FOSSology SCA Integration

Presenters:
Gaurav Mishra <gmishx@gmishx.in>
Shaheem Azmal M MD <shaheem.azmal@gmail.com>
Anupam Ghosh <anupamghosh.ind@gmail.com>
FOSSology – Linux Foundation Collaboration Project

www.fossology.org

- 2008 initial publication by HP
- 2015 Linux Foundation Collaboration Project
- It is a Linux Application
- Different tasks for OSS license compliance
  - Scanning for licenses
  - Copyright, authorship, e-mails
  - ECC statements
  - Generation of documentation
  - Export and import SPDX files
Features: Two License Scanners: Nomos and Monk

<table>
<thead>
<tr>
<th>Nomos Keywords</th>
<th>Nomos Reg. Expressions</th>
<th>Bulk Phrase Matches</th>
<th>Monk Full text Matches</th>
</tr>
</thead>
</table>
| • Finds all kind of license relevant texts  
• Finds unknown Licenses | • Finds most license relevant texts  
• Identifies also derivatives of licenses | • Good for finding actual licenses  
• Identifies also derivatives of licenses | • Certainty that known license text is actually found and wording is exactly reproduced |

Flexibility

<table>
<thead>
<tr>
<th>Nomos Keywords</th>
<th>Nomos Reg. Expressions</th>
<th>Bulk Phrase Matches</th>
<th>Monk Full text Matches</th>
</tr>
</thead>
</table>
| • Very imprecise  
• Does not identify license  
• High number of false positives | • Only limited precision for identifying actual licenses  
• False positives | • Limited to known phrases only  
• Does not provide certainty about original or derivative | • Works only on known license texts  
• Actual occurrences are minority |

Precision
#include<stdio.h>

/*
 * Written by John Doe
 *
 * SPDX-License-Identifier: Apache-2.0
 */

int main() {
    printf("Hello World\n");
    return 0;
}

Gets detected as Apache-2.0 – reuse.software
Combine Ojo results with the other scanners

**Automatize**
- The Ojo information can be combined with the other findings
- If no other scanner found a contradicting statement, the result can be concluded

- MIME-type Analysis (Determine mimetype of every file. Not needed for licenses)
  - Monk License Analysis, scanning for licenses performing a text comparison
  - Nomos License Analysis, scanning for licenses using regular expressions
  - Ojo License Analysis, scanning for licenses using SPDX-License-Identifier
  - Package Analysis (Parse package headers)

7. Automatic Concluded License Decider, based on
   - ... scanners matches if all Nomos findings are within the Monk findings
   - ... scanners matches if Ojo findings are no contradiction with other findings
   - ... bulk phrases from reused packages
### FOSSology Updates – What is new in 3.9.0

- **New agent Spasht**, works with ClearlyDefined.io
- **PostgreSQL 12 support**
- Ability to specify GIT branch in Upload from VCS
- **Reuse of deactivated copyrights**
- **Remove OpenSSL dependency**, use libgcrypt
- **Support for Ubuntu Focal Fossa (20.04)**
- **Obligations now refer to license conclusions**
- **Auto deactivation of copyrights for irrelevant files**
- **Display time in browser's timezone**
- **Ability to export Copyright CSV**

- [https://github.com/fossology/fossology/releases](https://github.com/fossology/fossology/releases)
- **FOSSology Updates**
  - What is new in 3.9.0
FOSSology – Of course you can automate!

REST API
- Manage folders, uploads
- Trigger scans and options
- Download reports
- More info at: https://www.fossology.org/get-started/basic-rest-api-calls/
- https://github.com/fossology/fossology/wiki/FOSSology-REST-API
- (complete flow explained)

REST Clients
- Available in many languages
  - Python (fossology-python)
  - C# (FOSSology.REST.dotnet)
  - Shell (FOSSology.REST.shell)
- FOSSdriver
  - Not only what REST API can do
    - … but also manage bulk scans
- More info at: https://github.com/fossology
- Write your own Python workflow

Command line tools
- Many functions and agents have command line interfaces
  - Nomos an Monk license scanners
  - Copyright scanner
  - License listings
  - …
- Upload and download tools
Uploading a package
Upload using REST Clients

