
SCANOSS
CODE FREELY. FINISH EARLY. TRUST COMPLETELY.

IMAGINE A WORLD WHERE...

SCANOSS

- SCA tools are 100% Open Source

No tricks. Just plain OSS

- SBOMs are standard (comparable, mergeable, diffable)

Imagine usable interaction across tools

- OSS Inventorying is a standardised industry process

Always-on SBOM
+ revision history

- M&As due diligence includes SBOM Validation (no auditing)

Imagine cutting over 90% of OSS due diligence costs
Auditing → Validation

SBOM M&A, mergeable into new dev environment

WHO ARE WE?

SCANOSS

- First 100% Open Source SCA tool
- Designed for DevOps environments
- Committed to standardise the OSS Inventorying process
- Contributed our entire SCA platform

https://github.com/scanoss

Sadly, none of the existing SCA vendors have embraced Open Source themselves. This has now changed!

SCA tools were born to cover the need for OSS auditing during M&As. During long years, developers struggled to adapt these tools to their environments

Beautiful architecture. Innovation in many areas.

We strongly believe OpenChain should be the core forum since this is your main focus, plus you have already formed a specific tools group

HOW DO WE MAKE MONEY?

SCANOSS

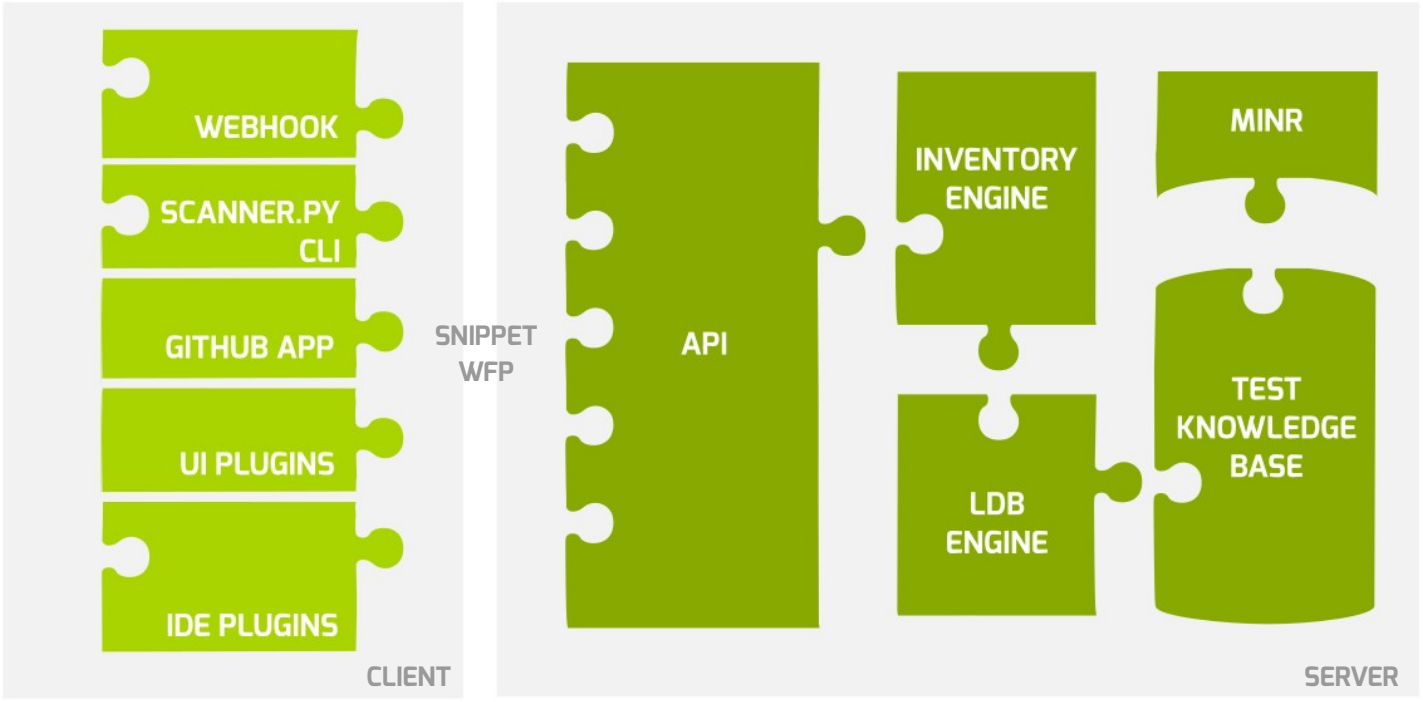
- Not by creating vendor dependencies
- Subscription to our big KB
- Additional Value Added Services Like
  - Data aggregation
  - Improved analysis result accuracy
  - UI, dashboarding
- Commercial SLAs

