llvm-gitbom
Building Software Artifact Dependency Graphs for Vulnerability Detection

Bharathi Seshadri

Yongkui Han

Ed Warnicke

Nov 9, 2022

Agenda

- Overview of GitBOM
- Llvm-gitbom
- CVE Detection (PoC)
- Demo
- Summary and next steps

Have you heard of?

Heartbleed

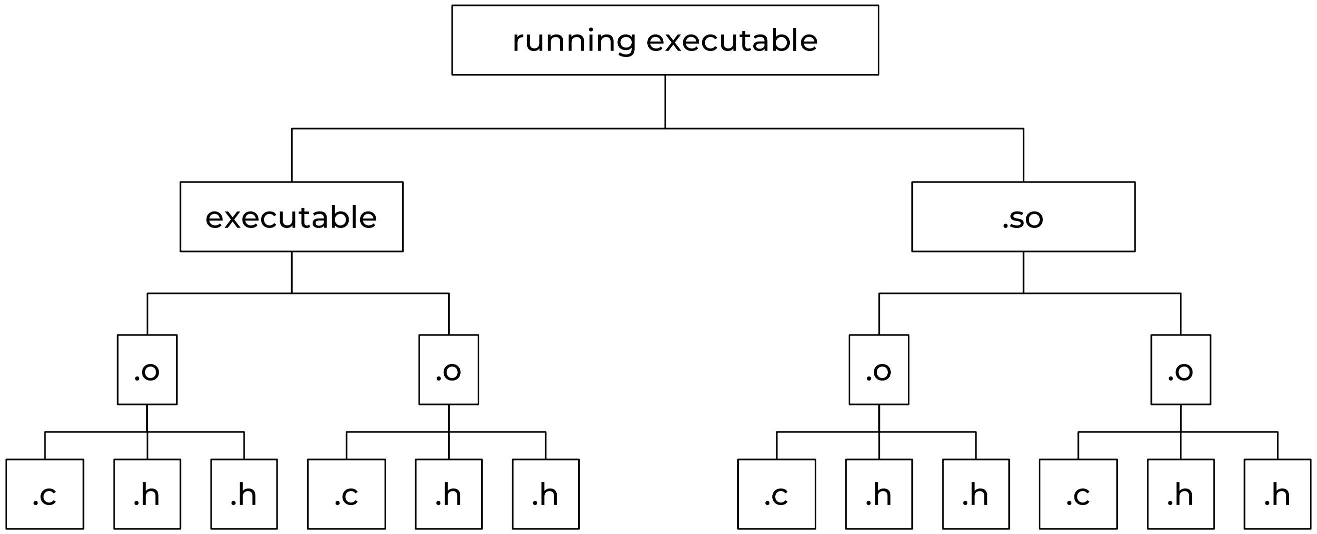
Log4Shell tm

Supply Chain Vulnerabilities: What’s baked in the pie?

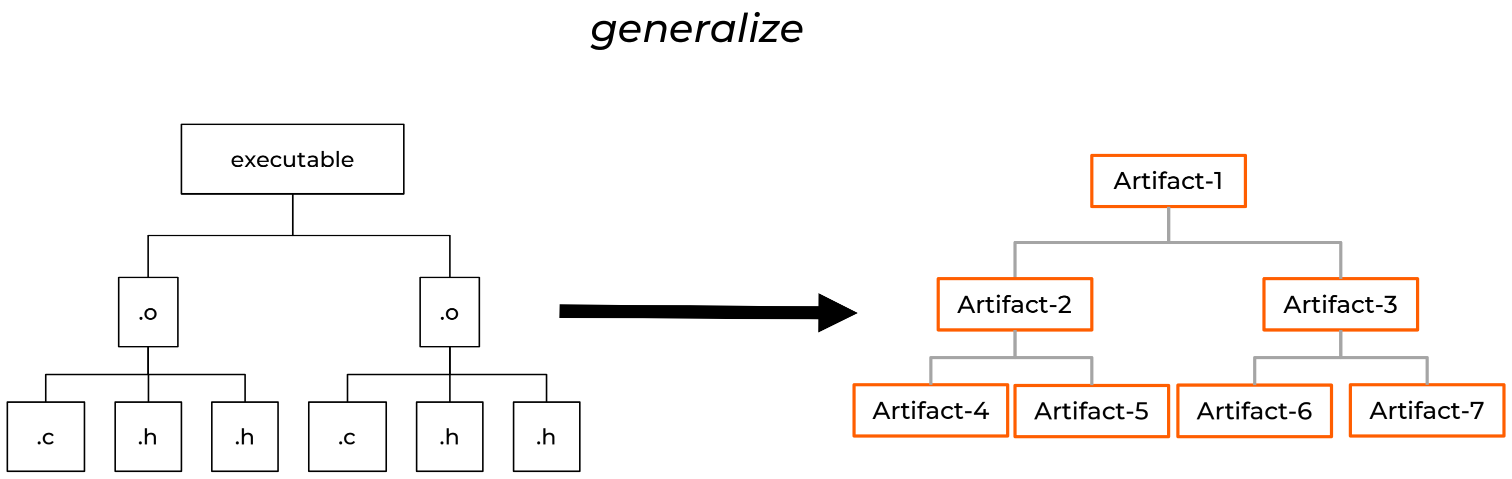
Scanners
- Many false positives
- Many false negatives

