공학 트렌드

소프트웨어공학센터 경영지원TF팀

JDK/WEB/WAS 서버 표준화와 오픈소스 거버넌스 Part 2 : 오픈소스 거버넌스 시스템 구축

2014. 11. 28. [제111호]

Contents

정보통신산업진흥원 부설
nipa 소프트웨어공학센터

Ⅰ. 오픈소스(Open Source) 소개

1.1 상용 소프트웨어와 오픈소스

상용 소프트웨어는 사용자가 비용을 지불하고 소프트웨어 사용할 수 있는 패키지를 받거나 웹 또는 이메일로 패키지를 다운로드 해서 사용한다. 소프트웨어는 법적 보호를 받을 수 있기 때문에, 개발자가 소프트웨어를 개발하면 저작권이 자동 생성되며, 사용 권리는 개발자나 회사에 부여 받는다. 저작권이 있는 저작물은 저작권자의 허락 없이는 저작물을 복제, 배포, 수정 등이 불가능하다.

소프트웨어 라이센스는 소프트웨어 자체에 대한 소유권과는 별개로 소프트웨어를 사용할 수 있는 권리를 의미한다. 즉, 단순히 라이센스를 받음(계약)으로서 저작권자로부터 일정 간 조건을 바탕으로 소프트웨어 사용 권리를 승인 받음을 의미한다. 따라서 사용권만 받은 소프트웨어를 복제나 전송 등 저작권 규정에 위반한 행위가 발생하면 저작권 침해가 일어날 수 있다. 상용 소프트웨어의 라이센스는 허용 기간, 사용 기준, 공급 형태에 따라 달라진다. 워낙 다양한 소프트웨어를 쓰고, 직원들의 이동이 잦을수록 라이센스 계약을 잘 챙기지 않아서 어려움을 많이 겪기도 한다. 이에 따라 관리 포인트 점검과 교육이 상당히 요구되는 상황이다.

상용 소프트웨어에 반발하여 자유로운 사용을 위해서 시작된 프리 소프트웨어 (Free Software) 가 Richard Stallman과 FSF[1]에 의해 시작되었다. 상용 소프트웨어의 한계를 넘어서는 프리 소프트웨어 운동은 소프트웨어 사용자에게 실행, 복제, 배포의 자유, 소스 코드 공유를 통해 학습, 수정, 개선할 수 있는 자유를 부여하자고 주장했다. 프리 소프트웨어 진영의 급격한 운동은 GPL(General Public License)로 구체적으로 발전되었고, Linux가 이 라인센스로 개발되었다. 그리고 OSI[2] 단체가 설립되었다. 오픈소스 소프트웨어로 인정받을 수 있는 라인센스 조건을 정의 했다. 소프트웨어 사용자에게 배포, 수정의 자유가 있으며 해당 소스 코드를 제공할 수 있도록 했다.

오픈소스라고 해서 완전히 자유롭게 쓸 수 있는 것은 아니다. 오픈소스의 철학이 있는데, 무료로 쓸 수 있다고 해서 상용 소프트웨어를 개발 할 수는 없다. 다만, 오픈소스