Shell Client

```bash
# Display options
  -c, --description        Upload description
  -d, --debug              Debug mode
  -e, --extra-debug        Extra Debug mode
  -f, --folder             Folder in which the upload will be added
  -g, --group-name         Fossology group name
  -h, --help               This help
  -i, --input              Filename to upload
  -k, --insecure           Skip certificate check in curl command
  -n, --username           Fossology username
  -p, --password           Fossology password
  -r, --rest-url           Full address to Rest API service
  -R, --reuse              Enable reuse
  -s, --site-url           Fossology portal address
                          Enables printing the resulting Fossology URL
  -t, --api-token          API Access Token
  -u, --git-url            URL GIT repository address
  -v, --version            Print current version

FILES_TO_UPLOAD=$(ls /files/*.tgz)
for file in $FILES_TO_UPLOAD
do
    /upload-rest.sh --api-token 'my.secret.token' --rest-url https://fossology.tld/repo/api/v1 --folder 3 --group-name fossy --input $file
done
```

Python Client

```python
# Import Fossology object
from fossology import Fossology

# Import Access levels
from fossology.obj import AccessLevel

# Create request object
foss = Fossology(FOSS_URL, FOSS_TOKEN, username)

# Upload all files inside the folder
uploads = list()
for root, dirs, files in os.walk(folder_to_scan, followlinks=True):
    for file in files:
        file_to_upload = os.path.join(root, file)
        my_upload = foss.upload_file(
            foss.rootFolder, 
            file=file_to_upload, 
            description="Dependency of my project", 
            access_level=AccessLevel.PUBLIC, 
            wait_time=30
        )
        uploads.append(my_upload)
```

Upload using CLI Tool

**cp2foss**

```
using cp2foss:
  -S  = upload from subversion repo
  -G  = upload from git repo
  --user string = user name
  --pass string = password

FOSsology storage options:
  -f path  = folder path for placing files (e.g., -f 'Fedora/ISOs/Disk 1')
  -A       = alphabet folders; organize uploads into folder a-c, d-f, etc.
  -n name  = (optional) name for the upload (default: name it after the file)
  -d desc  = (optional) description for the update

FOSsology processing queue options:
  -q       = list all available processing agents
  -Q       = specify a comma-separated list of agents, or 'all'
  -T       = TEST. No database or repository updates are performed.
  -I       = ignore scm data scanning
```

**Upload sources**

- All the methods support upload from:
  - From file
  - From URL
  - From GIT
  - From SVN
  - From Server
Scanning a package
Scanning upload using REST Clients

**Shell Client**

```bash
$ cat json-templates/scan-options.json
{
  "analysis": {
    "bucket": true,
    "copyright_email_author": true,
    "ec": true,
    "keyword": true,
    "mime": true,
    "monk": true,
    "nomos": true,
    "package": true
  },
  "decider": {
    "nomos_monk": true,
    "bulk_reused": true,
    "now_scanner": true
  }
}

$ ./upload-rest.sh
> --api-token 'my.secret.token' --rest-url https://fossology.ltd/repo/api/v1
> --folder 3 --group-name fossy --input 'file-to-scan.tar.gz'
```

**Python Client**

```python
# Import FOSSology object
from fossology import Fossology

# Import Access levels
from fossology.obj import AccessLevel

# Create request object
foss = Fossology(FOSS_URL, FOSS_TOKEN, username)

# Create scan specification
job_spec = {
    "analysis": {
        "copyright_email_author": True,
        "monk": True,
        "nomos": True,
        "ojo": True
    }
}

# Set the folder where upload is
upload_folder = foss.detail_folder(upload_to_scan.folderid)

# Trigger the scan
foss.schedule_jobs(upload_folder, upload_to_scan, job_spec, group='fossy',
wait=True, timeout=60)
```
Scanning upload using CLI Tool