Build tools
- Know exact inputs

Ingredients: Artifact Dependency Graph

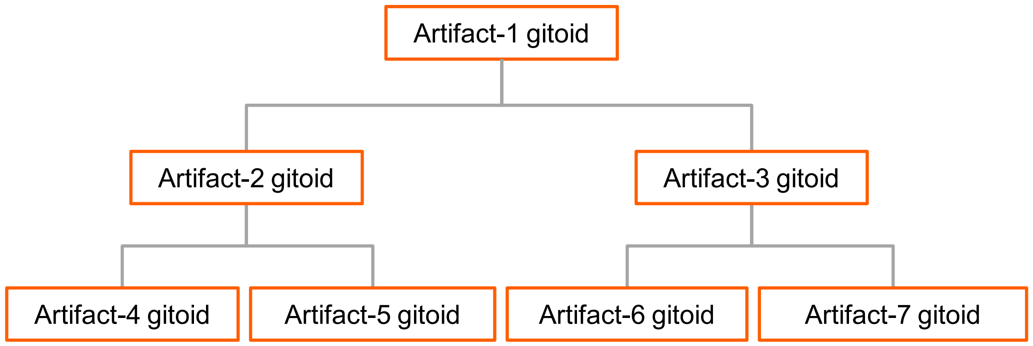


Artifact Dependency Graph: Generalize

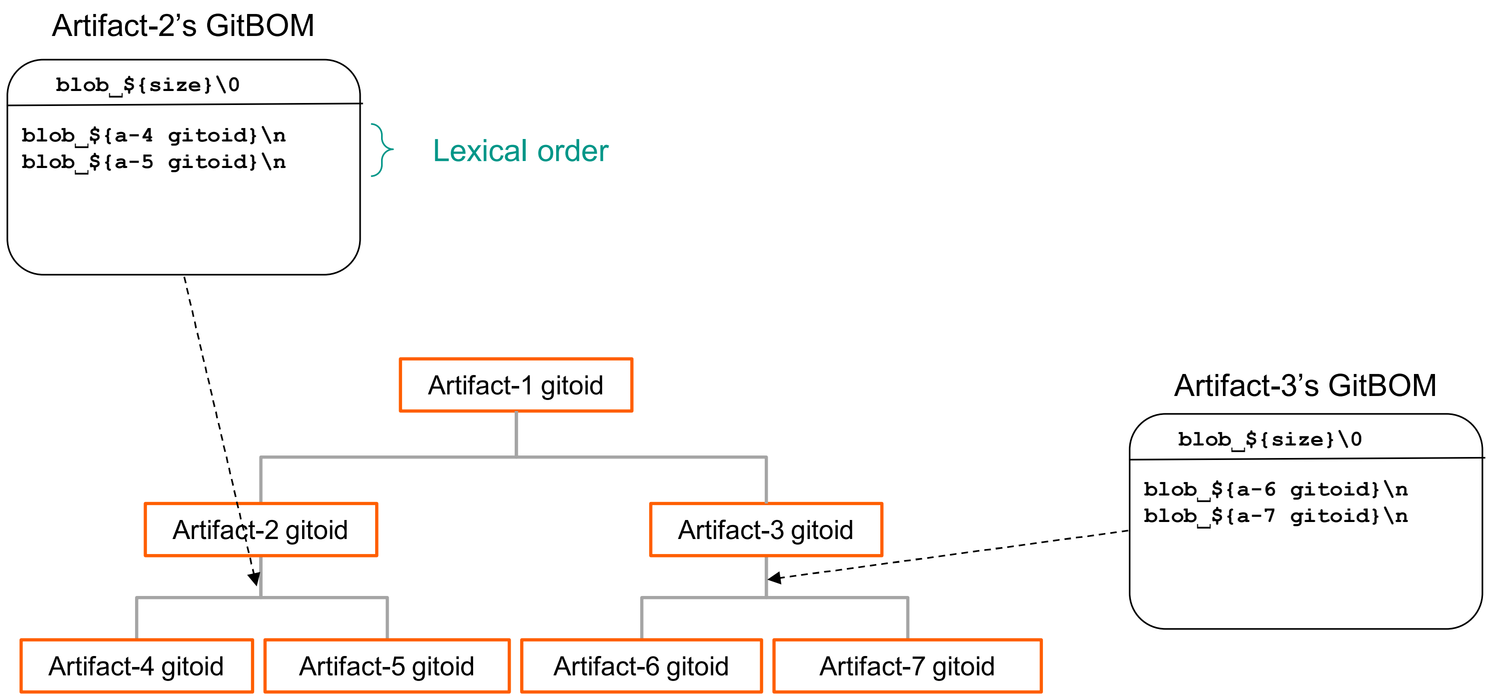