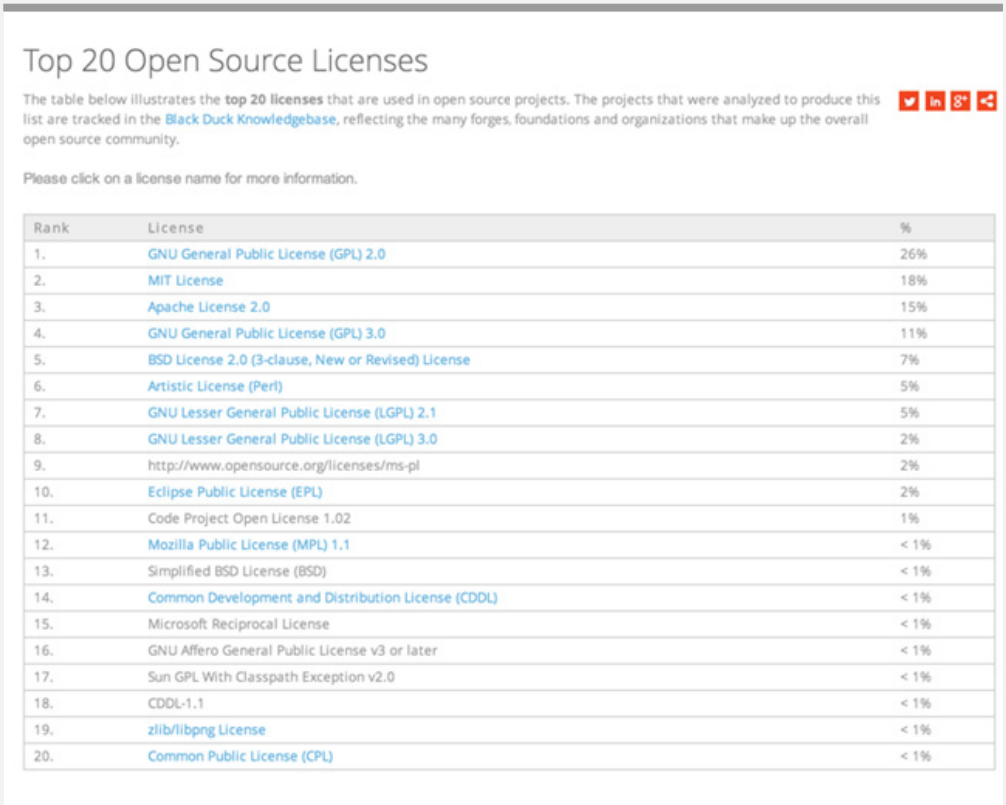
1) Free Software Foundation, http://www.fsf.org/
2) Open Source Initiative, http://opensource.org/

를 이용해 개발하면서 복제/설치/배포의 자유에 따라 오픈소스를 활용한 소프트웨어를 공개하는 의무가 있을 수 있다. 만약 이 부분을 준수하지 않을 경우에는 저작권 분쟁으로 이어질 수 있다.

OSI(Open Source Initiative)가 인증한 라이센스는 다음과 같다.

- Apache License 2.0
- BSD 3-Clause "New" or "Revised" license
- BSD 2-Clause "Simplified" or "FreeBSD" license
- GNU General Public License (GPL)
- GNU Library or "Lesser" General Public License (LGPL)
- MIT license
- Mozilla Public License 2.0
- Common Development and Distribution License
- Eclipse Public License

그림 1_Top 20 Open Source Licenses



Top 20 Open Source Licenses

The table below illustrates the **top 20 licenses** that are used in open source projects. The projects that were analyzed to produce this list are tracked in the Black Duck Knowledgebase, reflecting the many forges, foundations and organizations that make up the overall open source community.

Please click on a license name for more information.

Rank	License	%
1.	GNU General Public License (GPL) 2.0	26%
2.	MIT License	18%
3.	Apache License 2.0	15%
4.	GNU General Public License (GPL) 3.0	11%
5.	BSD License 2.0 (3-clause, New or Revised) License	7%
6.	Artistic License (Perl)	5%
7.	GNU Lesser General Public License (LGPL) 2.1	5%
8.	GNU Lesser General Public License (LGPL) 3.0	2%
9.	http://www.opensource.org/licenses/ms-pl	2%
10.	Eclipse Public License (EPL)	2%
11.	Code Project Open License 1.02	1%
12.	Mozilla Public License (MPL) 1.1	< 1%
13.	Simplified BSD License (BSD)	< 1%
14.	Common Development and Distribution License (CDDL)	< 1%
15.	Microsoft Reciprocal License	< 1%
16.	GNU Affero General Public License v3 or later	< 1%
17.	Sun GPL With Classpath Exception v2.0	< 1%
18.	CDDL-1.1	< 1%
19.	zlib/libpng License	< 1%
20.	Common Public License (CPL)	< 1%