Creating vendor dependencies is exactly what we DONT want. This is why we are OSS

Post scan intelligence. We improve match results and provide data aggregation (vulnerabilities, risk, cryptography for Export Control, etc)

OPEN SOURCE PLATFORM

SCANOSS



This is a high level overview of our platform to illustrate what is it that we have contributed in a single view.

Client side applications calculate **snippet** fingerprints (we call them wfp, for winnowing fingerprints, which is an open source algorithm we also released)

These wfps are send to the API via https.

The API passes the request to the inventory engine, which is the core application. The inventory engine performs a comparison against the KB

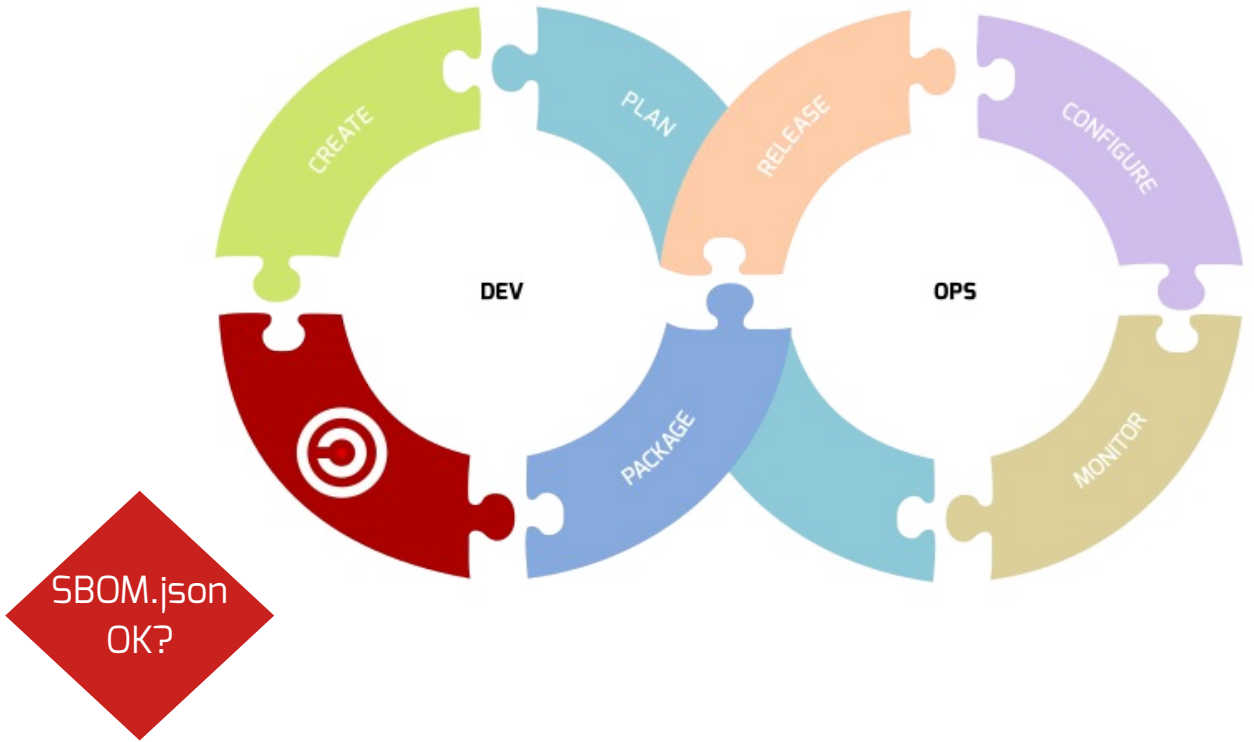
The Knowledge Base is stored using our LDB database (open source) which stands for “Linked list DB”. It is a mapped linked list database that allows storage of massive amounts of data (2T records) and query responses in microseconds.