Artifact Dependency Graph: Artifact Identity

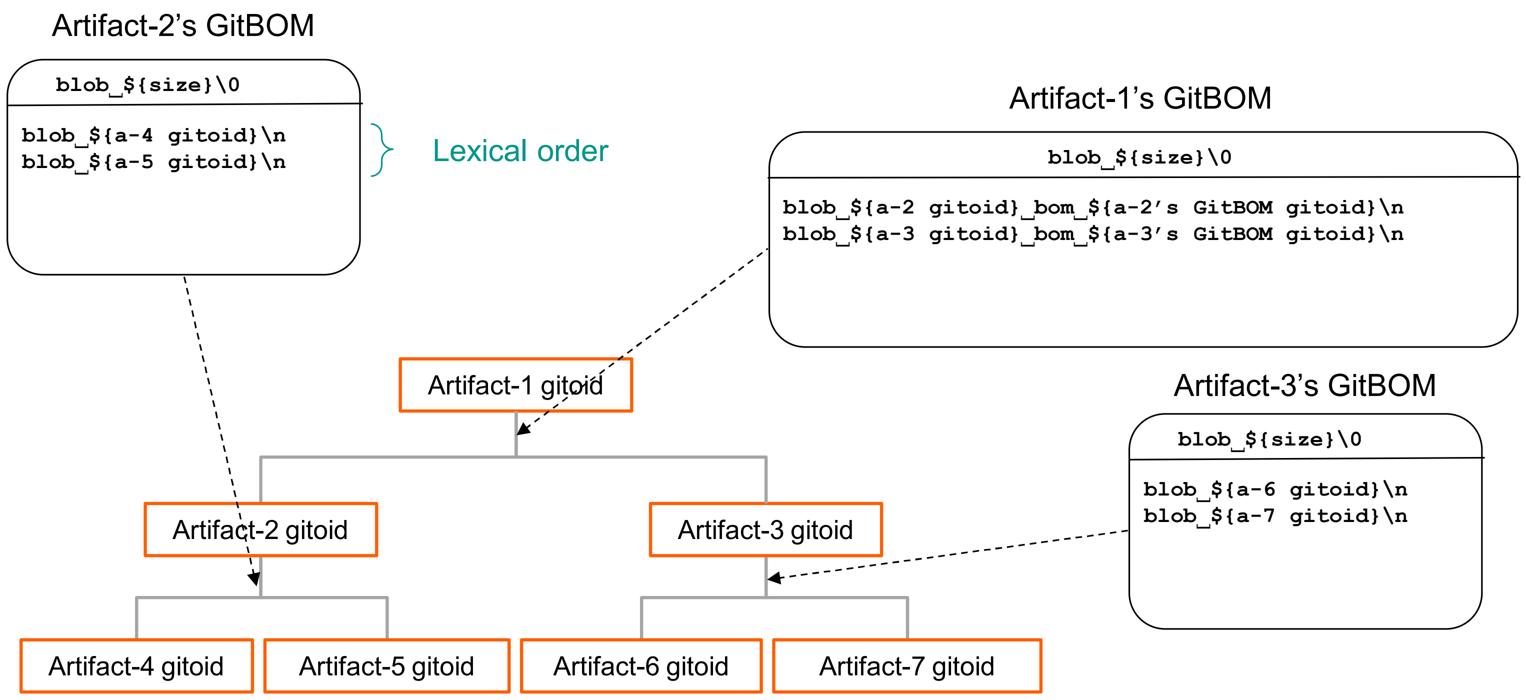
Use gitoids as artifact ids



Artifact Dependency Graph: Inputs



Artifact Dependency Graph: Inputs



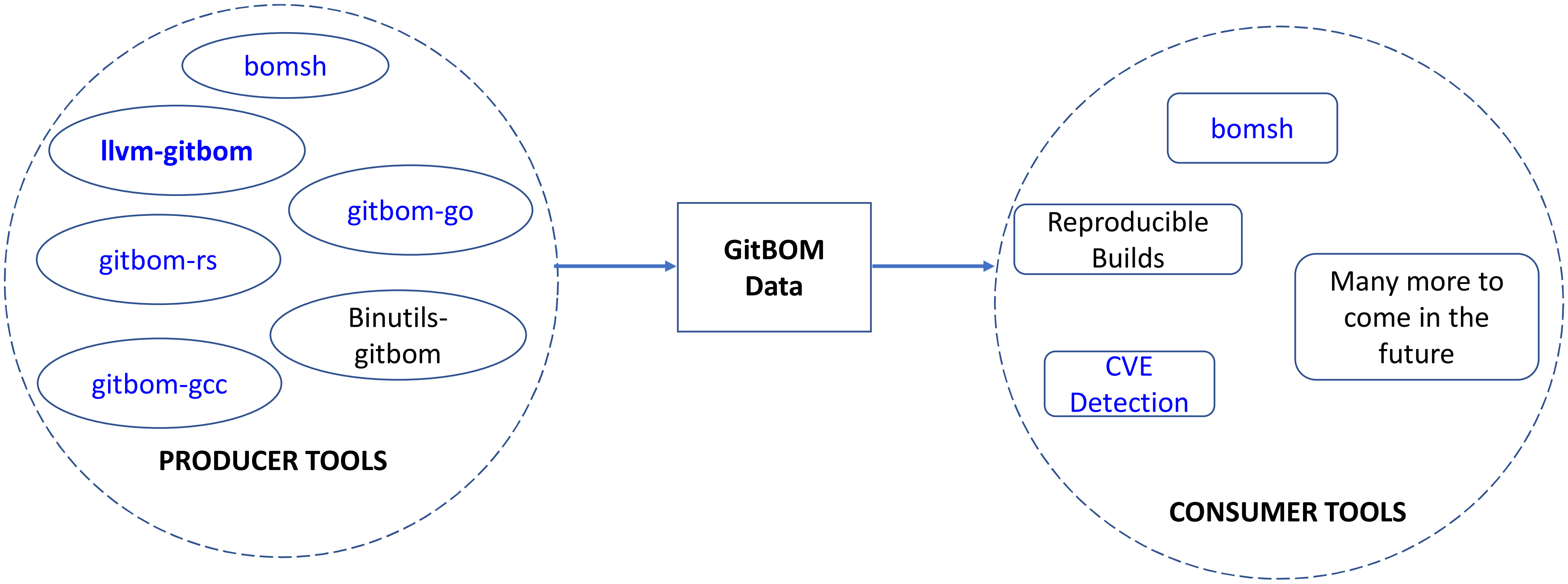
llvm-gitbom: Operation and Usage

Why llvm-gitbom?

- Build tools (/Compilers/Linkers) know what goes into an artifact
- Have the dependency information critical for implementing GitBOM
- Easy to embed GitBOM in the artifact

