05 - STDs*and Dependencies

*Software Transmitted Disease

What is a software dependency?

hello.c

```
#include <stdio.h>

void mygetname(char *);
void myprinthello(char *);

int main () {
    char name[20] = "";

    mygetname(name);
    myprinthello(name);

    return(0);
}

void mygetname(char *Name) {
    printf("name> ");
    fgets(Name, 20, stdin);
    return;
}

void myprinthello(char *Name) {
    printf("hello, %s\n", Name);
    return;
}
```

hello.h

```
#include <stdio.h>

void mygetname(char *);
void myprinthello(char *);
```

hello.c

```
#include <hello.h>

int main () {
    char name[20] = "";

    mygetname(name);
    myprinthello(name);

    return(0);
}
```

hellolib.c

```
#include <hello.h>

void mygetname(char *Name) {
    printf("name> ");
    fgets(Name, 20, stdin);
    return;
}

void myprinthello(char *Name) {
    printf("hello, %s\n", Name);
    return;
}
```

hello.h

```
#include <stdio.h>

void mygetname(char *);
void myprinthello(char *);
```

hello.c

```
#include <hello.h>

int main () {
    char name[20] = "";

    mygetname(name);
    myprinthello(name);

    return(0);
}
```

hellolib.c

```
#include <hello.h>

void mygetname(char *Name) {
    printf("name> ");
    fgets(Name, 20, stdin);
    return;
}

void myprinthello(char *Name) {
    printf("hello, %s\n", Name);
    return;
}
```

- Libraries written by other teams
- Libraries written by other companies
- “Standard” libraries that come with your language
- Package managers
- Modules from open source projects
- Container images


Would you eat this cookie?


Would you insert a found drive into your laptop?

```
$ npm install left-pad
```

```
npm ERR! 404 ‘left-pad’is not in the npm registry
```

https://qz.com/646467/how-one-programmer-broke-the-internet-by-deleting-a-tiny-piece-of-code/

Simple Node.js Code

https://github.com/MrDataScientist/Nodejs-10-projects-examples

```
# npm install -g express
# npm install -g express-generator
$ express express-website
$ cd express-website/
$ npm install
up to date, audited 122 packages in 629ms
19 vulnerabilities (3 low, 3 moderate, 8 high, 5 critical)
To address all issues (including breaking changes), run:
  npm audit fix --force
Run `npm audit` for details.
```

```
{
..
"dependencies": {
    "body-parser": "~1.13.2",
    "cookie-parser": "~1.3.5",
    "debug": "~2.2.0",
    "express": "~4.13.1",
    "jade": "~1.11.0",
    "morgan": "~1.6.1",
    "nodemailer": "^1.11.0",
    "serve-favicon": "~2.3.0"
  }
}
```

I love to pick on Node.js but … Rust too

In cargo.toml

```
[package]
name = "mypackage"
version = "0.0.1"

[dependencies]
foo = "0.1"
bar = {git = "https://github.com/example/project", package = "foo" }
baz = { version = "0.1", registry = "custom", package = "foo" }
```

In the code:

```
extern crate foo; // crates.io
extern crate bar; // git repository
extern crate baz; // registry `custom`
```

```
$ go get .
```

