Sandboxing .NET Assemblies for fun, profit and, of course Security!

Niels Tanis

NDC { Sydney }

Who am I?

- Niels Tanis
- Principal Security Researcher @ Veracode
  - Background .NET Development, Pentesting/ethical hacking, and software security consultancy
  - ISC^2 CSSLP
  - Research on static analysis for .NET apps

Agenda

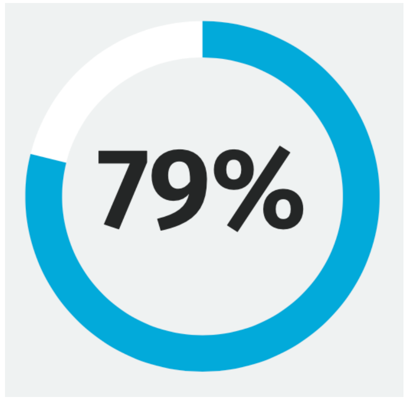
- Introduction
- The security risks of third party libraries
- Sandboxing techniques
- Let’s create a sandbox!
- Conclusion
- QA

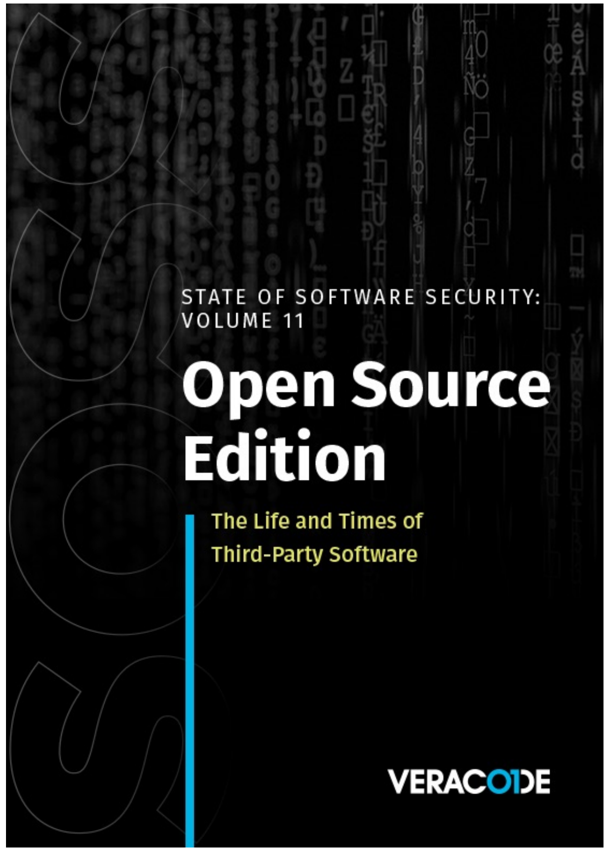
Third Party Libraries

- Big chunk (80%+) of our apps consists of 3rd party libraries
- Efficient in time, why reinvent the wheel?
- How actively is it maintained?
- What do they do for security?

State Of Software Security v11 2021

"Despite this dynamic landscape, 79 percent of the time, developers never update third-party libraries after including them in a codebase."



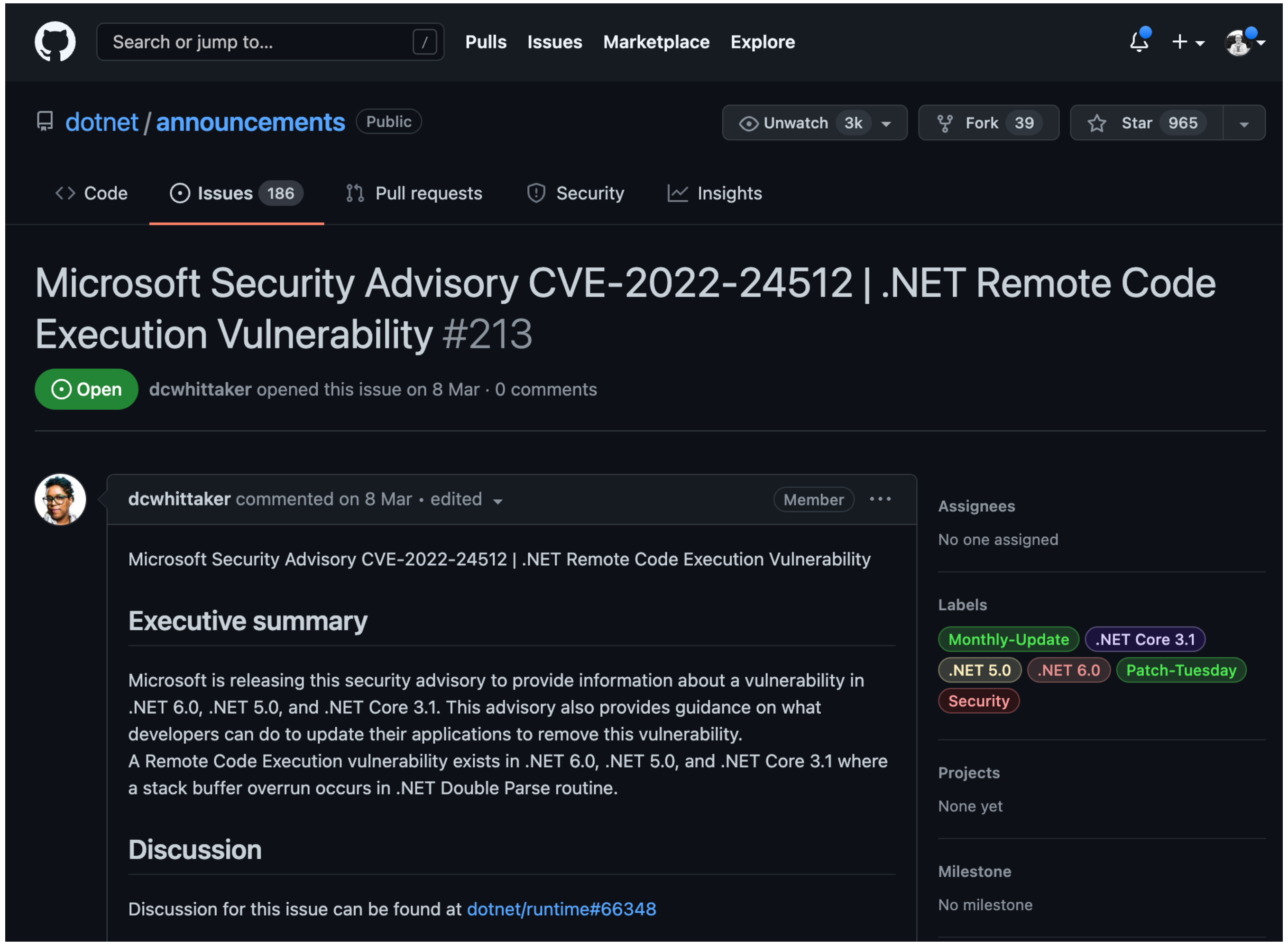

STATE OF SOFTWARE SECURITY: VOLUME 11

Open Source Edition

The Life and Times of Third-Party Software

VERACODE

Vulnerabilities in libraries



Microsoft Security Advisory CVE-2022-24512 | .NET Remote Code Execution Vulnerability #213