llvm-gitbom is an implementation of GitBOM in the LLVM compiler infrastructure

GitBOM: Tooling Infrastructure



Prototype Tooling available

Llvm-gitbom: Generate GitBOM data

- Prototype based on llvm-14.0
- Clang Compiler

  -frecord-gitbom, -frecord-gitbom=<gitbom_dir>

  env GITBOM_DIR=<gitbom_dir>
- Lld Linker

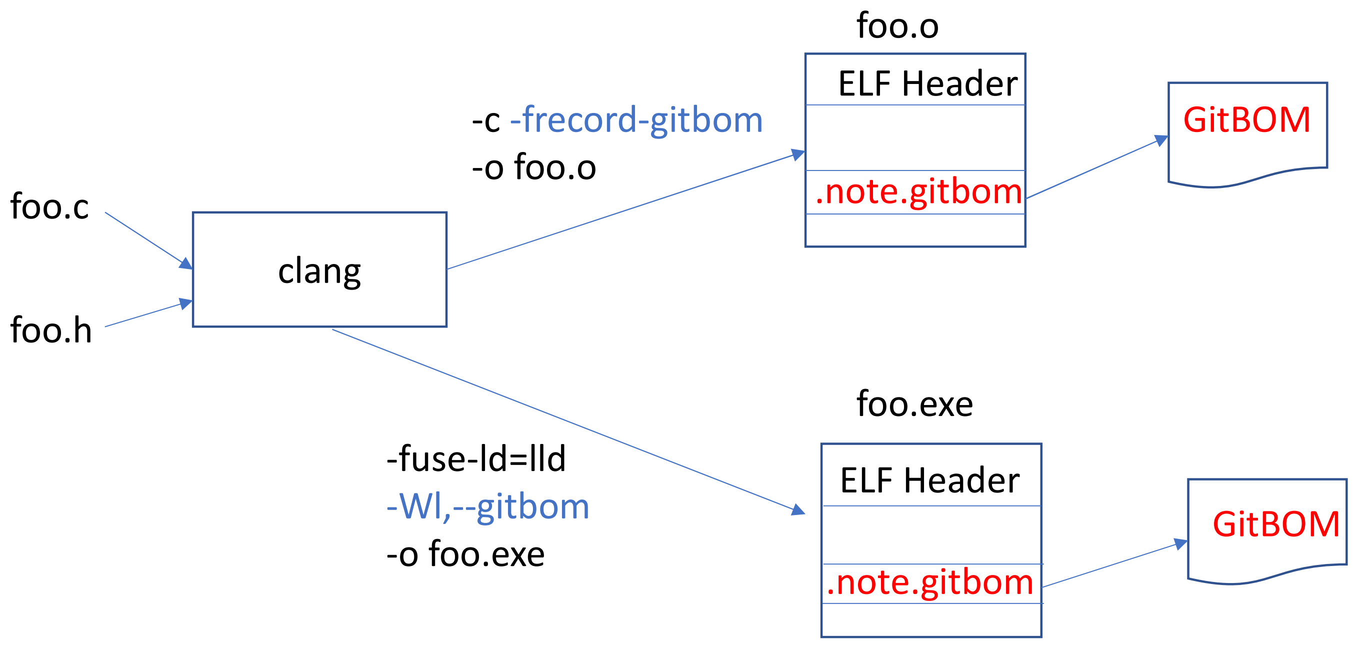
  lld option to generate gitbom information:

  --gitbom, --gitbom=<gitbom_dir>

  env GITBOM_DIR=<gitbom_dir>

  Currently supports ‘C Language’ and ELF format

llvm-gitbom


foo.c
foo.h
clang
-c -frecord-gitbom
-o foo.o
foo.o
ELF Header
.note.gitbom
GitBOM
-fuse-ld=lld
-Wl,--gitbom
-o foo.exe
foo.exe
ELF Header
.note.gitbom
GitBOM

GitBOM document

- Describes the immediate children of an artifact in the ADG
- Leaf artifact:

blob␣ ${artifact id of the child}\n

- Child artifact:

blob␣${artifact id of child}␣bom␣${GitBOM Id of child's GitBOM Document}\n

- How is it computed ?
  1. Collect all the dependencies (.h, .c, .o, .so, linker script)
  2. Record gitoid of the dependencies in lexicographic order
  3. Compute the GitBOM Id (gitoid of the contents in step #2)
  4. Name the GitBOM document as ${GitBOMId:0:2}/${GitBOMId:2:}
  5. sha1 & sha256 supported

GitBOM document (Example)

```
$ cat .gitbom/objects/sha1/64/7ef46ced31ef86c0a8dbcd1e43cceed0d62ed8
gitoid:blob:sha1
blob bfb4feb0a12d6226a33c44138b6d0bd7505167e1
blob c0b1bf12ffd95ee2b70a0cfe8ed955290003fe38 bom dca3131eb50e099856c0fbf361dfe132066cf1e7
```

Embedding GitBOM identifier

- Embed GitBOM identifier into the artifact
- .note.gitbom section
- Type: SHT_NOTE; Attribute: SHF_ALLOC
- Supported hash types by *git* (sha1, sha256)
- One Note entry per hash type