출처: https://www.blackducksoftware.com/resources/data/top-20-open-source-licenses

오픈소스는 소스 코드를 대중들에게 공개해서 누구나 자유롭게 사용할 수 있고, 수정

이 가능하다. 그러나 수정과 재배포에 관련해서 라이센스 내용이 각각 달라 조심스럽게 사용할 필요가 있다. 라이센스마다 특징이 있기 때문에 기업에서 일하는 개발자는 신중하게 라이브러리 선택을 할 필요가 있다. 참고로 오픈소스 소프트웨어를 검증하는 한 업체인 Black Duck의 정보[3]에 따르면, GPL 라이센스가 37% 정도 되며, MIT 라이센스가 18%, Apache 라이센스가 15% 정도 된다고 한다.

다음 장에서 주요 라이센스에 대해서 간단히 설명하고자 한다.

II. 오픈소스 라이센스

오픈소스 라이센스의 자세한 내용은 참조를 이용한다. 기업에서 개발자가 소스 수정과 소프트웨어 Linking하는 프로그래밍 시 주의해야 하는 중요한 저작권, 소스 공개 관점에서 접근한다.

필자는 코드 프로젝트 라이센스 site와[4] OSI를[5] 바탕으로 정리했다.

- **GPL (General Public License)**

GPL 라이센스는 다른 오픈소스 라이센스에 비해서 강하다. 소스를 수정하거나 새로운 소프트웨어에 Linking 할 때 소스 코드를 모두 공개해야 한다.

- **LGPL (Lesser General Pulbic License)**

GPL 2.0 보다는 완화된 형태이다. GPL 과 비슷하게 소스 수정할 때는 소스 코드를 공개해야 한다. 새로운 소프트웨어를 라이브러리 Linking 할 때는 소스 코드를 공개할 필요는 없다.

- **BSD (Berkely Software Distribution)**

GPL, LGPL처럼 강한 라이센스가 아니다. 소스 수정 시 소스 코드 공개 의무가 없다. 게다가 특허권 행사가 가능하지 않다.

- **Apache License**

Apache Tomcat, Apache Httpd 서버와 같이 아파치 재단 (Apache Foundation)이 지원하

3) https://www.blackducksoftware.com/resources/data/top-20-open-source-licenses
4) http://www.codeproject.com/info/Licenses.aspx
5) http://opensource.org/licenses/alphabetical

는 모든 소프트웨어에 적용된다. 소스 수정 시 소스 코드 공개 의무가 없다. 그러나 Apache 라는 상표권을 침해하지 않아야 한다는 명시적인 조항이 들어가 있으며, 저작권 보호를 하고 있다.

- **MPL (Mozilla Public License)**

Netscape 브라우저의 소스 코드를 공개하기 위해 만들어졌다. 원래 소스코드가 아닌 새로운 파일을 추가하는 경우 공개의 의무는 없다. 그러나 소스 코드를 수정할 경우에는 공개해야 한다. 특허권을 주장할 수 없다는 것이 특징이다.

- **Eclipse Public License**

Eclipse 재단이 지원하고 있으며, 소스 코드를 수정 및 Linking 시 해당 모듈을 공개해야 한다.

- **MIT License**

저작권이 보호되지만, 특허권을 행사할 수 없다. 소스 코드를 수정해도 소스 코드 공개 의무가 없다.

- **CDDL (Common Development and Distribution License)**

저작권이 보호되며, 소스 코드를 수정해도 소스 공개 의무가 없다.

- **AGPL (GNU Affero Generl Public License)**

Affero사의 요청에 의해 만들어졌다. 기존 GPL 보다 더 강하여 해당 라이선스를 수정 또는 배포하는 경우 컴퓨터 네트워크를 기반으로 동작하는 응용 프로그램의 리소스를 사용하는 모든 사용자에게 소스를 공개해야 한다.

GPL, LGPL, MPL 라이센스를 가진 오픈소스를 수정 할 경우 수정한 소스 코드를 공개해야 한다. GPL의 경우는 Linking을 하면 사용하고 있는 관련 소스를 모두 공개해야 하며 라이센스 전파 의무를 수행해야 한다.