Open dcwhittaker opened this issue on 8 Mar · 0 comments

dcwhittaker commented on 8 Mar • edited Member

Microsoft Security Advisory CVE-2022-24512 | .NET Remote Code Execution Vulnerability

Executive summary

Microsoft is releasing this security advisory to provide information about a vulnerability in .NET 6.0, .NET 5.0, and .NET Core 3.1. This advisory also provides guidance on what developers can do to update their applications to remove this vulnerability.
A Remote Code Execution vulnerability exists in .NET 6.0, .NET 5.0, and .NET Core 3.1 where a stack buffer overrun occurs in .NET Double Parse routine.

Discussion

Discussion for this issue can be found at dotnet/runtime#66348

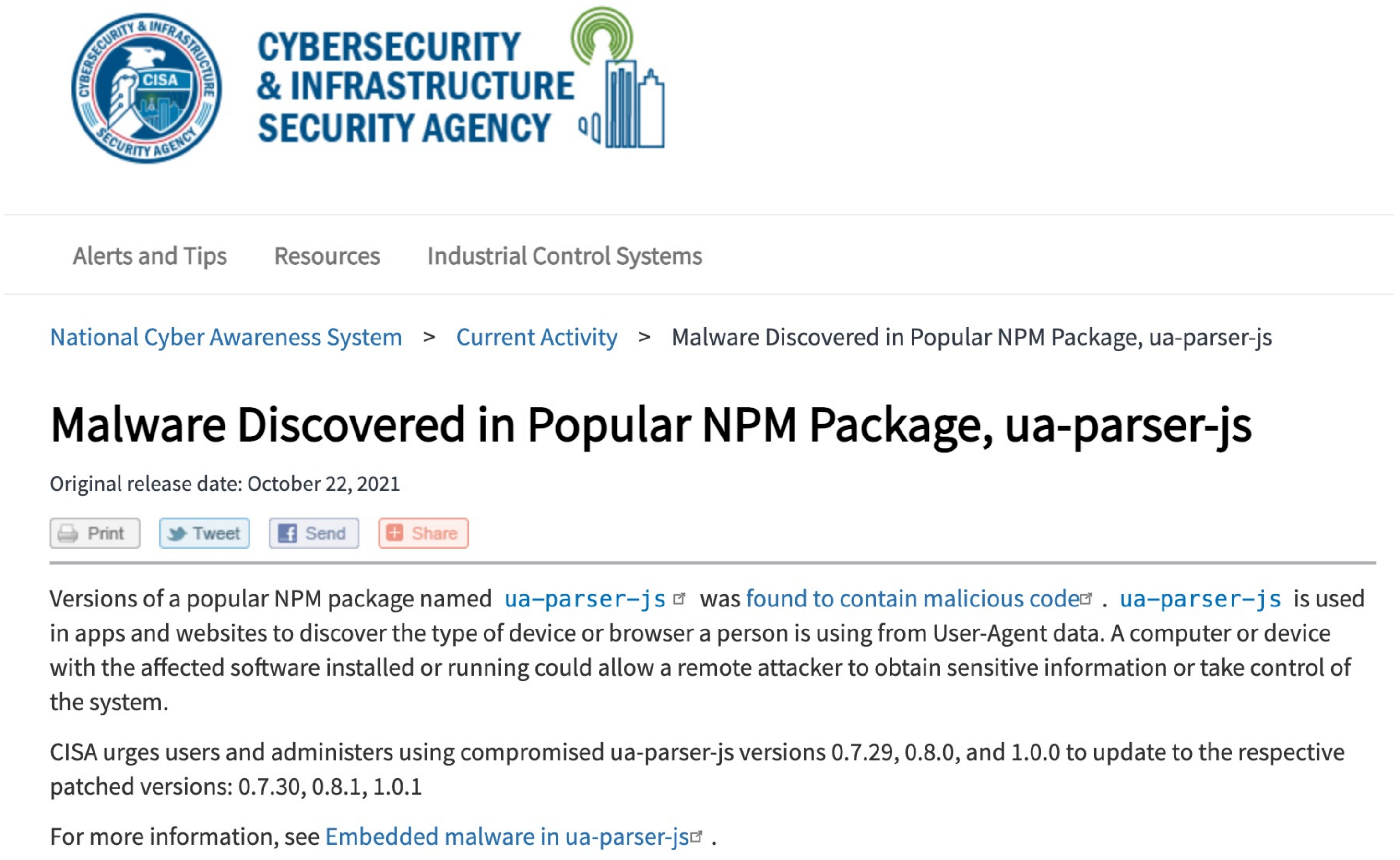
Assignees
No one assigned

Labels
Monthly-Update .NET Core 3.1 .NET 5.0 .NET 6.0 Patch-Tuesday Security

Projects
None yet

Milestone
No milestone

Vulnerabilities in libraries



CYBERSECURITY & INFRASTRUCTURE SECURITY AGENCY

Alerts and Tips   Resources   Industrial Control Systems

National Cyber Awareness System > Current Activity > Malware Discovered in Popular NPM Package, ua-parser-js