Type	NT_GITBOM
Name size	7
Name (Owner)	GITBOM
Descriptor size	Length of gitoid
Descriptor	Gitoid

.note.gitbom

```
$ llvm-readelf -n vmlinux

....
  GITBOM              0x00000014       NT_GITBOM (SHA1 GITOID)
   SHA1 GitOID: dbe86614f17d7846d24549370c2d794a7cb280c4
  GITBOM              0x00000020       NT_GITBOM (SHA256 GITOID)
   SHA256 GitOID: 79caa61277c6374e4b74facaeb87af4a28a031f3f3ff2823aa220966c2a1f469
  ....
```

Expected change in binary size:

+92 bytes for .note.gitbom

+32 bytes section header

+/- padding adjustments for alignment

llvm-gitbom: Benchmarking

Very low overhead for build time and code size

OpenSSL (libcrypto.so, libssl.so)

- Openssl version 3.0.7 built on Ubuntu 20.04.1 with -j8

Parameter	GitBOM Enabled Build
Build Time	**+6%** (< 3 s)
Size of Build Dir	**+0.2%** (~652Kb out of 332M)
Size of shared lib (crypto, ssl)	**+(0.001%, 0.03%)**
Size of GitBOM Docs	29M (sha1: 12.5M, sha256: 16.5M)
Compressed Size of GitBOM Docs	1.6M
# of GitBOM Docs	3152 (sha1: 1576, sha256: 1576)

Note: Only production builds need to be gitbom enabled.

Linux Kernel

- Linux kernel version 6.0.2 built on Ubuntu 20.04.1 with -j8

Parameter	GitBOM Enabled Build
Build Time	**+4%** (< 2 m)
Size of Build Dir	**+0.03%** (< 5MB out of 1.7G)
Size of vmlinux	**Negligible** (+64b out of 590M)
Size of GitBOM Docs	**1.6G** (sha1: 646M, sha256: 957M)
Compressed Size of GitBOM Docs	600M
# of GitBOM Docs	~60K (sha1: 30K, sha256: 30K)

Note: Only production builds need to be gitbom enabled.

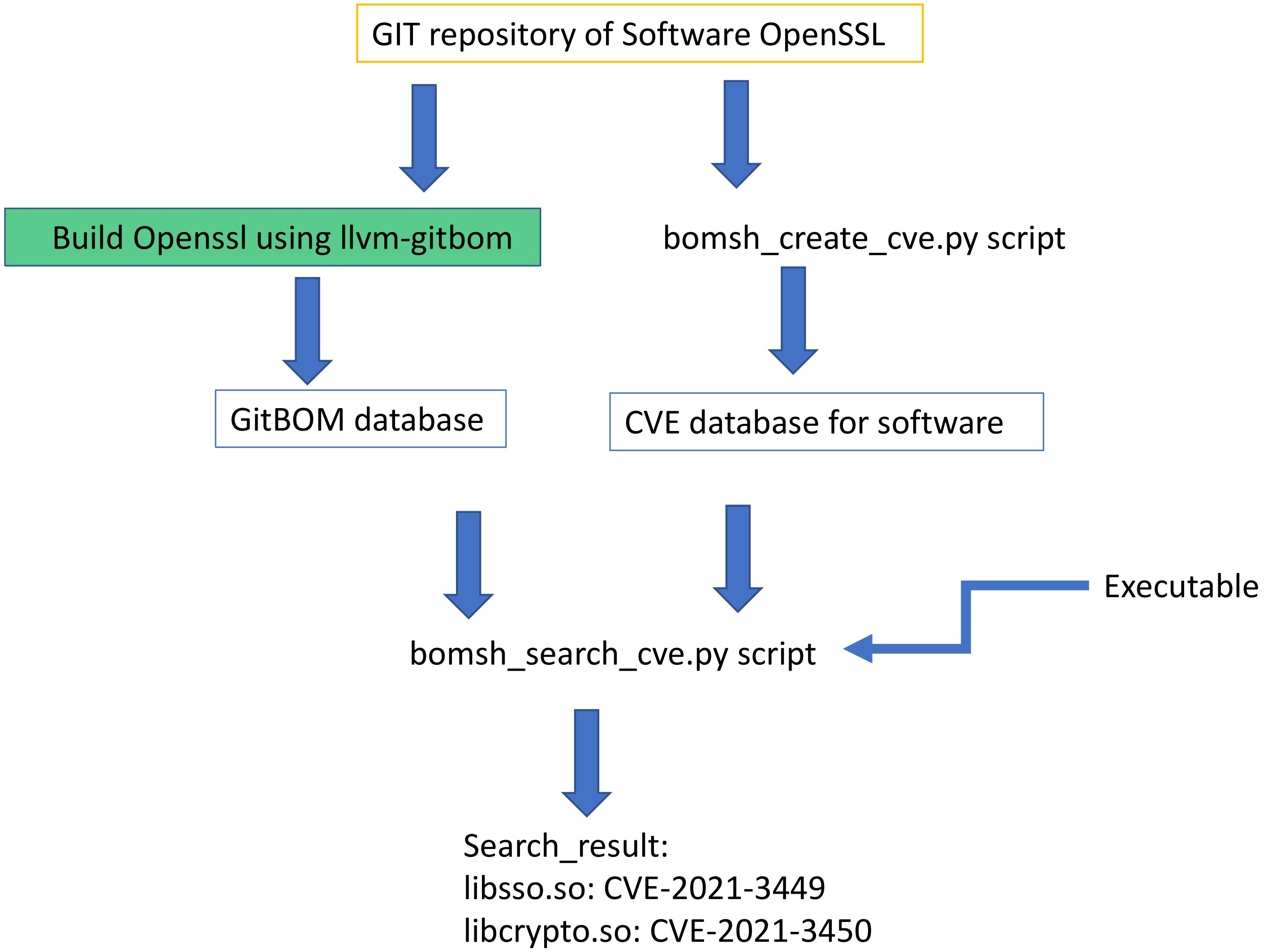
llvm-gitbom: Application to CVE Detection