GPL 라이센스 경우에는 어떻게 사용하고 있는지 잘 살펴봐야 한다. 특히 동영상 코덱을 사용하는 클라이언트 배포의 경우에는 소스 공개의 의무가 있다. 예를 들어 ffmpeg는 GPL 또는 LGPL 라이선스[6]라서 서버 쪽은 링킹 이슈가 아니면 쉽게 쓸 수 있지만, 사용자 PC에 배포되는 클라이언트 App 또는 안드로이드/IOS App으로 배포되는 경우에는 치명적인 이슈를 부를 수 있다.

AGPL 라이센스는 클라이언트 뿐만 아니라 서버에도 모두 공개해야 하는 강한 라이센스를 가지고 있다. 때문에 수정 또는 배포하여 통신하는 경우, 통신하는 부분의 모든 사

6) 참조 https://www.ffmpeg.org/legal.html

용자에게 오픈소스로 공개해야 한다. 예를 들어 Nosql로 유명한 neo4j, mongodb 의 특정 파트는 AGPL 로 되어 있다. 자칫 실수로 모든 소스를 공개하는 상황이 올 수도 있으므로 정확하게 파악하고 사용해야 한다. 만약 포털 기업이 AGPL 라이센스에 해당하는 오픈소스를 수정, 배포한다면 사실상 모든 고객에게 소스를 공유해야 하므로 완전한 소스 공개인 것과 다름없다.

오픈소스 라이센스에 대한 자세한 정보는 한국 소프트웨어 포털의 공개 SW 라이센스 링크[7] 내부 지면 가이드 PDF 문서 참고하기를 추천한다.

그림 2_한국 소프트웨어 포털의 공개 SW 라이센스 지면

SW 라이선스 분류

소스코드 공개 여부에 따른 라이선스 비교

구 분	무료 라이선스	유료 라이선스
공 개	대부분 OSS (Apache, Tomcat, JBoss)	일부 배포판 OSS (Red Hat, MySQL)
비공개	Freeware(Winzip, RealAudio) Shareware(초기에만 무료) Adware(JetAudio)	상용 비공개 소프트웨어 (MS Office, Oracle, Shareware)

주요 공개SW 라이선스 비교

	무료 이용가능	배포 허용가능	소스코드 취득가능	소스코드 수정가능	2차적 저작물 재공개 의무	독점SW와 결합가능
GPL	O	O	O	O	O	X
LGPL	O	O	O	O	O	O
MPL	O	O	O	O	O	O
BSD license	O	O	O	O	X	O
Apache lilcense	O	O	O	O	X	O

출처: http://www.oss.kr

7) http://www.oss.kr/oss_intro06

Ⅲ. 오픈소스 라이센스 관리 / 활용

3.1 클라이언트 및 서버 관점에서 접근

클라이언트와 서버 관점에서 접근할 수 있다.

클라이언트인 스마트폰, PC 소프트웨어로 개발하는 경우로서, 배포할 버전을 배포하기 전에 오픈소스 거버넌스 부서에 전달해 오픈소스 검증을 받도록 한다. 오픈소스 거버넌스의 조직에서는 두 가지로 분류된 정보를 저장한다. 하나는 프로젝트에서 쓰이는 라이브러리 정보를 수집하는 정보이며 나머지는 수집된 정보를 바탕으로 라이브러리 관점에서 수집하는 정보이다. 후자의 경우는 라이브러리 단위로 저장하기에 신규 프로젝트에서 쉽게 라이브러리 정보를 이용할 수 있으며 시간이 지날수록 빠른 검증이 가능하다.

프로젝트 관점에서 오픈소스 거버넌스 DB에 프로젝트이름, 개발 플랫폼, 개발언어, 라이브러리 이름, 라이브러리 버전, 개발자/관리자 정보를 구분하여 저장한다.

프로젝트 이름	개발 플랫폼	개발 언어	라이브러리 이름	라이브러리 버전	개발자/관리자 정보
rookie	android	java	guava	2.1	nipa@gmail.com
...					