Malware Discovered in Popular NPM Package, ua-parser-js

Original release date: October 22, 2021

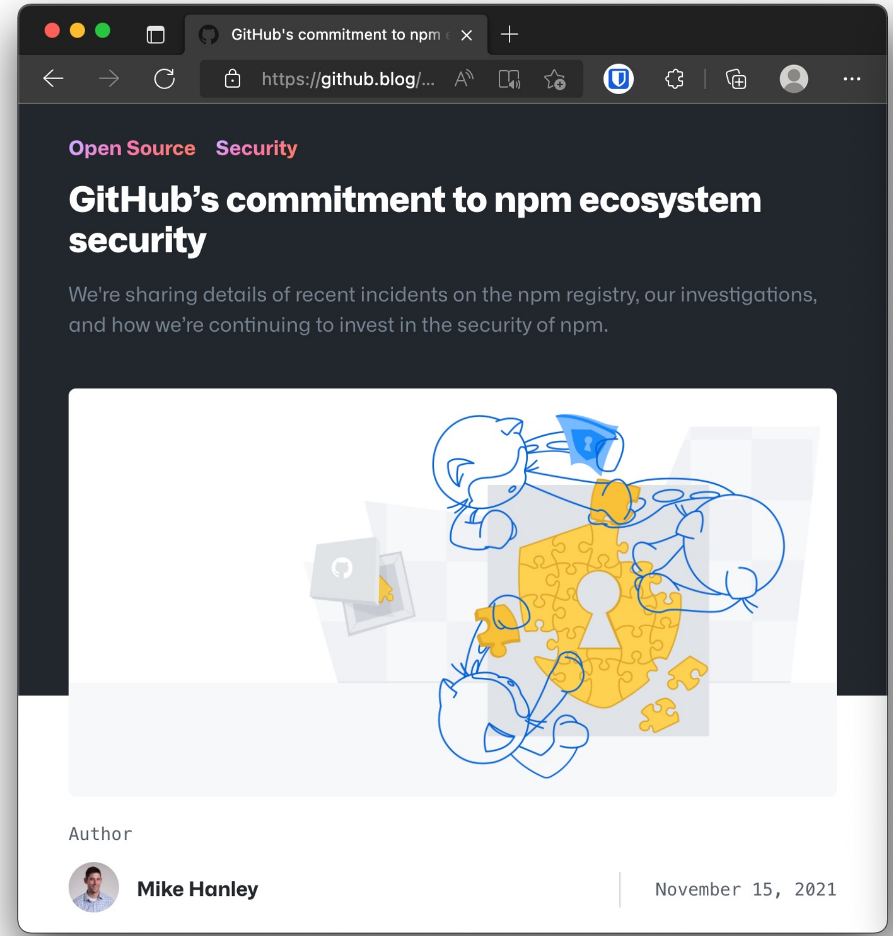
Print Tweet Send Share

Versions of a popular NPM package named `ua-parser-js` was found to contain malicious code. `ua-parser-js` is used in apps and websites to discover the type of device or browser a person is using from User-Agent data. A computer or device with the affected software installed or running could allow a remote attacker to obtain sensitive information or take control of the system.

CISA urges users and administers using compromised ua-parser-js versions 0.7.29, 0.8.0, and 1.0.0 to update to the respective patched versions: 0.7.30, 0.8.1, 1.0.1

For more information, see Embedded malware in ua-parser-js.

Vulnerabilities in libraries



GitHub's commitment to npm

https://github.blog/...

Open Source Security

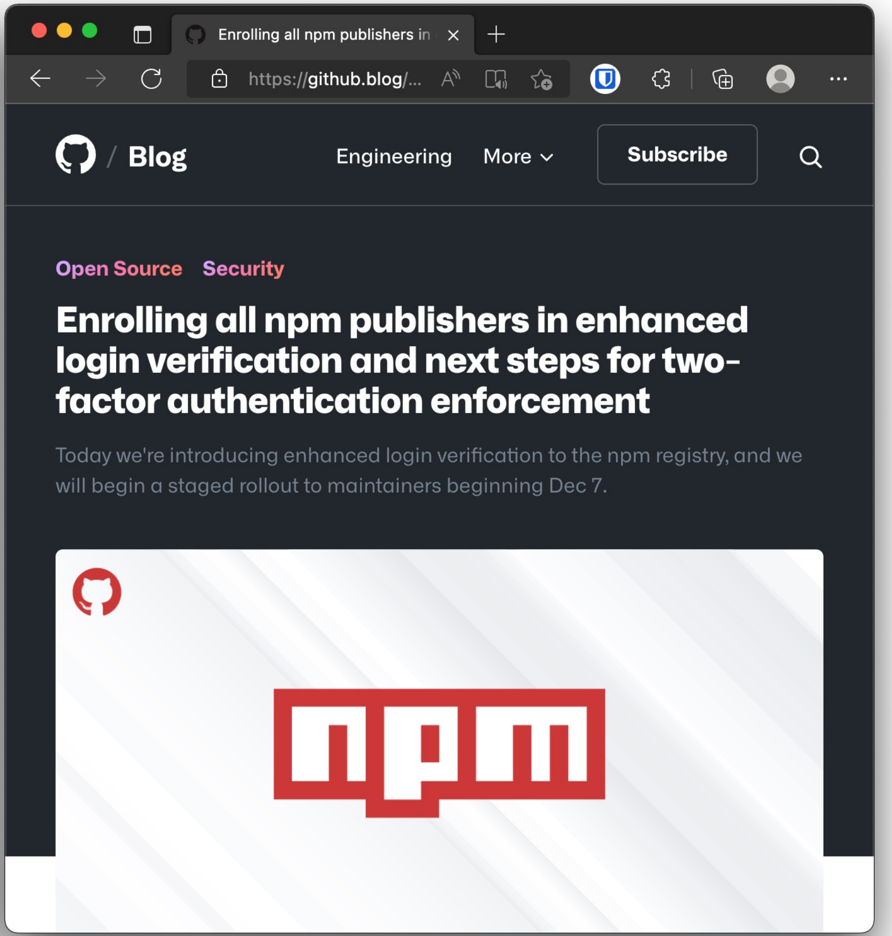
GitHub's commitment to npm ecosystem security

We're sharing details of recent incidents on the npm registry, our investigations, and how we're continuing to invest in the security of npm.