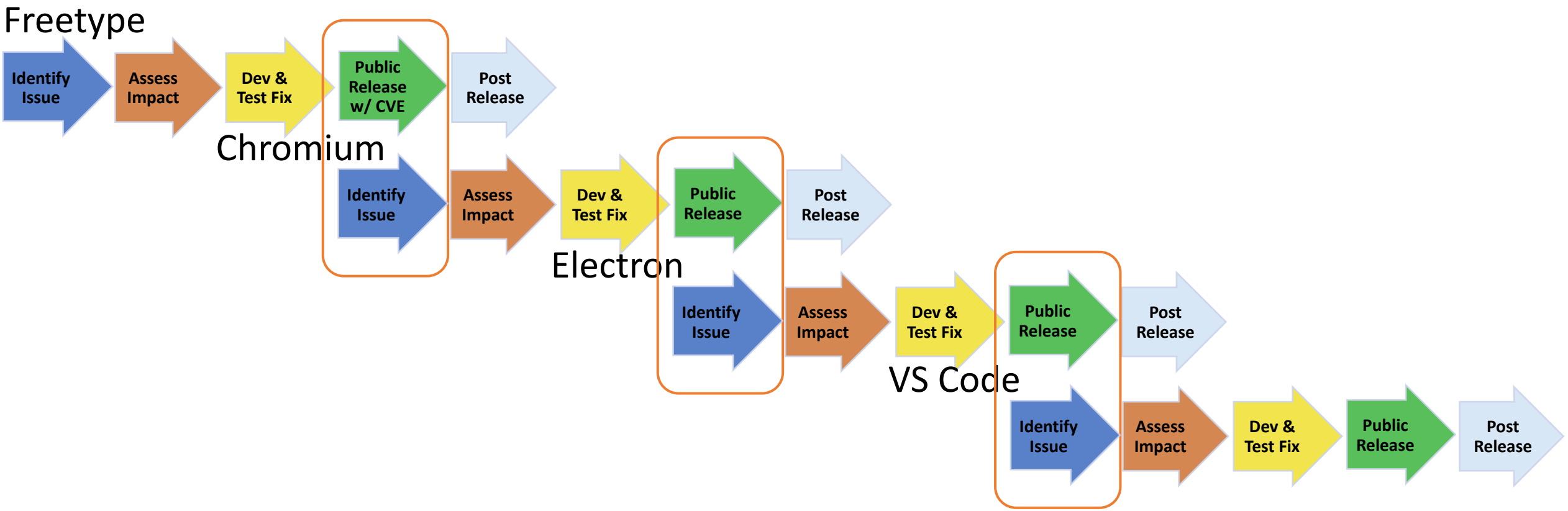
But WAIT – it’s worse than that …

SSIRP #19001

- CVE-2019-5736: Escape from Docker and Kubernetes containers to root on host
- Microsoft notified ~18 Jan 2019.
- Embargo until 11 Feb. Publication risk == Zero Day exploit.
- runc is part of the Docker container runtime – so everywhere Docker CE or EE or MS Moby runs, we needed to have a plan that worked under embargo