라이브러리 관점에서 저장하는 정보는 개발플랫폼, 개발언어, 라이브러리 이름, 라이브러리, 라이센스, 사용 가능 여부이다.

개발 플랫폼	개발 언어	라이브러리 이름	라이브러리 버전	라이센스	사용 가능 여부
android	java	guala	2.1	GPL	X
...					

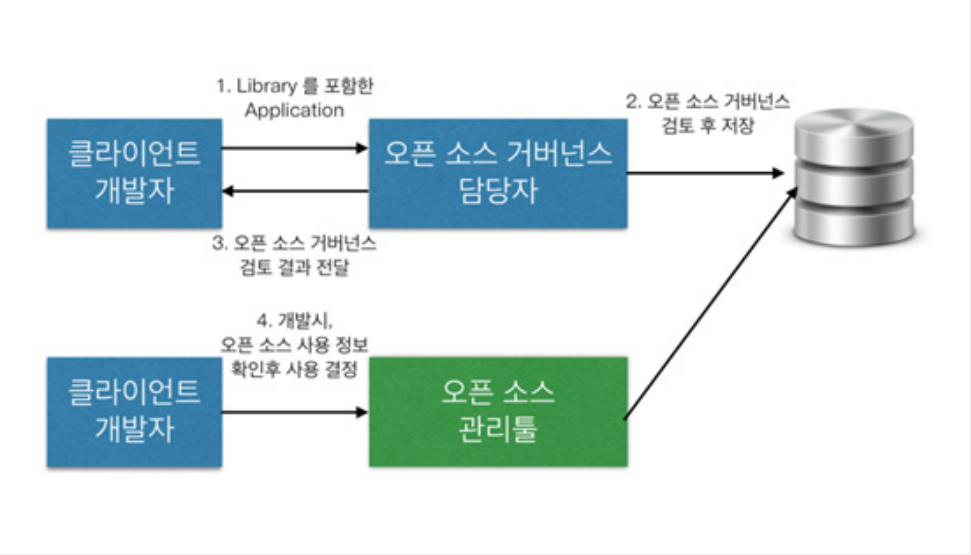
이 두 개의 정보를 바탕으로 라이센스 이슈(위반)이 발생하면 오픈소스 거버넌스 조직은 개발 부서에 관련 내용을 전달하고 사용 여부 및 개선 가이드를 전달한다. 클라이언트는 특성상 한 번 배포되면 영향력이 크기 때문에 사람의 매번 체크하는 매뉴얼 방식을 채택할 수 있으나 자동화할 수 있는 부분은 최대한 높여야 한다.

축적된 오픈소스 거버넌스 데이터를 볼 수 있는 오픈소스 관리 툴을 내부로 공유한

후에 클라이언트 사내 개발자들에게 공유해 오픈소스 라이브러리를 개발 또는 배포 전에 미를 정보를 쉽게 이용할 수 있다.

클라이언트 개발자 관점의 오픈소스 거버넌스 과정이 〈그림 3〉에 담겨 있다.

그림 3_클라이언트 개발자 관점에서의 오픈소스 거버넌스 과정



클라이언트 단말은 고객에게 주어지게 되고 배포하는 과정이 포함되기 때문에 라이센스 이슈가 매우 중요하다. Skype가 운영하는 VoIP 클라이언트 단말에 GPL 라이센스 라이브러리 2개가 포함되면서 소스 코드 공개와 벌금이 처해지는 사건이 있었다.[8] 그 외 삼성전자에서도 비슷한 사례가 발생했다. 따라서 스마트폰과 같은 어플리케이션 클라이언트는 오픈소스 라이센스 정책을 명확히 알고 사용할 수 있어야 한다.

최근에는 스마트폰 제조회사에서 탑재되는 어플리케이션의 오픈소스 라이브러리 의 라이센스를 검증해 탑재 여부를 어플리케이션에 통보하고 위와 같은 이유로 어플리케이션 선 탑재를 하지 않겠다는 공문을 보내고 있다. 따라서 오픈소스 사용 여부에 대한 세밀한 검토가 필요하다.

