High quality SBOMs for C/C++ and native code

Andreas Bielk

2024-09-24

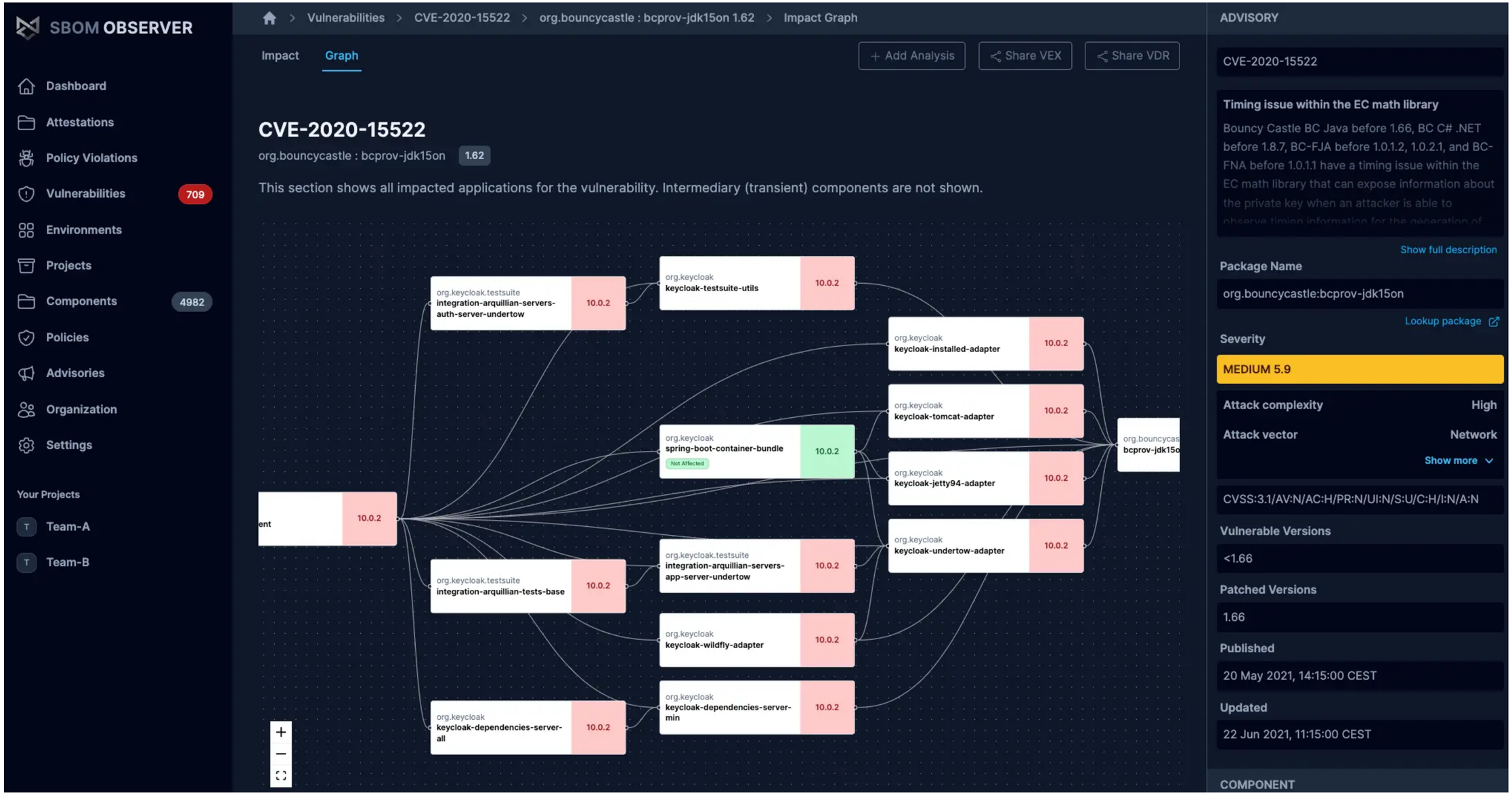
https://sbom.observer/

NORDIC SOFTWARE SECURITY SUMMIT

NSSS.SE

THE SOFTWARE SECURITY REGULATION REVOLUTION

By now, we all know what an SBOM is right?