Author

Mike Hanley

November 15, 2021



Enrolling all npm publishers in

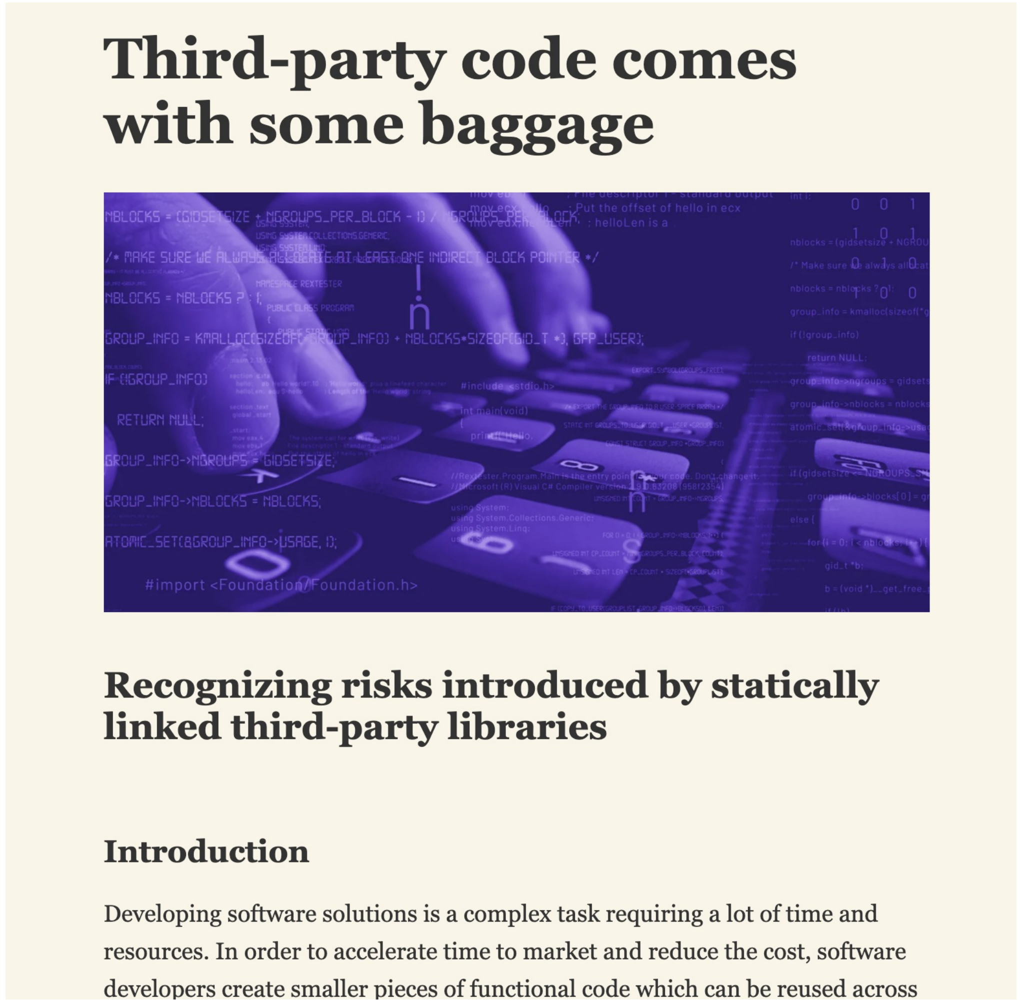
https://github.blog/...

Open Source Security

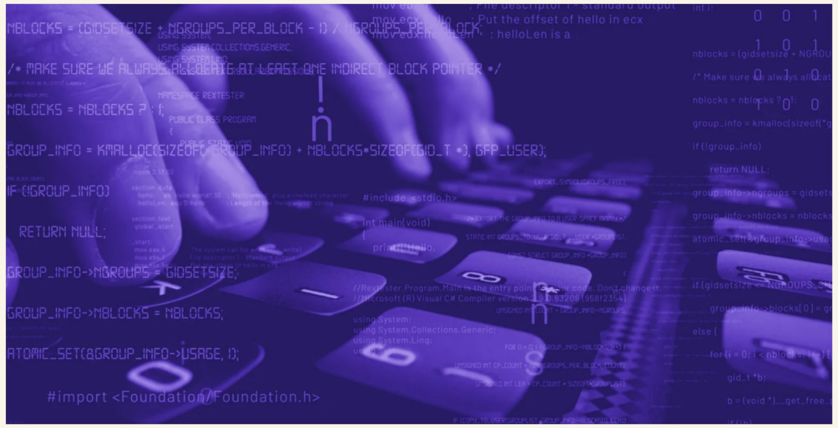
Enrolling all npm publishers in enhanced login verification and next steps for two-factor authentication enforcement

Today we're introducing enhanced login verification to the npm registry, and we will begin a staged rollout to maintainers beginning Dec 7.

Vulnerabilities in libraries



Third-party code comes with some baggage



Recognizing risks introduced by statically linked third-party libraries

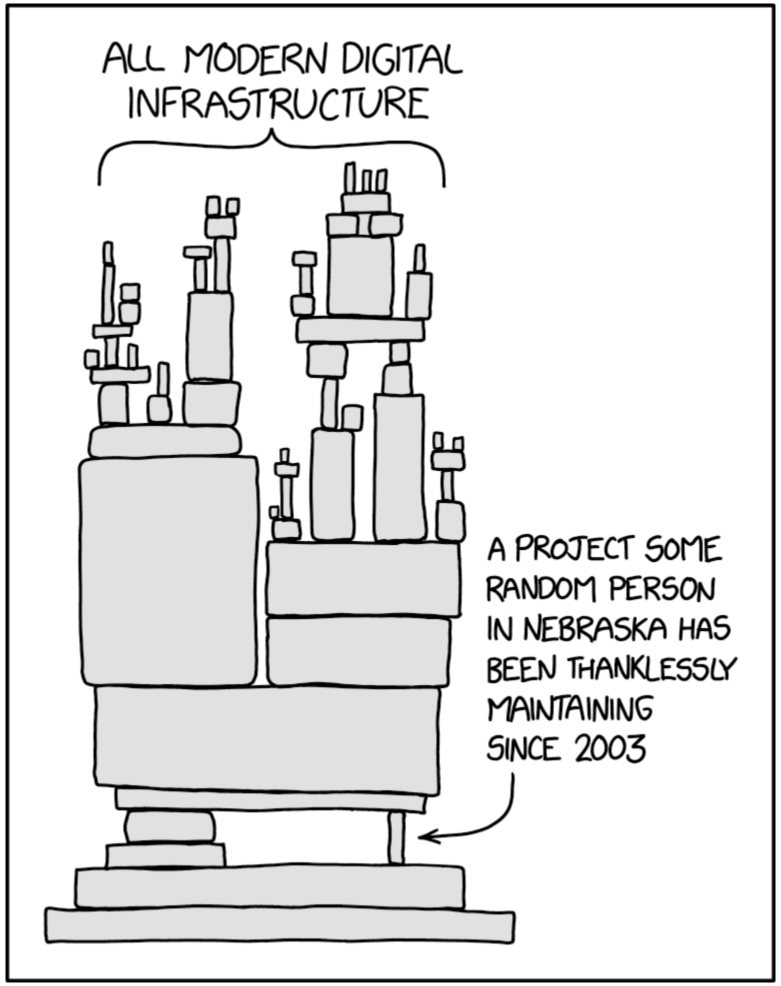
Introduction

Developing software solutions is a complex task requiring a lot of time and resources. In order to accelerate time to market and reduce the cost, software developers create smaller pieces of functional code which can be reused across

