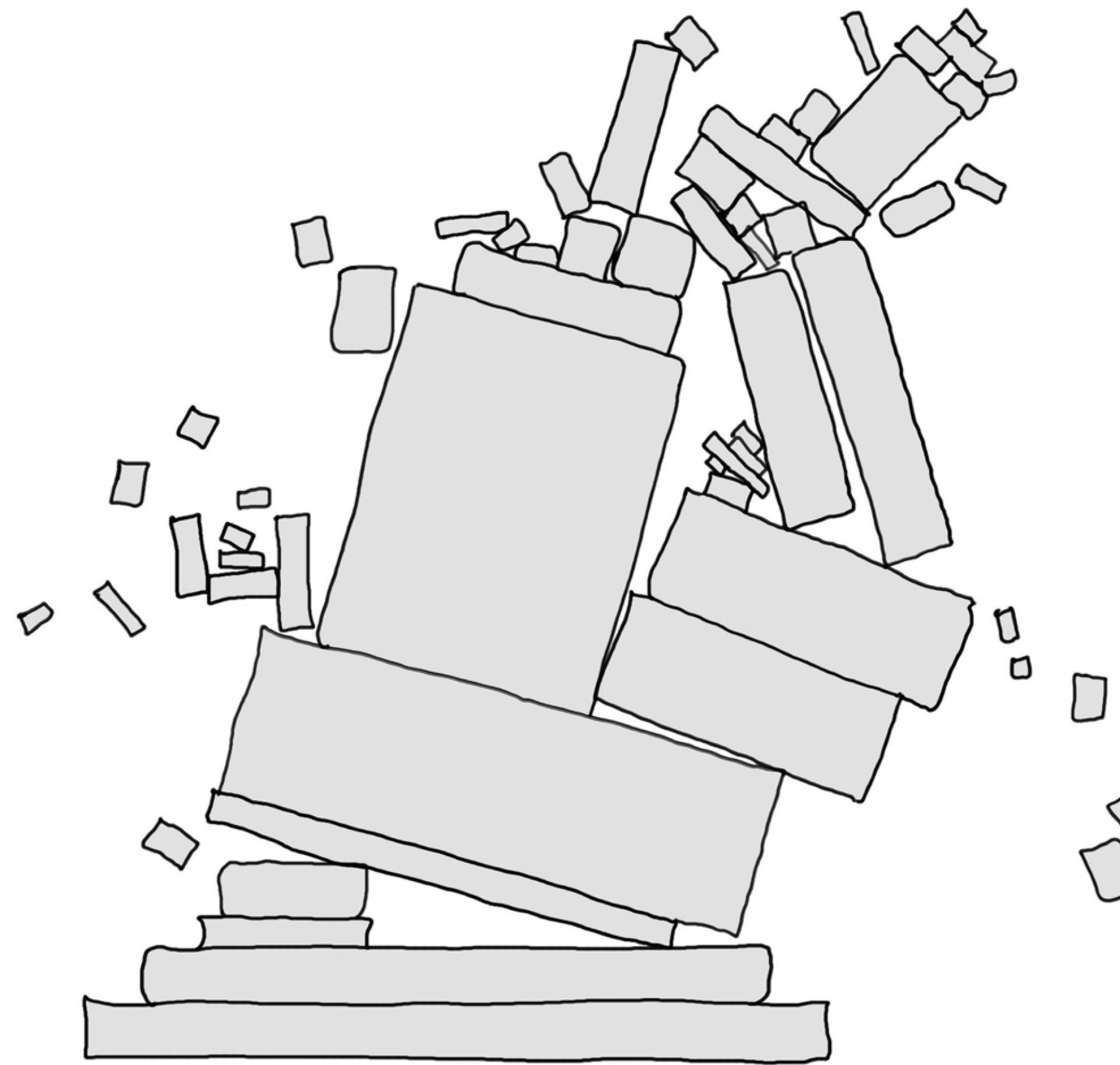


Amazing & thriving ecosystem!