We provide a test Knowledge Base in Github, as well as minr, which allows you to create your own KB

This is all in Github already

ALWAYS-ON SBOM

SCANOSS



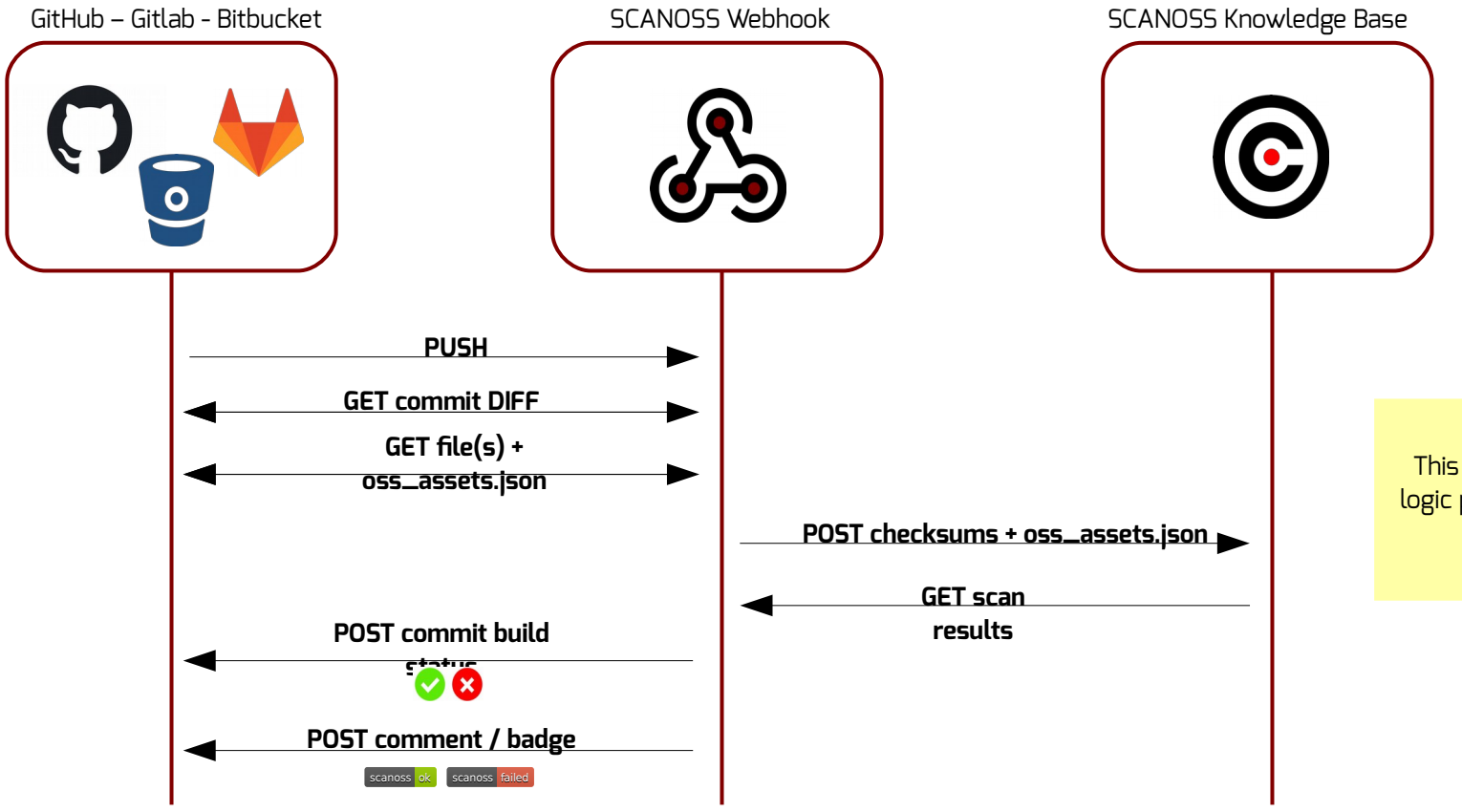
SBOM.json is an SPDX or CycloneDX (in their JSON format) file that declares OSS components included and implemented