**fossjobs**

```bash
drax@empress:~/development/fossology (master)$ sudo /usr/local/bin/fossjobs -h
fossjobs [options]
Options:
- H :: help, this message
- v :: verbose output
- a :: list available agent tasks
- A string :: specify agent to schedule (default is everything from -a)
  The string can be a comma-separated list of agent tasks.
- u :: list available upload ids
- U upload :: the upload identifier for scheduling agent tasks.
  The string can be a comma-separated list of upload ids.
  Or use 'ALL' to specify all upload ids.
- D upload :: the upload identifier for scheduling delete tasks.
  The string can be either be 'ALL', a string (the upload_pk),
  or an array of upload_pk's if multiple -D's were specified.
--username string :: user name
--password string :: password
--groupname string :: group name
-c string :: Specify the directory for the system configuration
```

**Scanning options**

- **Analysis**
  - Various agents
- **Decider**
  - Automatic conclusions
  - nomos-monk, ojo, new scanner
- **Bulk reuse**
- **Reuse**
  - Copy clearings from existing upload
Analysing the results
Analysing results using cURL

Upload summary

```
drax@empress:/development/fossology (master)$ curl --silent --request GET
> --header "groupName: fossy"
> --header "Authorization: Bearer eyJhbGciOiJIUlwiX2lXa19pbmF0dG9fZjIyZjI6MzUzNjExODM2NjU1eCIsIkZzdGFyZWh6
3YzY4OTY1MDE4MjA2ODc1MTkzMyIsImxhc3NpZ25UaWNrZ3JvdW5jcnlwdCI6IkF1c2Vmb2luZyIsImlhdCI6MTUyNDIwMDc1NiwicmVmaWVu
LWJhY2t1dGF0aW9uIjoiOntcZDhlYmE1OTflMmI3LTAzYS1hZmFlLTg5MzgtYzFmOGM1ZjltYjJhNikieSxmcDI2N2IwZjBiZS00ZTVi
LWFiOGRlLWFlZjg2LTIxZDEifSwiaWQiOiJkZGlnaXZlLWJhY2t1dGF0aW9uLWZhc3JfZjIyZjI6MzUzNjExODM2NjU1eCIsImNvbXBh
Z2UiOiJub29tZSIsInN0YXJ0IjoiMGFkZmZkNTQwZjE0Mzc5ZjE5MjE2ZTE1Y2E5NzgxN2E0NzZkNjY3M2U1ZjAwZTEiLCJzcGVjaW5kZiI
\n python -m json.tool

{ "id": 195, "uploadName": "virtualenv-20.0.31.tar.gz", "mainLicense": "MIT", "uniqueLicenses": 14, "totalLicenses": 240, "uniqueConcludedLicenses": 9, "totalConcludedLicenses": 356, "filesToBeCleared": 0, "clearingStatus": "Open", "copyrightCount": 106 }
```

Upload licenses

```
drax@empress:/development/fossology (master)$ curl --silent --request GET
> --header "groupName: fossy"
> --header "Authorization: Bearer eyJhbGciOiJIUlwiX2lXa19pbmF0dG9fZjIyZjI6MzUzNjExODM2NjU1eCIsIkZzdGFyZWh6
3YzY4OTY1MDE4MjA2ODc1MTkzMyIsImxhc3NpZ25UaWNrZ3JvdW5jcnlwdCI6IkF1c2Vmb2luZyIsImlhdCI6MTUyNDIwMDc1NiwicmVmaWVu
LWJhY2t1dGF0aW9uIjoiOntcZDhlYmE1OTflMmI3LTAzYS1hZmFlLTg5MzgtYzFmOGM1ZjltYjJhNikieSxmcDI2N2IwZjBiZS00ZTVi
LWFiOGRlLWFlZjg2LTIxZDEifSwiaWQiOiJkZGlnaXZlLWJhY2t1dGF0aW9uLWZhc3JfZjIyZjI6MzUzNjExODM2NjU1eCIsImNvbXBh
Z2UiOiJub29tZSIsInN0YXJ0IjoiMGFkZmZkNTQwZjE0Mzc5ZjE5MjE2ZTE1Y2E5NzgxN2E0NzZkNjY3M2U1ZjAwZTEiLCJzcGVjaW5kZiI
\n python -m json.tool