Crate Registry



Crate Registry



Rust Crate Review System

A system that records **guidance** from enterprise developers on using Rust crates, both **public** and **internal** ones

- What crates should my project use, or not use?
- How should I **evaluate** public crates? (and record the evaluation)
- What are the **preferred crates** for particular purposes?
- How to keep a rigorous **SBOM** posture for the project?

Rust Crate Review System

Rust Crate Review System

- External crate dependencies, come with the inherent **risk**, in the form of potential **security** issues, **stability** issues, and **support** and **maintenance** related issues

Rust Crate Review System

- External crate dependencies, come with the inherent **risk**, in the form of potential **security** issues, **stability** issues, and **support** and **maintenance** related issues

Rust Crate Review System

- External crate dependencies, come with the inherent **risk**, in the form of potential **security** issues, **stability** issues, and **support** and **maintenance** related issues
- Proactively ensure that enterprise Rust ecosystem is built on the **stable** and **thriving** part of the OSS Rust ecosystem, lowering the risk of being affected
 - eg. vulnerability reported on a hobby-project crate, with a single owner who is not maintaining it anymore

Rust Crate Review System

- External crate dependencies, come with the inherent **risk**, in the form of potential **security** issues, **stability** issues, and **support** and **maintenance** related issues
- Proactively ensure that enterprise Rust ecosystem is built on the **stable** and **thriving** part of the OSS Rust ecosystem, lowering the risk of being affected
 - eg. vulnerability reported on a hobby-project crate, with a single owner who is not maintaining it anymore

Rust Crate Review System

- External crate dependencies, come with the inherent **risk**, in the form of potential **security** issues, **stability** issues, and **support** and **maintenance** related issues
- Proactively ensure that enterprise Rust ecosystem is built on the **stable** and **thriving** part of the OSS Rust ecosystem, lowering the risk of being affected
 - eg. vulnerability reported on a hobby-project crate, with a single owner who is not maintaining it anymore
- A set of unbiased Rust **crate evaluation criteria**, used for assessment of adoptability of third-party crates by any internal Rust project
 - lowering the company's vulnerability on third-party OSS solutions

Rust Crate Review System

- External crate dependencies, come with the inherent **risk**, in the form of potential **security** issues, **stability** issues, and **support** and **maintenance** related issues
- Proactively ensure that enterprise Rust ecosystem is built on the **stable** and **thriving** part of the OSS Rust ecosystem, lowering the risk of being affected
 - eg. vulnerability reported on a hobby-project crate, with a single owner who is not maintaining it anymore
- A set of unbiased Rust **crate evaluation criteria**, used for assessment of adoptability of third-party crates by any internal Rust project
 - lowering the company's vulnerability on third-party OSS solutions

Rust Crate Review System

- External crate dependencies, come with the inherent **risk**, in the form of potential **security** issues, **stability** issues, and **support** and **maintenance** related issues
- Proactively ensure that enterprise Rust ecosystem is built on the **stable** and **thriving** part of the OSS Rust ecosystem, lowering the risk of being affected
 - eg. vulnerability reported on a hobby-project crate, with a single owner who is not maintaining it anymore
- A set of unbiased Rust **crate evaluation criteria**, used for assessment of adoptability of third-party crates by any internal Rust project
 - lowering the company's vulnerability on third-party OSS solutions
- A **unified**, **unbiased**, highly **automatable** crate **scoring system** used throughout all teams/projects in the company



Crate security in 2025 - Adam Harvey

youtube.com/watch?v=GXkvX9A9xME

ONE DOES NOT SIMPLY



makeameme.org

ONE DOES NOT SIMPLY

But we still do it!

REWRITE IN RUST

makeameme.org



Rust: Cargo Cult?

ACCU

April 2025

 @ciura_victor

 @ciura_victor@hachyderm.io

 @ciuravictor.bsky.social

Victor Ciura
~~Principal Engineer~~
Rambling Idiot
Rust Tooling @ Microsoft