As part of the validation process in a DevOps environment, commits are validated against the SCANOSS KB and the SBOM.json

If undeclared matches are found, the commit is flagged by setting the build status to failed and adding the details

The SBOM is not only always on, but it could also be pulled for any past commit too

SCANOSS Webhook – CI



This diagram zooms into the logic presented in the previous slide

SCANOSS AT WORK: MINR

SCANOSS

```
$ minr -d scanoss,webhook,1.0 -u https://github.com/scanoss/webhook/archive/1.0.tar.gz
Downloading https://github.com/scanoss/webhook/archive/1.0.tar.gz
$
```

Minr mining a URL
Leaves captured, mergeable metadata in mined/

```
$ minr -z mined
mined/sources/146a.mz
...
$
```

Minr extracting snippet wfp from captured sources

```
$ minr -i mined/
$
```

Minr importing mined/ data into the LDB

SCANOSS ENGINE AGAINST FULL COMPONENT

SCANOSS

```
$ scanoss 1.0.tar.gz
{
  "1.0.tar.gz": [
    {
      "id": "component",
      "elapsed": "0.001139s",
      "lines": "all",
      "oss_lines": "all",
      "matched": "100%",
      "vendor": "scanoss",
      "component": "webhook",
      "version": "1.0",
      "latest": "1.0",
      "url": "https://github.com/scanoss/webhook/archive/1.0.tar.gz",
      "file": "all",
      "size": "N/A",
      "dependencies": [],
      "licenses": []
    }
  ]
}
```

SCANOSS engine
Scanning an entire component

SCANOSS ENGINE AGAINST FULL FILE

SCANOSS

```
$ scanoss webhook-1.0/scanoss/github.py
{
  "webhook-1.0/scanoss/github.py": [
    {
      "id": "file",
      "elapsed": "0.000395s",
      "lines": "all",
      "oss_lines": "all",
      "matched": "100%",
      "vendor": "scanoss",
      "component": "webhook",
      "version": "1.0",
      "latest": "1.0",
      "url": "https://github.com/scanoss/webhook/archive/1.0.tar.gz",
      "file": "webhook-1.0/scanoss/github.py",
      "size": "6624",
      "dependencies": [],
      "licenses": []
    }
  ]
}
$
```