Work by Yongkui Han

CVE Detection using GitBOM (PoC)

- GitBOM tells us what constitutes an artifact
- List of artifact ids can be inferred from GitBOM
- CVE is associated with source files
- Generate a database recording all CVEs
- Compare against the DB to find CVE in any binary

CVE Detection Framework Overview


GIT repository of Software OpenSSL
Build Openssl using llvm-gitbom
bomsh_create_cve.py script
GitBOM database
CVE database for software
Executable
bomsh_search_cve.py script
Search_result:
libsso.so: CVE-2021-3449
libcrypto.so: CVE-2021-3450

How to create an accurate CVE Database

- The goal is to create a database for all CVE-relevant source file blobs.
- All artifact IDs are stored in git repo.
- All artifact IDs must be classified as CVE-vulnerable or not based on some criteria.
- Git commits can be used to do the CVE classification (just a proposal).
  - CVE-add and CVE-fix commits
  - CVE checking rules
- One time effort

(Discussion with MITRE to add gitoid to CVE info)

OpenSSL CVE-info Repository

- An example CVE-info repo for OpenSSL is here:
  - https://github.com/yonhan3/openssl-cve
- It contains the CVE-info for 7 high-severity CVEs
  - The CVE-add/CVE-fix commits
  - The CVE-checking rules
  - CVE-2014-0160
  - CVE-2020-1967
  - CVE-2020-1971
  - CVE-2021-3449
  - CVE-2021-3450
  - CVE-2021-3711
  - CVE-2022-0778