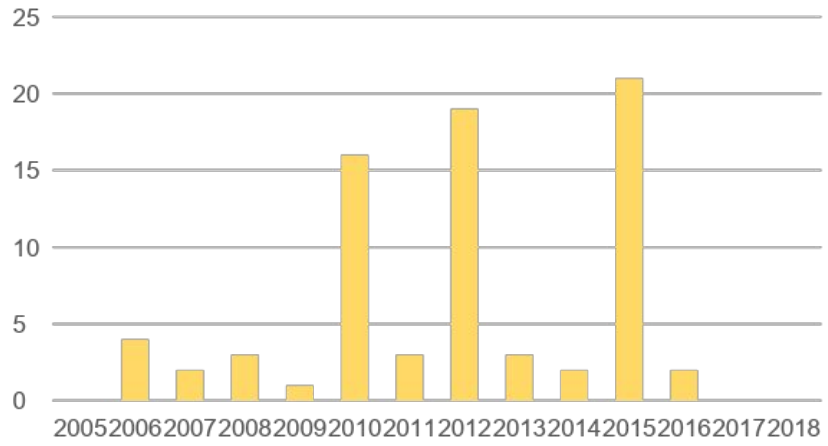
But WAIT – it’s worse than that …

OSS Vulnerability Lifecycle

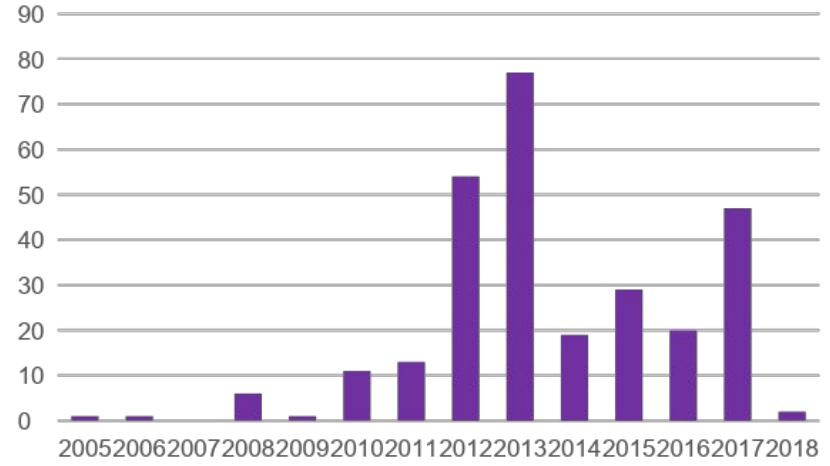


Component Vuln Data

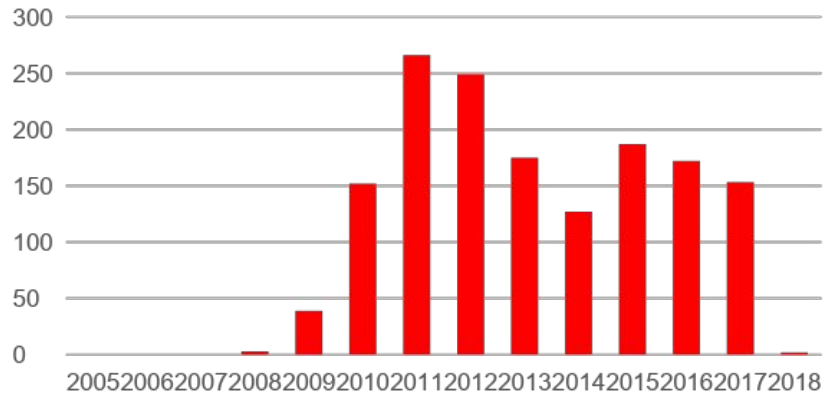
Freetype



FFMPEG

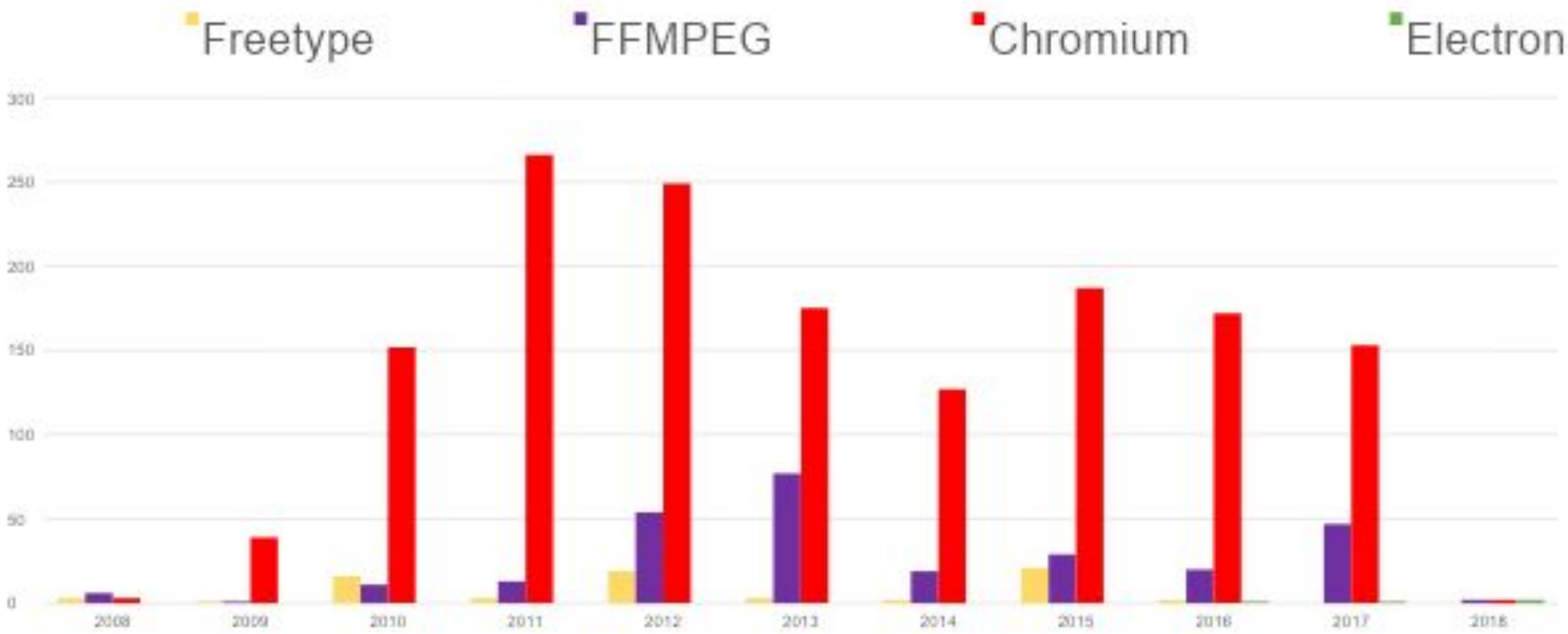


Chromium



Electron

Component Vuln Data



In a world of promiscuous software communities, STDs* are real

* Software Transmitted Disease

Digging Deeper on Dependencies

Dependency Management

- How do we update between versions of external dependencies?
- How do we discuss and describe versions of those dependencies?
- What types of changes are ‘allowed’?
- How do we decide on when we depend on 3rd party code?