SCANOSS engine
Scanning an entire file

SCANOSS ENGINE AGAINST FULL SNIPPETS

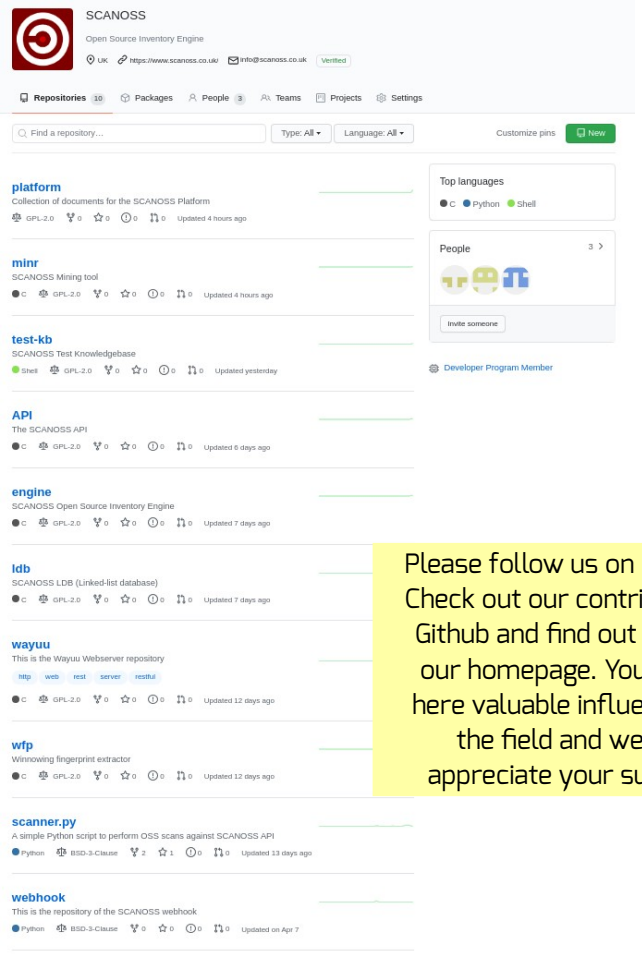
SCANOSS

```
$ scanoss webhook-1.0/scanoss/github.py
{
  "webhook-1.0/scanoss/github.py": [
    {
      "id": "snippet",
      "elapsed": "0.001812s",
      "lines": "1-192",
      "oss_lines": "3-194",
      "matched": "99%",
      "vendor": "scanoss",
      "component": "webhook",
      "version": "1.0",
      "latest": "1.0",
      "url": "https://github.com/scanoss/webhook/archive/1.0.tar.gz",
      "file": "webhook-1.0/scanoss/github.py",
      "size": "6624",
      "dependencies": [],
      "licenses": []
    }
  ]
}
```

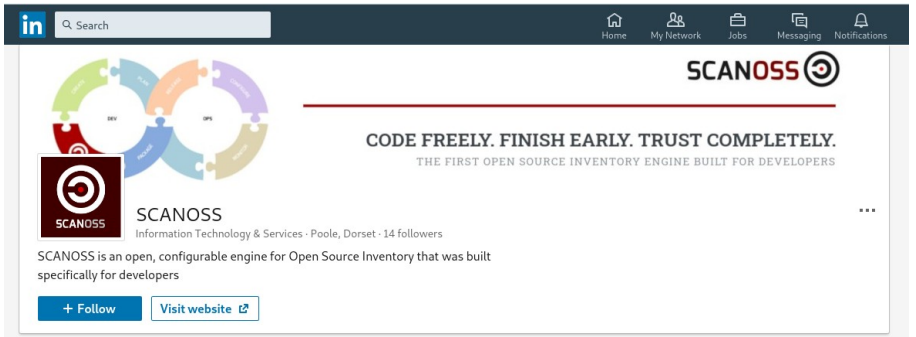
SCANOSS engine
Scanning an modify file

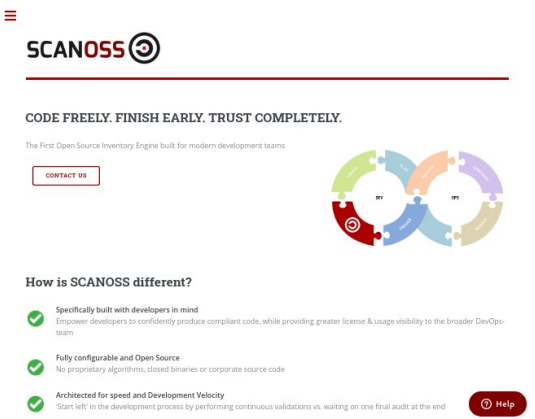
FOLLOW US, FORK, SHARE!