반면, 서버 관점에서는 오픈소스 정보를 받는 시점이 여러 부분에서 존재 할 수 있다. 빌드하는 시점, 배포하는 시점 상용 서버 군에서 배포하는 시점, 배포된 상용 서버에서 오픈소스 라이브러리 정보를 수집할 수 있다. 필자가 소속해있던 회사는 다양한 형상관리, 브랜치 정책이 존재해서 배포가 일어난 후의 상용 서버 정보를 수집했다. 사용 서버의 WEB/WAS 표준 디렉터리에서 실행하는 프로세스의 라이브러리 정보 수집을

8) http://www.oss.kr/index.php?mid=oss_license&sort_index=readed_count&order_type=desc&document_srl=66978

하는 방식을 사용했다.(원칙적으로 접근하기 위해서는 상용서버에 배포하기 전에 오픈소스 거버넌스가 진행되도록 하는 것이 좋다. 배포 서버를 통일한다면 배포서버를 활용해 지속적인 데이터 수집이 가능하고, 배포 전에 문제를 해결 할 수 있을 것이다.)

클라이언트와 비슷한 관점으로 프로젝트 단위와 라이브러리 관점으로 라이브러리를 수집할 수 있다. 프로젝트 관점에서 쓰이는 라이브러리를 프로젝트 이름, 호스트 명, 개발 언어, 라이브러리 이름, 라이브러리 버전, 개발자/관리자 정보로 분류해 저장한다.

프로젝트 이름	호스트 이름	개발 언어	라이브러리 이름	라이브러리 버전	개발자/관리자 정보
google mail	gm123	java	guava	1.2	nipa@gmail.com
...					

라이브러리 관점으로 개발언어, 라이브러리 이름, 라이브러리 버전, 라이센스, 사용 가능 여부를 저장한다.

개발 언어	라이브러리 이름	라이브러리 버전	라이센스	사용 가능 여부
java	guava	1.2	MIT	O
...				

언급한 두 개의 정보를 바탕으로 라이센스 사용에 적합하지 않은 경우, 또는 허락을 받지 않은 상용 라이브러리를 쓰는 경우에 Error 메시지를 전달한다. 그러나 아주 크리티컬 하지 않는다면 Warn 메시지를 전달한다.

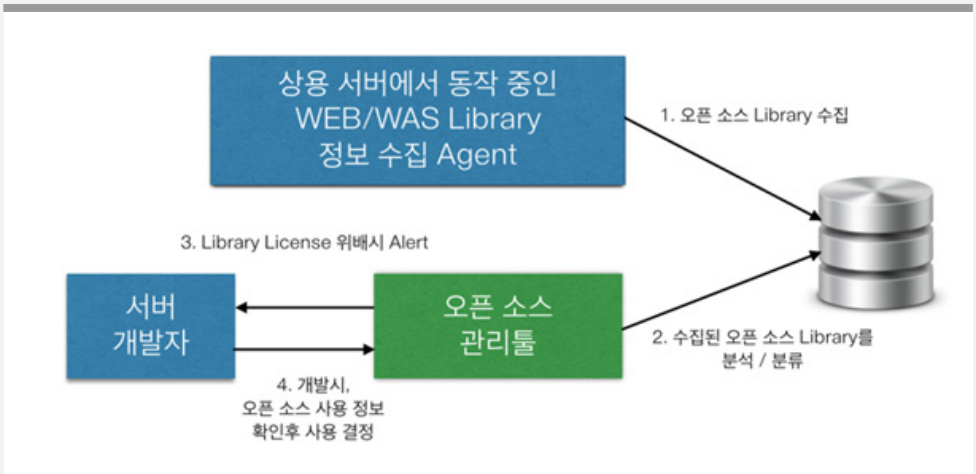
매일마다 서버를 체크하고 데이터를 수집했다. 데이터 수집 정보는 호스트의 JDK, WEB, WAS 의 모든 라이브러리와 버전 정보다.

프로젝트 이름	호스트 이름	분류	버전	라이브러리 이름	라이브러리 버전
google mail	gm123	JDK	1.7	rt.jar	1.7
...					