Source Control vs Dependency Management

“All else being equal, prefer source control problems over dependency management problems.”

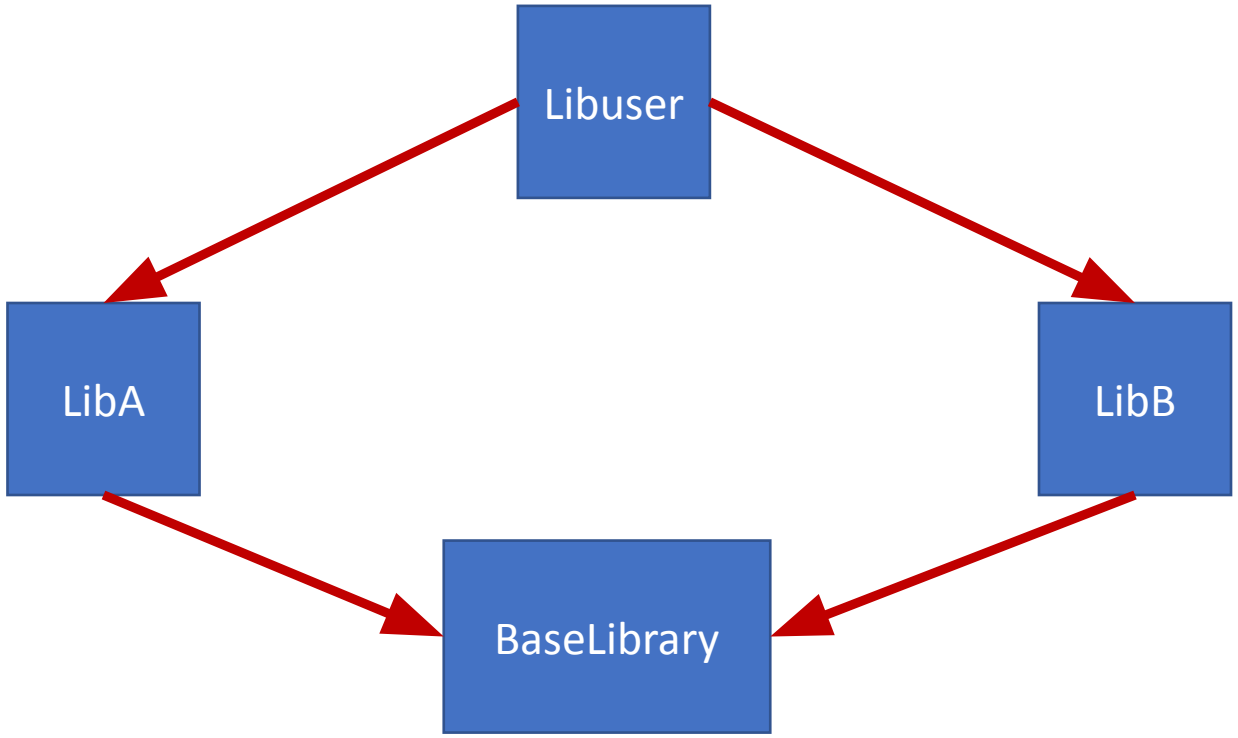
Hyrum’s Law

With a sufficient number of users of an API,
it does not matter what you promise in the contract:
all observable behaviors of your system
will be depended on by somebody.