SCANOSS



Please follow us on Linkedin. Check out our contribution in Github and find out more in our homepage. You are all here valuable influencers in the field and we will appreciate your support!





http://scanoss.com

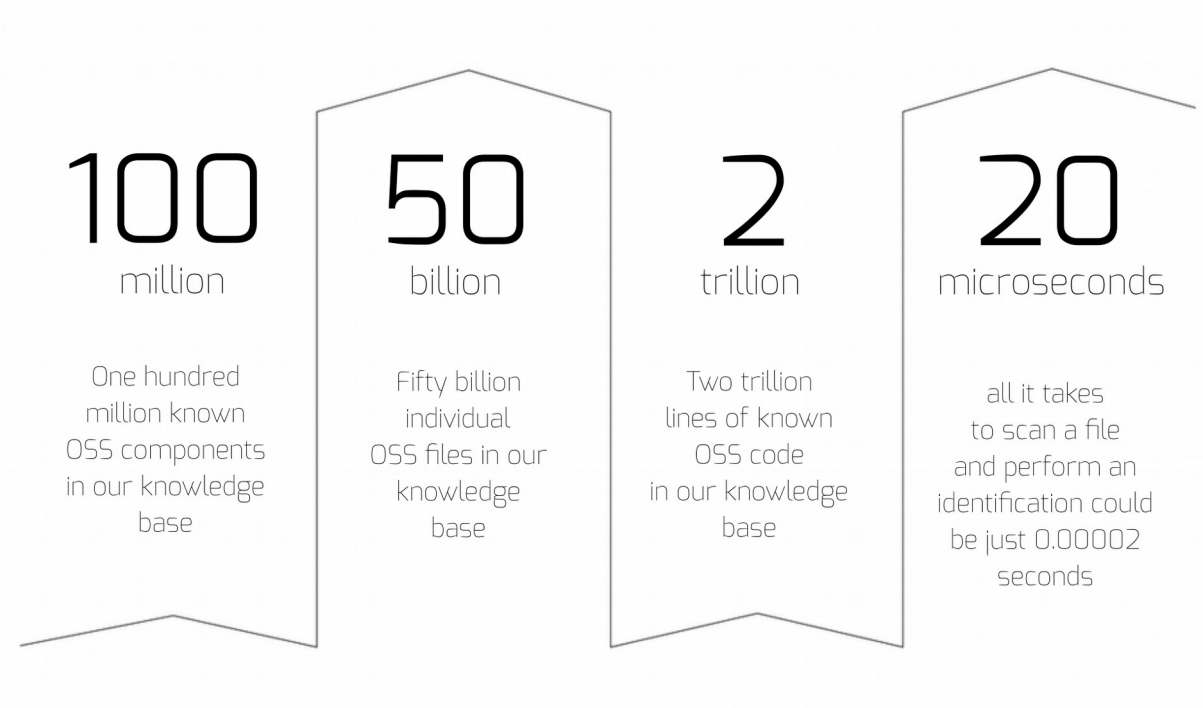
Q&As

SCANOSS

- Interaction with FOSSology
- Interaction with SW360


SCANOSS
CODE FREELY. FINISH EARLY. TRUST COMPLETELY.

SCANOSS


100
million
One hundred million known OSS components in our knowledge base
50
billion
Fifty billion individual OSS files in our knowledge base
2
trillion
Two trillion lines of known OSS code in our knowledge base
20
microseconds
all it takes to scan a file and perform an identification could be just 0.00002 seconds

Fingerprinting: Winnowing*

SCANOSS

test.c

```
uint32_t winnowing (char *src, uint32_t *hashes, uint32_t *lines, uint32_t limit)
{
    uint32_t line = 1;
    uint32_t counter = 0;
    uint32_t hash = MAX_UINT32;
    uint32_t last = 0;
    uint8_t *gram = malloc (GRAM);
    uint32_t gram_ptr = 0;
    uint32_t *window = malloc (WINDOW * sizeof(uint32_t));
    uint32_t window_ptr = 0;
    uint32_t src_len = strlen(src);

    for (uint32_t i = 0; i < src_len; i++)
    {
        if (src[i] == '\n') line++;

        uint8_t byte = normalize(src[i]);
        if (!byte) continue;
        gram[gram_ptr++] = byte;

        if (gram_ptr >= GRAM)
        {
            window[window_ptr++] = calc_crc32c((char *) gram, GRAM);

            if (window_ptr >= WINDOW)
            {
                hash = smaller_hash(window);
                last = add_hash(hash, line, hashes, lines, last, &counter);

                if (counter >= limit) break;

                shift_window(window);
                window_ptr = WINDOW - 1;
            }
            shift_gram(gram);
            gram_ptr = GRAM - 1;
        }
    }

    free (gram);
    free (window);
    return counter;
}
```