SBOM OBSERVER
Dashboard
Attestations
Policy Violations
Vulnerabilities 709
Environments
Projects
Components 4982
Policies
Advisories
Organization
Settings
Your Projects
Team-A
Team-B
Vulnerabilities > CVE-2020-15522 > org.bouncycastle : bcprov-jdk15on 1.62 > Impact Graph
Impact
Graph
+ Add Analysis
Share VEX
Share VDR
CVE-2020-15522
org.bouncycastle : bcprov-jdk15on 1.62
This section shows all impacted applications for the vulnerability. Intermediary (transient) components are not shown.
org.keycloak.testsuite integration-arquillian-servers-auth-server-undertow 10.0.2
org.keycloak keycloak-testsuite-utils 10.0.2
org.keycloak keycloak-installed-adapter 10.0.2
org.keycloak keycloak-tomcat-adapter 10.0.2
org.keycloak spring-boot-container-bundle Not Affected 10.0.2
org.keycloak keycloak-jetty94-adapter 10.0.2
org.bouncycas bcprov-jdk15o
ent 10.0.2
org.keycloak.testsuite integration-arquillian-tests-base 10.0.2
org.keycloak.testsuite integration-arquillian-servers-app-server-undertow 10.0.2
org.keycloak keycloak-undertow-adapter 10.0.2
org.keycloak keycloak-wildfly-adapter 10.0.2
org.keycloak keycloak-dependencies-server-min 10.0.2
org.keycloak keycloak-dependencies-server-all 10.0.2
ADVISORY
CVE-2020-15522
Timing issue within the EC math library
Bouncy Castle BC Java before 1.66, BC C# .NET before 1.8.7, BC-FJA before 1.0.1.2, 1.0.2.1, and BC-FNA before 1.0.1.1 have a timing issue within the EC math library that can expose information about the private key when an attacker is able to
Show full description
Package Name
org.bouncycastle:bcprov-jdk15on
Lookup package
Severity
MEDIUM 5.9
Attack complexity High
Attack vector Network
Show more
CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:N/A:N
Vulnerable Versions
<1.66
Patched Versions
1.66
Published
20 May 2021, 14:15:00 CEST
Updated
22 Jun 2021, 11:15:00 CEST
COMPONENT

The Problem

Compiled languages like C/C++ without a standard package manager means that the tactics used to build SBOMs in other ecosystems doesn't really work

All tools that provide material to built artefacts are potential attack vectors!

The Challenge

- Minimal changes to the existing build system
- Reasonably exact - i.e. we include all dependencies*
- Support custom build configurations (i.e. optional modules)
- Document build tools used

Possible solutions

- Manual BOM, maybe with the help of some internal tooling
- Retrofit a dependency manager into the project (vcpkg, conan)
- Instrument the test-suite
- Instrument the compiler and linker

The Plan

- Watch all files the build process opens and executes to complete the build
- Map observed files to dependencies
- Create an SBOM
- Profit(!?)