NDC { Sydney }

@nielstanis

XKDC - Dependency


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

https://xkcd.com/2347/

Sandboxing .NET Assemblies

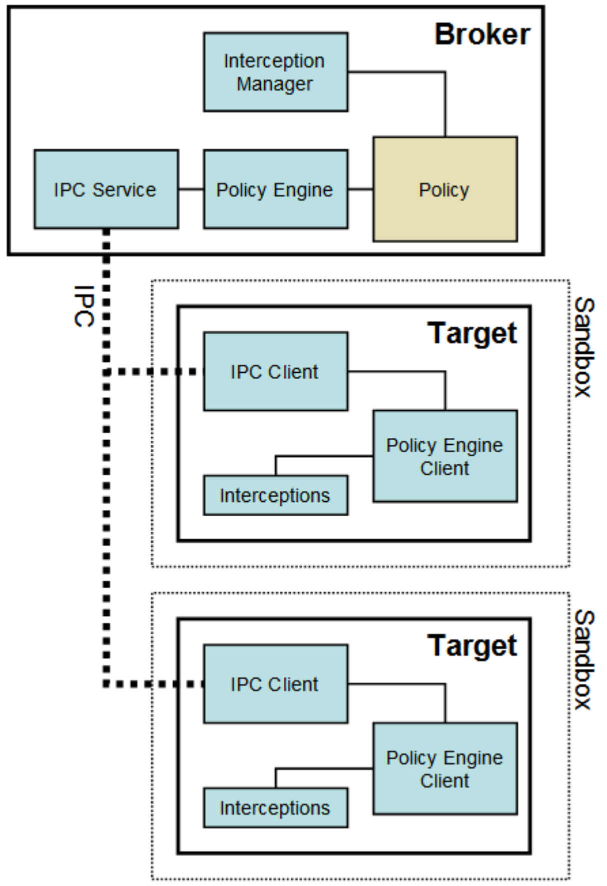
- Is there a way we can do a better job?
- A way for us to reduce the security risks?
- Keep in mind it's not a matter of how it's more when!

Sandboxing .NET Assemblies

- We want to use the library without modification
- Can we maybe create a controlled (restricted) sandbox?
- A sandbox with limited capabilities?

Browser Sandbox

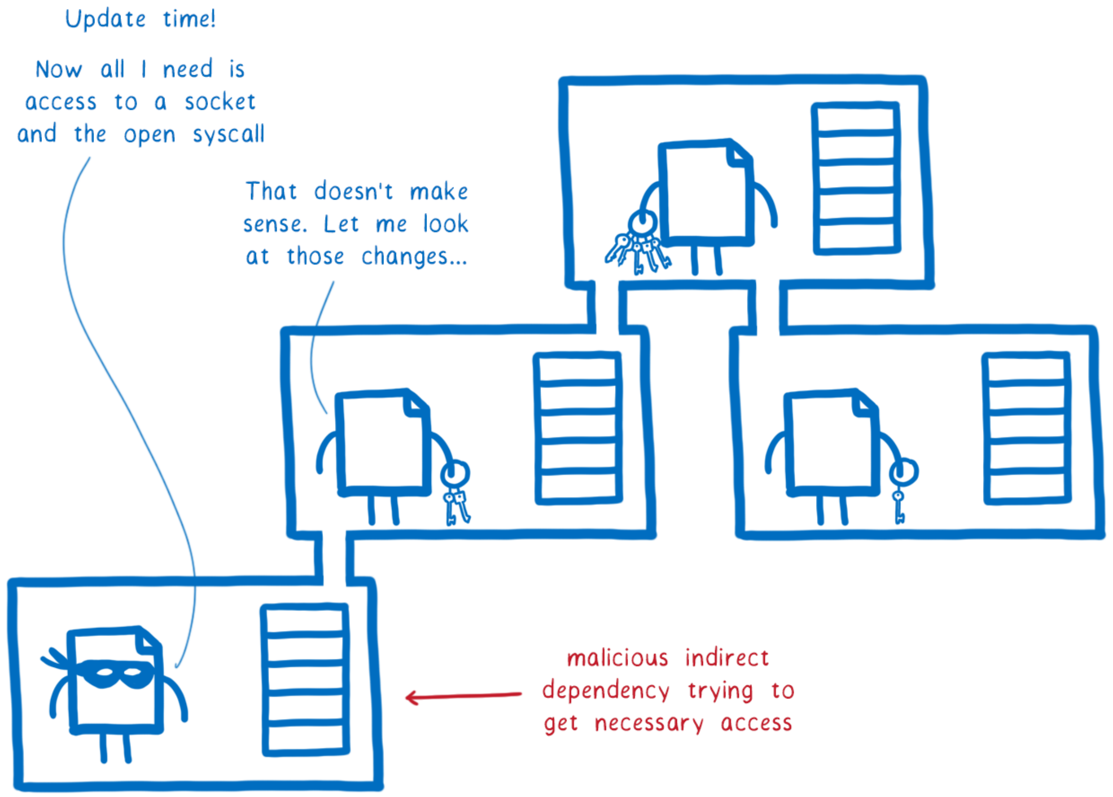
- Chromium Sandbox
- No direct system access
- Each OS related call is done via IPC
- FireFox Sandbox
  - Containers & Site Isolation
  - RLBox