normalized

```
uint32twinnowingcharsrcuint32thashesuint32t
linesuint32tlimituint32tline1uint32tcounter
0uint32thashmaxuint32uint32tlast0uint8tgram
mallocgramuint32tgramptr0uint32twindowmallo
cwindowsizeofuint32tuint32twindowptr0uint32
tsrclenstrlensrcforuint32ti0isrcleniifsrcin
lineuint8tbytenormalizesrciifbytecontinuegr
amgramptrbyteifgramptrgramwindowwindowptrca
lccrc32cchargramgramifwindowptrwindowhashsm
allerhashwindowlastaddhashhashlinehasheslin
eslastcounterifcounterlimitbreakshiftwindow
windowwindowptrwindow1shiftgramgramgramptrg
ramfreegramfreewindowreturncounter
```

test.wfp

```
component=8d9ca79ba2dcc8477f826d9e95d24ff6,N/A,N/A,N/A,test.c
file=8d9ca79ba2dcc8477f826d9e95d24ff6,1131,test.c
5=a7c5df62
9=30265914,401bc5d2,2cd4d08f
10=c5ce0edc
17=0e7e07e9,a00e735a
23=f505b7c8,ce9a8b46
27=36317243,2a326c59
32=a35690cf,408051a6
33=a5a355a7
35=242c564f
36=77477f2a,bd13ac71
41=a212e995
```

Source: https://theory.stanford.edu/~aiken/publications/papers/sigmod03.pdf

Example scan using scanner.py

SCANOSS

```
> % python scanner.py ../demo/demo1
Scanning directory: ../demo/demo1
Saved generated WFP File in 'scan_wfp'
{
    "../demo/demo1/try_modified.c": [
        {
            "id": "snippet",
            "elapsed": "0.246385s",
            "lines": "10-80",
            "oss_lines": "4-74",
            "matched": "87%",
            "vendor": "pigz",
            "component": "pigz",
            "version": "2.3.4.orig",
            "latest": "2.3.4.orig",
            "url": "http://ftp.debian.org/debian/pool/main/p/pigz/pigz_2.3.4.orig.tar.gz",
            "file": "pigz-2.3.4/try.c",
            "size": "653"
        }
    ],
    "../demo/demo1/pigz_try.c": [
        {
            "id": "file",
            "elapsed": "0.000282s",
            "lines": "all",
            "oss_lines": "all",
            "matched": "100%",
            "vendor": "pigz",
            "component": "pigz",
            "version": "2.3.4.orig",
            "latest": "2.3.4.orig",
            "url": "http://ftp.debian.org/debian/pool/main/p/pigz/pigz_2.3.4.orig.tar.gz",
            "file": "pigz-2.3.4/try.c",
            "size": "653"
        }
    ],
    "../demo/demo1/zlib_crc32.c": [
        {
            "id": "file",
            "elapsed": "0.000586s",
            "lines": "all",
            "oss_lines": "all",
            "matched": "100%",
            "vendor": "node-js.mirror",
            "component": "zlib",
            "version": "6.16.0",
            "latest": "6.16.0",
            "url": "https://sourceforge.net/projects/node-js.mirror/files/v6.16.0/2018-12-26%2C%20Version%206.16.0%20_Boron_%20%28LTS%29%2C%20%40MylesBorins.tar.gz",
            "file": "nodejs-node-02a9eb8/deps/zlib/crc32.c",
            "size": "4053"
        }
    ]
}
```