Sample Tag info to track CVE commits

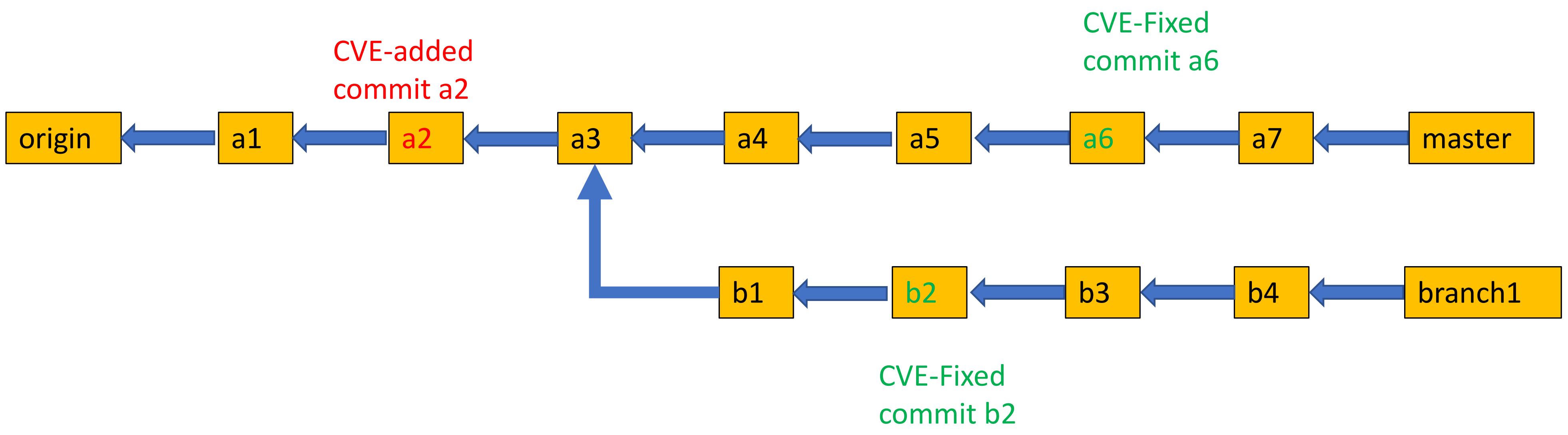
```
$ cat cveinfo.5235ef4.yaml
Added:
 CVE-2020-1967:
  src_files:
   - ssl/t1_lib.c

 $ cat cveinfo.a87f3fe.yaml
Fixed:
 CVE-2020-1967:
  src_files:
   - ssl/t1_lib.c
```

Compilation of CVE info

```
"CVE-2020-1967": {
       "Added": [
           {
               "commit": "5235ef4",
               "src_files": [
                   "ssl/t1_lib.c"
               ]
           }
       ],
       "Fixed": [
           {
               "commit": "a87f3fe",
               "src_files": [
                   "ssl/t1_lib.c"
               ]
           },
           {
               "commit": "eb56324",
               "src_files": [
                   "ssl/t1_lib.c"
               ]
           }
       ]
   },
```

Common scenario for CVE commits in OpenSSL



OpenSSL: CVE Search

Version	Open CVE	Fixed CVE	Open CVE	Fixed CVE
	Libcrypto		libssl	
3.0.0	CVE-2022-0778	CVE-2021-3711	CVE-2022-0778	CVE-2014-0160, CVE-2020-1967, CVE-2021-3711, CVE-2021-3449,
3.0.1	CVE-2022-0778	CVE-2021-3711	CVE-2022-0778	CVE-2014-0160, CVE-2020-1967, CVE-2021-3711, CVE-2021-3449,
3.0.2		CVE-2022-0778, CVE-2021-3711		CVE-2014-0160, CVE-2020-1967, CVE-2021-3711, CVE-2021-3449,
3.0.3 - 3.0.6		CVE-2022-0778, CVE-2021-3711		CVE-2014-0160, CVE-2020-1967, CVE-2021-3711, CVE-2021-3449, CVE-2022-0778

OpenSSL: CVE Search

Version	Open CVE	Fixed CVE	Open CVE	Fixed CVE
	Libcrypto		libssl	
9-Nov-2018	CVE-2022-0778, CVE-2021-3711,		CVE-2022-0778, CVE-2021-3711, CVE-2021-3449, CVE-2020-1967,	CVE-2014-0160,
9-Nov-2019	CVE-2022-0778, CVE-2021-3711,		CVE-2022-0778, CVE-2021-3711, CVE-2021-3449, CVE-2020-1967,	CVE-2014-0160,
9-Nov-2020	CVE-2022-0778, CVE-2021-3711		CVE-2022-0778, CVE-2021-3449, CVE-2021-3711,	CVE-2014-0160, CVE-2020-1967,
9-Nov-2021	CVE-2022-0778,	CVE-2021-3711	CVE-2022-0778	CVE-2014-0160, CVE-2020-1967, CVE-2021-3711, CVE-2021-3449,
Oct-2022		CVE-2022-0778, CVE-2021-3711		CVE-2022-0778 CVE-2014-0160, CVE-2020-1967, CVE-2021-3711, CVE-2021-3449,

llvm-gitbom: Demo

- Usage
- llvm-gitbom for openssl builds
- CVE detection for open-ssl

- Recorded Demo video comes here

Summary and Next Steps

- llvm-gitbom: clang and lld prototypes available
- Apply llvm-gitbom for CVE detection
- Prototyping to keep pace with evolving GitBOM spec
- Identify useful metadata to capture
- Prototype more applications
- Upstream plans
- Welcome participation from the llvm-community
- GitBOM → New name coming up!

Get Involved!

https://gitbom.dev/community/

Thank you!