WebAssembly Nanoprocess

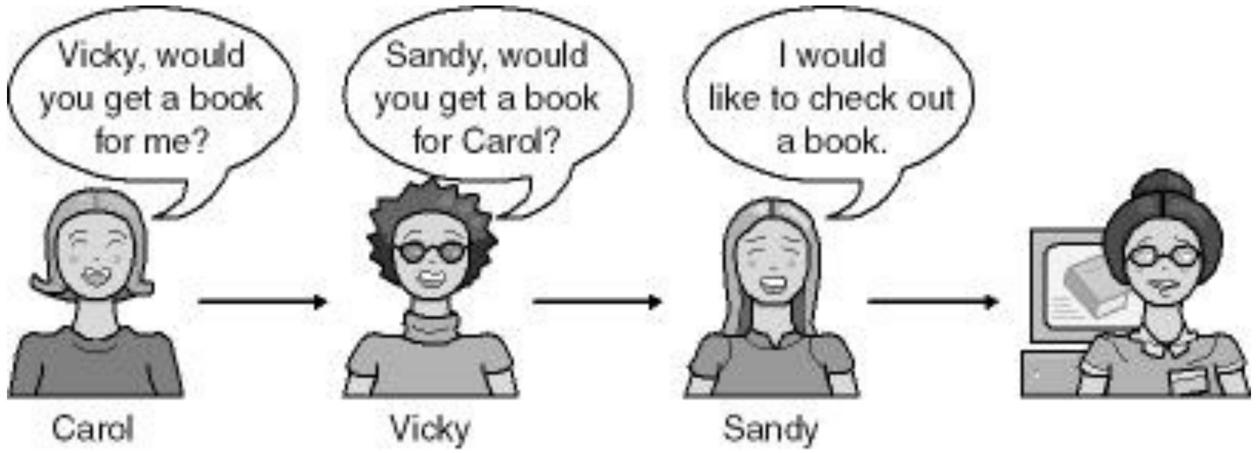
- Lineair memory model
- WASM module isolation

- Declaritive permissions
- Interface types
- WASI for BCL calls


Update time!
Now all I need is access to a socket and the open syscall
That doesn't make sense. Let me look at those changes...
malicious indirect dependency trying to get necessary access

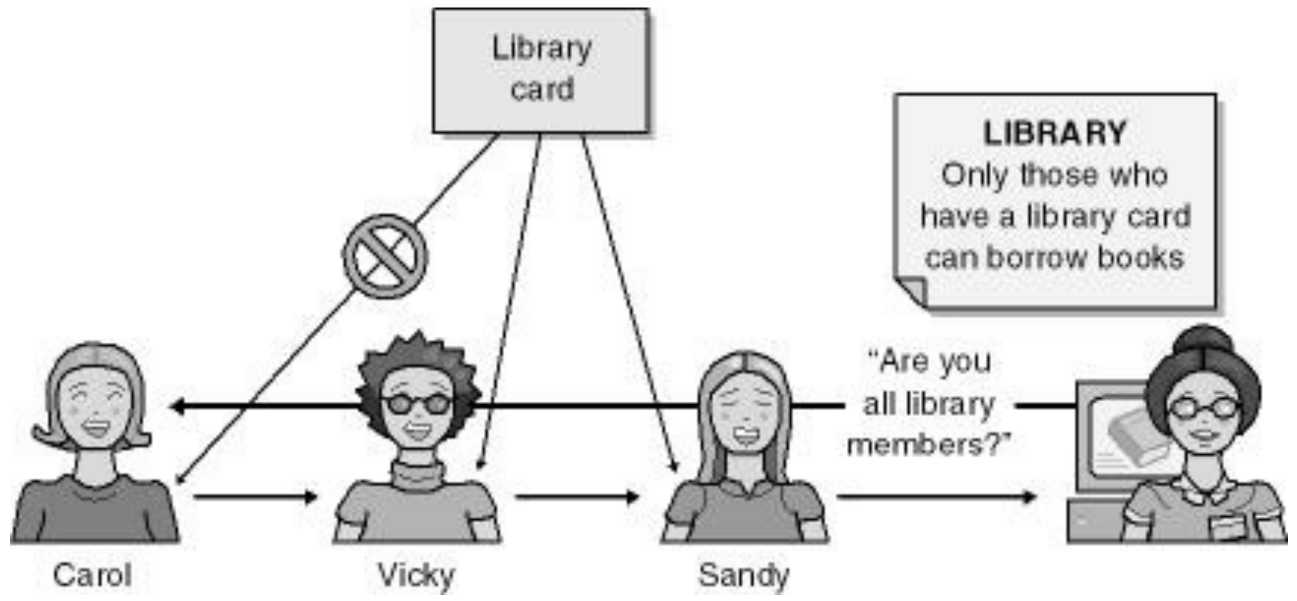
Code Access Security

- Evidence based model
- Code from different origins have different sets of rights
- Stack-walks that protect against luring attacks


Vicky, would you get a book for me?
Sandy, would you get a book for Carol?
I would like to check out a book.
Carol
Vicky
Sandy