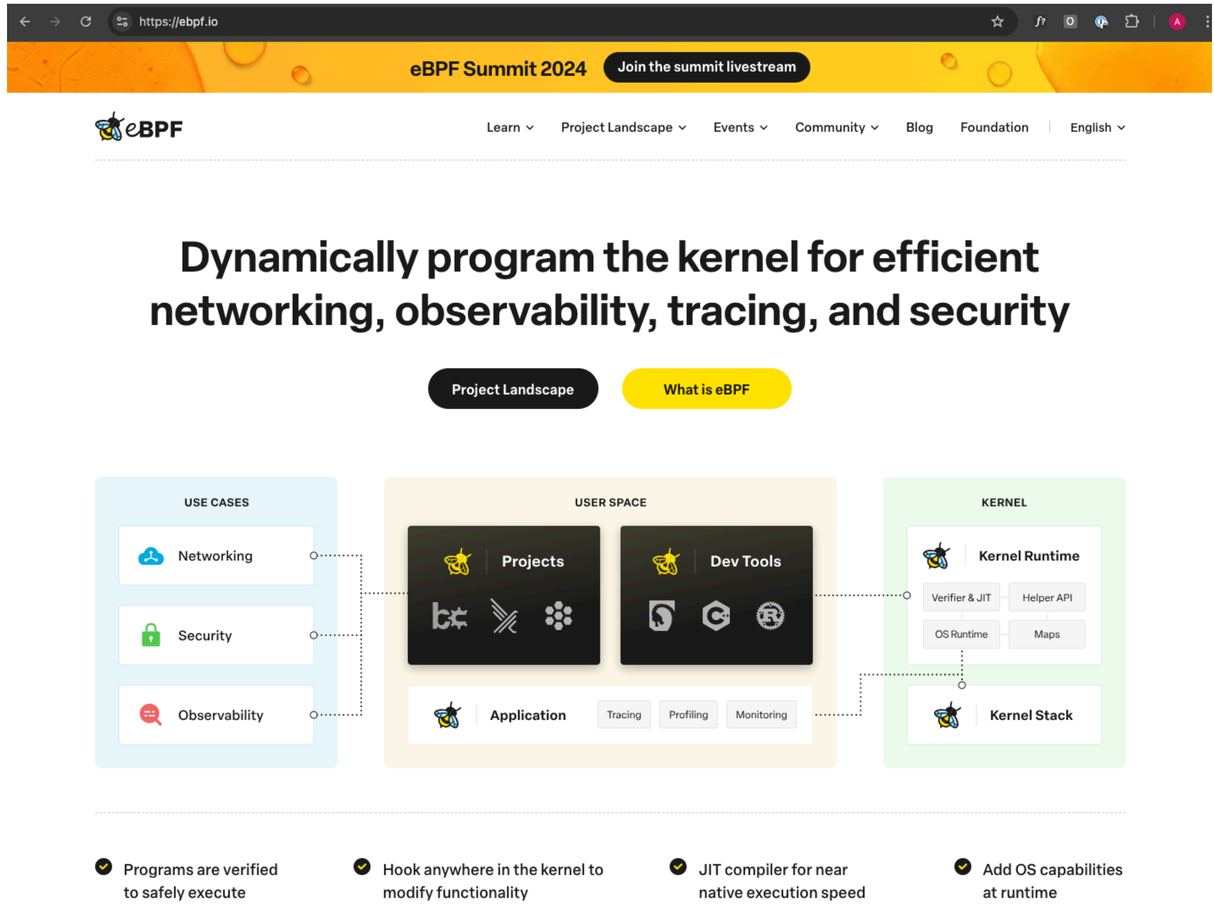
strace is slow

- Instrument process to observe all opened files (syscalls)
- Works, but very slow
- Asterisk build went from 5 min to 40 min

eBPF

"eBPF is a revolutionary technology with origins in the Linux kernel that can run sandboxed programs in a privileged context such as the operating system **kernel**.

It is used to **safely** and **efficiently** extend the capabilities of the kernel without requiring to change kernel source code or load kernel modules."



eBPF is fast

- Instrument kernel to observe all opened and executed files (syscalls)
- 10% build time overhead (~30 sec added for 5 min Asterisk build)

```
$ sudo build-observer --user vagrant -- make -f Makefile.linux build-examples
```

build-observer output

```
start 2024-09-21T16:07:45
exec /usr/bin/sudo
exec /usr/bin/make
....
open make Makefile
exec /usr/bin/gcc
...
exec /usr/lib/gcc/x86_64-linux-gnu/12/cc1
...
open cc1 menuselect.c
open cc1 /usr/include/stdc-predef.h
open cc1 /usr/include/stdlib.h
...
```

Creating an SBOM

- We now have a logfile of syscalls (opens, and executions)
- Use some heuristics to prune and deduplicate file dependencies
- Lookup which OS package each file belongs to
  - Optionally track transitive dependencies (depending on use case)
- Lookup any Git repo dependencies
- Create a dependency tree
- Output an SBOM

Library Dependency


```
{
  "bom-ref": "pkg:deb/debian/libcodec2-dev@1.0.5-1?arch=amd64&distro=debian-12.5",
  "type": "library",
  "name": "libcodec2-dev",
  "version": "1.0.5-1",
  "licenses": [
    {
      "license": {
        "id": "LGPL-2.1"
      }
    },
    {
      "license": {
        "id": "LGPL-2.1-or-later"
      }
    },
    {
      "license": {
        "id": "BSD-3-Clause"
      }
    }
  ],
  "purl": "pkg:deb/debian/libcodec2-dev@1.0.5-1?arch=amd64&distro=debian-12.5"
},
```