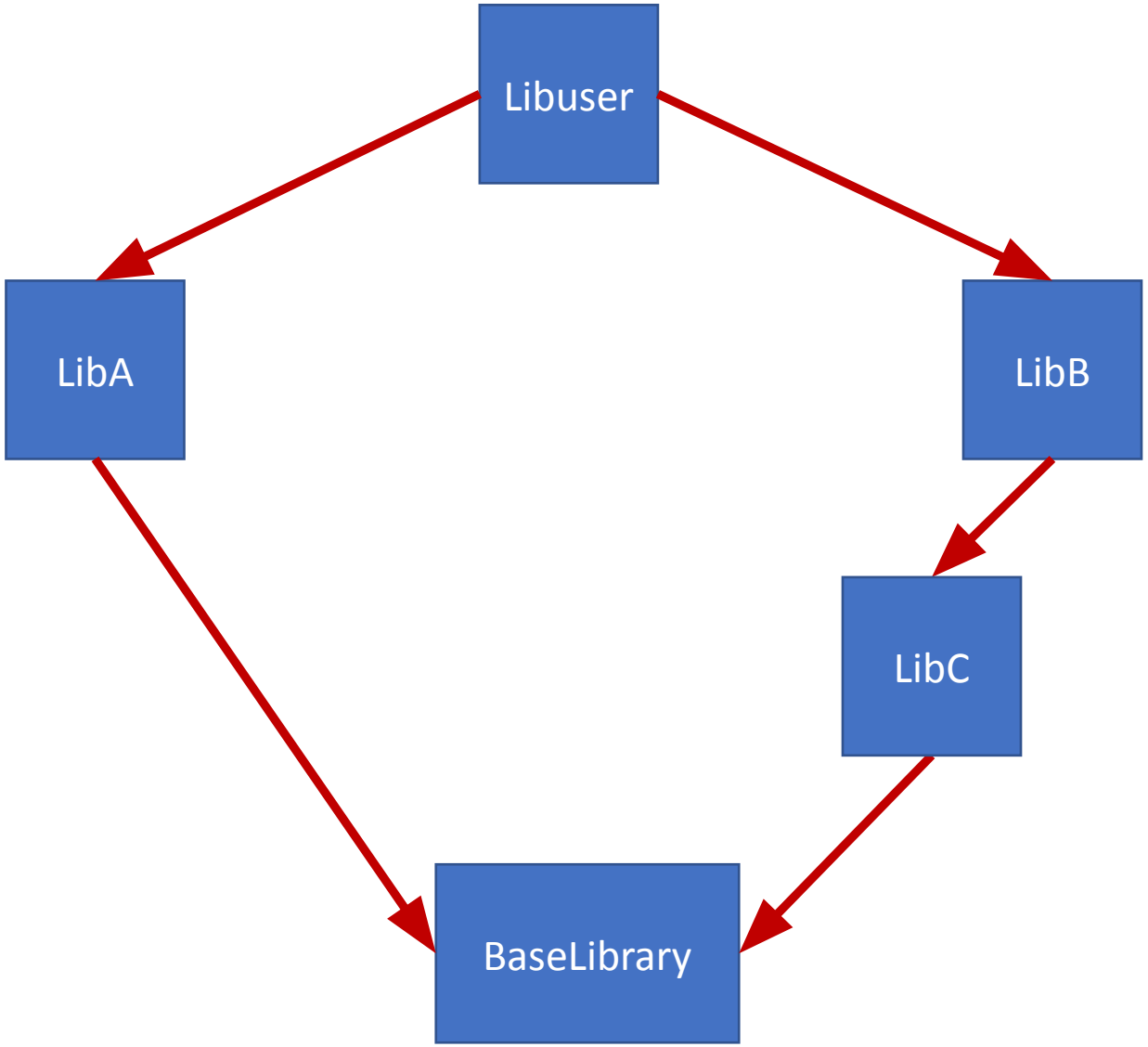
The Difficulty in Dependency Management

- It’s not about managing a single dependency – it’s the network of dependencies
- Dependencies each evolve over time
- Over a long enough period of time, all the nodes in the dependency graph will have new versions