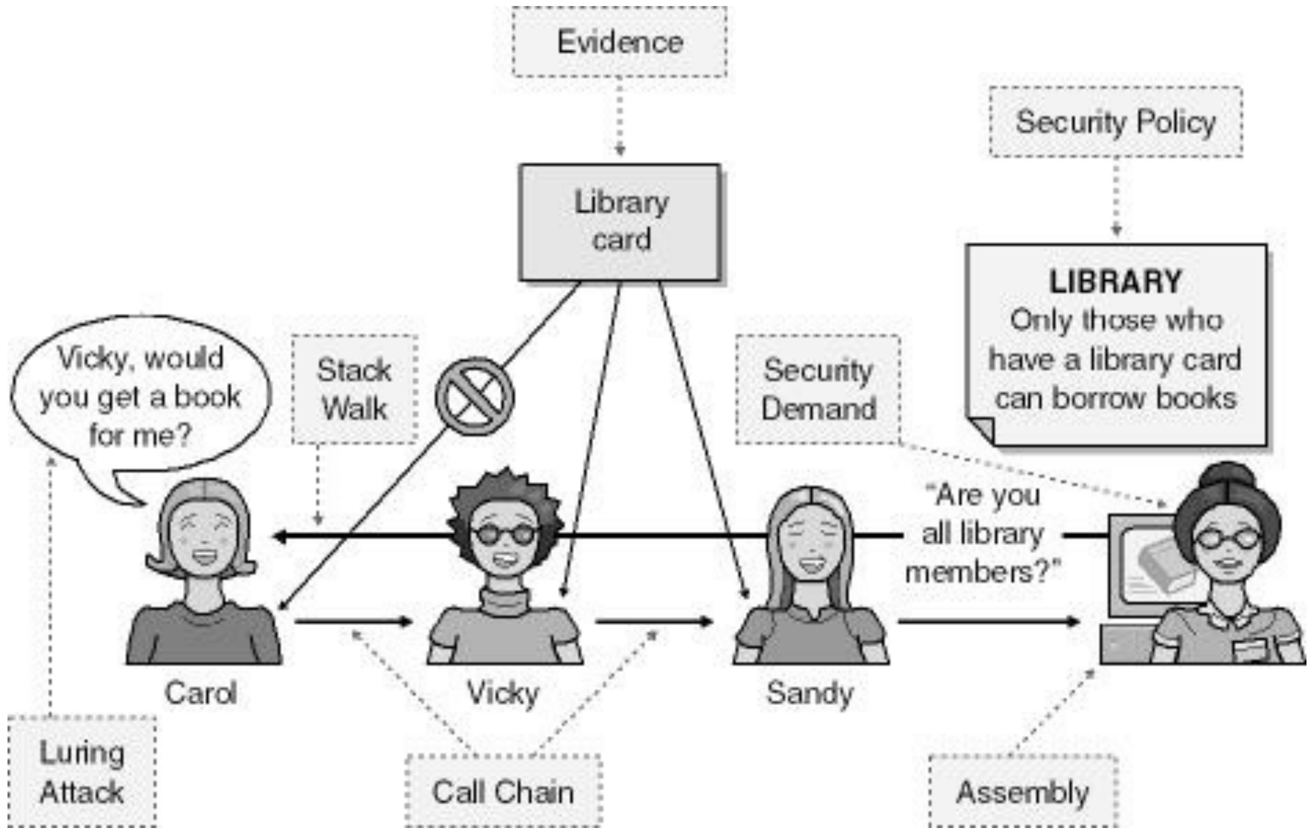
Code Access Security

- Evidence library card
- Policy → Librarian only allows members


Library card
LIBRARY
Only those who have a library card can borrow books
"Are you all library members?"
Carol
Vicky
Sandy

Code Access Security

- Stack walk


Evidence
Library card
Security Policy
LIBRARY
Only those who have a library card can borrow books
Vicky, would you get a book for me?
Stack Walk
Security Demand
"Are you all library members?"
Carol
Vicky
Sandy
Luring Attack
Call Chain
Assembly

Code Access Security

- Most practical example, ASP.NET Medium Trust
- CAS is deprecated since .NET Framework 4
- Too complex in administering and use?
- Too early?

Demo time!

DocumentProcessor Package

- Use package as is!
  - Disclaimer: always comply with library license!
  - Not allowed to reverse engineer/decompile
- We do want to change behaviour:
  - Opening documents directly from URL – SSRF
  - Writing files to any arbitrary directory – Path Traversal
- There are *several* ways to *fix* this!

AssemblyLoadContext

- Only single AppDomain in .NET Core.
- AssemblyLoadContext replaces the isolation mechanisms provided by multiple AppDomain instances in .NET Framework.
- Conceptually, a load context creates a scope for loading, resolving, and potentially unloading a set of assemblies.

AssemblyLoadContext

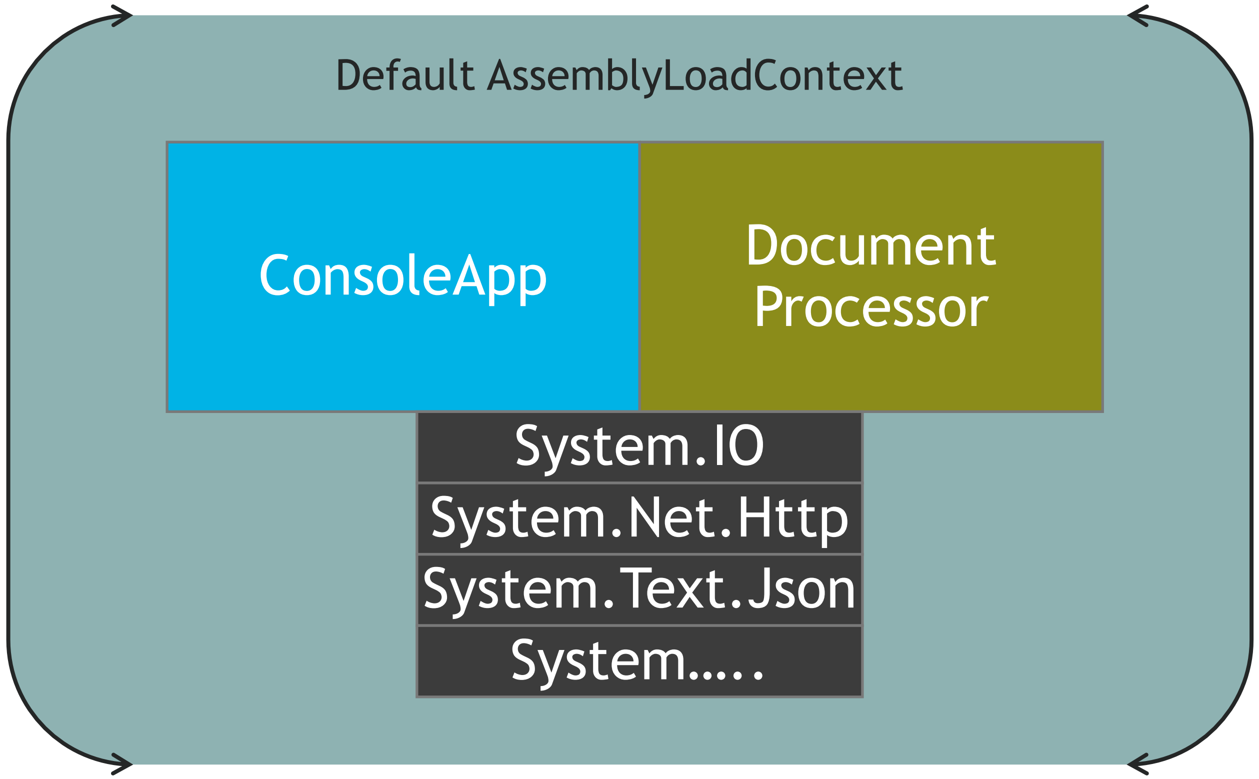
- It allows multiple versions of the same assembly to be loaded within a single process.
- It does not provide any security features. All code has full permissions of the process.
- But it does allow us to control what gets loaded!