Compiler Dependency


```
{
  "bom-ref": "pkg:deb/debian/cpp-12@12.2.0-14?arch=amd64&distro=debian-12.5",
  "type": "application",
  "name": "cpp-12",
  "version": "12.2.0-14",
  "purl": "pkg:deb/debian/cpp-12@12.2.0-14?arch=amd64&distro=debian-12.5",
  "components": [
    {
      "type": "file",
      "name": "/usr/lib/gcc/x86_64-linux-gnu/12/cc1",
      "hashes": [
        {
          "alg": "SHA-256",
          "content": "a9d27bdb13cfcae82035677c84a28ad3b35fd5e6b3f5e19060d3ba1a69f3c3fe"
        }
      ]
    }
  ]
},
```

Mixed language applications

- Applications built from a mix of languages, like Javascript, or Python mixed with C/C++
- Make sure to observe the complete build, including installing dependencies
- Force recompilation of dependencies (you should already be doing this)
- Merge SBOM with dependencies for the other ecosystems
- Works surprisingly well!

Current Status

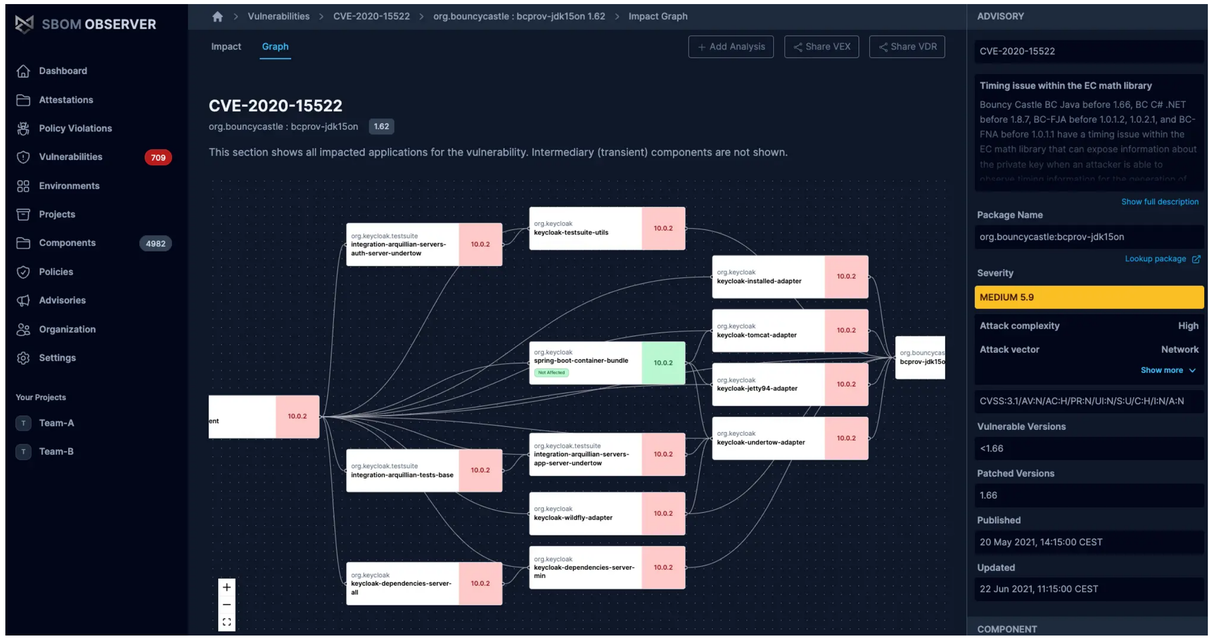
- Proof-of-concept
  - Debian build machines only
  - bpftrace is great for prototyping eBPF
- “Production ready” version later this fall (Open Source)
  - RPM based build machines
  - Support for Git dependencies
  - Support for internal dependencies
  - Log files suitable for archiving together with artefacts

Future work

- Support vendored dependencies
- Support Windows environments
- Generating Runtime SBOMs using eBPF
  - Dynamic/late linking introduces another supply chain
  - Runtimes (nodejs, python, php etc)
- Runtime SBOM conformance/enforcement
- Capture other assertions about build environment
  - Compiler flags
  - Security related build steps (SAST tools etc)

Resources

- Find me
  - andreas@sbom.observer
  - https://www.linkedin.com/in/andreas-bielk/
- Day job
  - https://sbom.observer/
  - Consulting
- Source Code
  - https://github.com/sbom-observer/build-observer
  - https://github.com/sbom-observer/observer-cli



BYTESAFE

SBOM OBSERVER