```
## Analysing results using REST Client

### Upload summary

```python
# Import FOSSology object
from fossology import FOSSology
# Import Access levels
from fossology.obj import AccessLevel
# Create request object
foss = FOSSology(FOSS_URL, FOSS_TOKEN, username)
# Get the upload object
upload = foss.detail_upload(195, group='fossy')
# Get the upload summary
upload_summary = foss.upload_summary(upload, group='fossy')
# Print the upload details
print("upload name: ", upload_summary.upload_name)
print("upload main license: ", upload_summary.main_license)
print("upload state: ", upload_summary.clearing_status)
print("unique licenses in upload: ", upload_summary.unique_licenses)
print("clearing: ", upload_summary.files_cleared) +
    f"/[upload_summary.files_to_be_cleared]"
```

### Upload licenses

```python
# Import FOSSology object
from fossology import FOSSology
# Import Access levels
from fossology.obj import AccessLevel
# Create request object
foss = FOSSology(FOSS_URL, FOSS_TOKEN, username)
# Get the upload object
upload = foss.detail_upload(195, group='fossy')
# Get the upload licenses
upload_licenses = foss.upload_licenses(upload, group='fossy',
    agent="nomos,monk,ojo", containers=False)
# Print the licenses
for license in upload_licenses:
    print("In file ", license.filepath) =>")
    print(" Scanner: ", end='')
    print(" Conclusion: ", end='')
    print(license.findings.scanner)
    print(license.findings.conclusion)
```
## Analysing results using CLI tools

<table>
<thead>
<tr>
<th>Nomos Licenses</th>
<th>Other tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>• fo_nomos_license_list</td>
<td>• fo_nomos_license_list</td>
</tr>
<tr>
<td>• All findings by nomos scanner</td>
<td>• All findings by monk scanner</td>
</tr>
<tr>
<td>• Can exclude file paths</td>
<td>• Can exclude file paths</td>
</tr>
<tr>
<td>• fo_monk_license_list</td>
<td>• fo_monk_license_list</td>
</tr>
<tr>
<td>• All findings by monk scanner</td>
<td>• All copyright findings</td>
</tr>
<tr>
<td>• Can exclude file paths</td>
<td>• Can filter copyright content</td>
</tr>
<tr>
<td>• fo_copyright_list</td>
<td></td>
</tr>
<tr>
<td>• All copyright findings</td>
<td></td>
</tr>
<tr>
<td>• Can filter copyright content</td>
<td></td>
</tr>
</tbody>
</table>
Generating reports
Generating reports using REST Clients

**Shell Client**

```bash
$ curl -X POST https://fossology.org/api/v1/reports
```

**Python Client**

```python
# Import FOSSology object
from fossology import FOSSology

# Import report formats
from fossology.report import ReportFormat

# Create request object
foss = FOSSology(FOSS_URL, FOSS_TOKEN, username)

# Get the upload object
upload = foss.upload_detail(upload_id, group='fossy')

# Generate the report
report_id = foss.generate_report(upload, ReportFormat.UNIFIEDREPORT, group='fossy')

# Download the report
with open(report_name, 'w+') as report_file:
    report_file.write(report_content)
```

Report formats available: dep5, spdx2, spdx2tv, readmeoss, unifiedreport
### Atarashi
- Standalone license scanner
- Written in Python
- Implement multiple text statistics and information retrieval algorithms

More info at: [https://github.com/fossology/atarashi](https://github.com/fossology/atarashi)

### FOSSologySlides
- Slides for Presenting FOSSology
- Make your presentation with FOSSology
- Material for a 1-day training

More info at: [https://github.com/fossology/FOSSologySlides](https://github.com/fossology/FOSSologySlides)

### Nirjas
- Python library and tool
- Extract source code and comments
- Supports 25 languages
- Differentiate single line, multiline and continued single line comments

More info at: [https://github.com/fossology/Nirjas](https://github.com/fossology/Nirjas)
Questions? – Consider to “Star Us”!

Gaurav Mishra  
Siemens AG  
gmishx@gmishx.in

Shaheem Azmal  
Siemens AG  
shaheem.azmal@gmail.com

Anupam Ghosh  
Siemens AG  
anupamghosh.ind@gmail.com

FOSSology links  
https://www.fossology.org/  
https://github.com/fossology/fossology

FOSSology - YouTube  
https://www.youtube.com/channel/UCZGPJnQZVnEPQWxOuNamLpw