AssemblyLoadContext

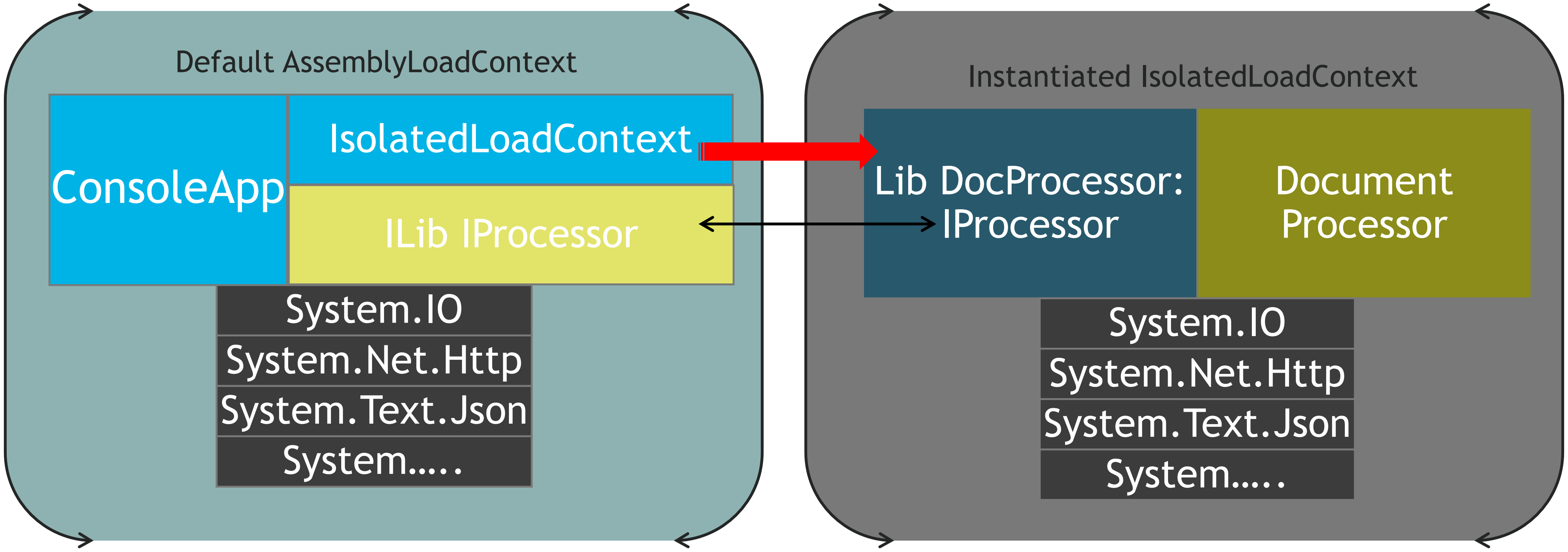
- Interface project used as shared contract
- Remove DocumentProcessor package from ConsoleApp
  - Add reference to interface project
- Create Library that implements interface
  - Reference interface project and DocumentProcessor Package
  - Self-contained deployment to folder that has all to be loaded by our sandboxed loadcontext

Sandboxing DocumentProcessor

ConsoleApp Start



ConsoleApp & Sandboxed Library



Removing Types?

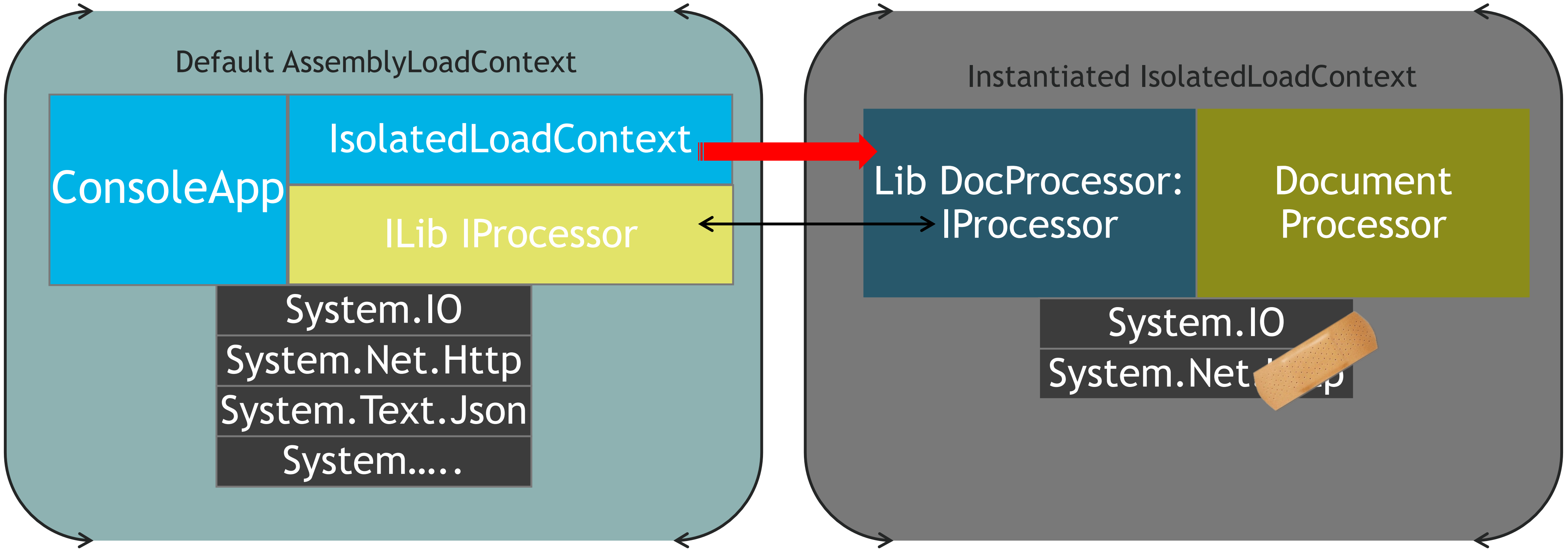
- Self contained set of assemblies, could we not remove types?
- What about trimming that got introduced with .NET 5?
- Maybe we need something more rigorous?

Patching with Harmony2

- A library for patching, replacing and decorating .NET and Mono methods during runtime.
  - Patch at runtime (pre- and postfix)
  - Transpile at compile time (rewrite IL)
- Harmony v2
  - Lib.Harmony on NuGet
  - https://github.com/pardeike/Harmony

Sandbox & Patching with Harmony2

ConsoleApp & Sandboxed Library



Conclusion

- Update libraries; security problems get fixed
- Integrate security into your development lifecycle
- Know what libraries are used, where and what’s inside and most important what you’d expect from it.

Conclusion

- Futures of this Sandbox Concept
  - Easier developer integration (e.g. source generator)
  - Package + good guidance on how this can be used in different application contexts like ASP.NET Core.
  - Basic patches/policy that can be applied on libraries

Thanks! Questions?

https://github.com/nielstanis/ndcsydney2022
ntanis at veracode.com
@nielstanis on Twitter