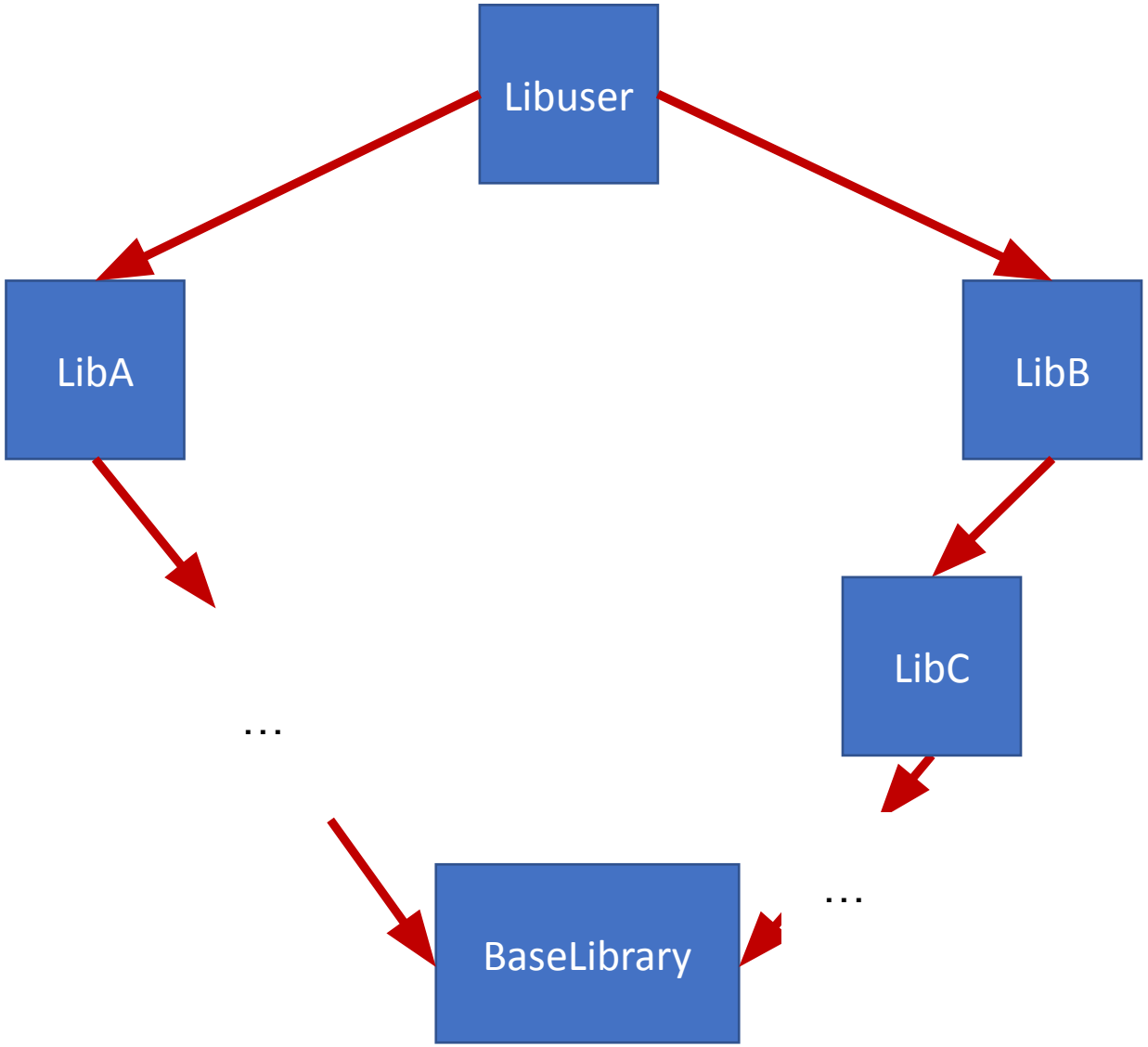
Diamonds



Diamonds



Diamonds



Diamond discovery is a hard problem ...

Engineering Cost Trade-off of Development vs Sustainability and Maintenance

Questions (from Google essay):

- Does the project have tests?
- Do those tests pass?
- Reputation of provider?
- Does the project explain its compatibility goals?
- Does the project detail expected usage?
- Project popularity?
- How long might we depend on the project?
- How often does the project make breaking changes?

Software Engineering at Google https://abseil.io/resources/swe-book

CACM: Why Google Stores Billions of Lines of Code in a Single Repository, https://cacm.acm.org/research/why-google-stores-billions-of-lines-of-code-in-a-single-repository/

Policy and Alice and Bob … and Charlie

What are the 4 Common Ways to Manage Dependencies

A Theory of Dependency Management

- Nothing Changes (Static Dependencies)
- Semantic Versioning
- Bundled Distribution Models
- Live at Head

Semantic Versioning

- Changes to the micro number only must be both forward- and backward-compatible. The changes should be bug fixes only.
- Changes to the minor number must be backward-compatible, but not necessarily forward-compatible. It's normal to introduce new features in a minor release, but usually not too many new features at once.
- Changes to the major number mark compatibility boundaries. A new major release can be forward- and backward-**incompatible**. A major release is expected to have new features and may even have entire new feature sets.

https://semver.org/

Semantic Versioning Challenges

- Over constrainted
- Over promising
- Engineering team objectives – it’s the social challenge again