오픈소스 거버넌스 포털을 만들어 오픈소스 거버넌스 정보를 모두에게 공유하고 개발자가 어떤 라이브러리를 적절하게 잘 사용했는지 알린다. 〈그림 4〉는 서버 관점에서의 오픈 거버넌스 과정을 설명하고 있다.

서버 관점에서는 오 소스 라이브러리 이슈는 그리 크지 않다. 특별히 GPL/LGPL/MPL 라이센스를 가진 라이브러리의 소스를 직접 수정하지 않을뿐더러 일반 사용자 배포를 하지 않기 때문에 그냥 사용해도 큰 무리가 없는 것으로 알려져 있다.

그림 4_서버 개발자 관점에서의 오픈소스 거버넌스 과정



클라이언트나 서버에서 개발할 때 오픈소스 라이브러리의 라이센스를 잘 모르는 부분이 존재한다. 이슈가 되는 오픈소스 라이브러리를 사용할 때는 법무팀을 통해서 확인해야 한다. 이와 같은 도움을 받을 수 없는 경우에는 공개 소프트웨어 포털에[9] 문의 하도록 한다. 현재 정보통신산업진흥원 공개 소프트웨어 포털에서 국내 중소기업 및 개발자 개인, 학교 및 비영리 연구기관을 대상으로 무료 오픈소스 라이센스 검증 서비스를 하고 있다. 오픈소스 투자가 가능하면, 오픈소스 거버넌스를 진행할 수 있는 상용툴을 구매해 정확한 데이터와 생산성을 높일 수 있다.

그리고 클라이언트 또는 서버의 운영체제에서도 마찬가지로 체크하는 것도 좋다. 리눅스 배포판에서의 GPL 라이선스 이슈가 라이브러리를 포함한 운영체제를 사용했다가 소송이 걸리는 사례가 있었으니, 운영 체제 선택에도 신중할 필요가 있다. 수집 서버에 JDK, WEB, WAS 뿐 아니라 운영체제와 DB 배포판의 정보와 라이브러리 정보도 취합해 저장한다. 클라이언트와 서버의 오픈소스 관리는 비슷하므로 함께 저장하여 중복 작업이 없게 할 수 있다.

3.2 서버 운영 관점의 오픈소스 거버넌스

지금까지 단순히 오픈소스 라이브러리의 라이센스만 체크하는 형태를 가졌다. 저작권, 특허권 침해를 방어 또는 방지할 수 있는 여건들은 모아졌다. 그러나 이것이 모든 이슈를 포함하지 않는다. 보안 이슈와 성능 이슈, 안정성(크리티컬 버그) 이슈 등이 발생할 수 있다. 개발자들의 집단 지성이 모아 이슈를 해결할 수 있도록 방안이 필요하다.

9) http://www.oss.kr/oss_open1_3

첫 번째는 보안 이슈이다.

운영체제의 SSH 보안 버그, Web, WAS, JDK 의 자체적인 보안 버그도 상당히 많이 발생하고 있다. 만약 데이터 수집을 하지 않고 있다면 모든 개발자들이 체크를 해야 하는 상황이다. 그러나 오픈소스 정보를 수집한 경우라면 보안 이슈가 있는 버전이 설치된 호스트 이름을 한 번에 찾을 수 있을 뿐 아니라 관리하는 개발자나 운영자들에게 관련 정보를 전달하여 빠른 패치를 일어나게 할 수 있다.

오픈소스 관리 툴에 운영체제의 보안 이슈에 해당되는 운영 체제 라이브러리 정보를 수집 / 저장하도록 했다.

두 번째는 성능 이슈이다.

WEB, WAS 오픈소스 라이브러리들이 쓰기에 편한 것을 선택하여 개발자들이 사용하기도 했으나, 업그레이드되면서 특정 버전에서는 성능 저하가 발생하는 경우가 있다. 그리고 비슷한 기능을 하는 라이브러리 중 성능이 안 좋은 라이브러리를 사용하면서 전체적인 성능 저하가 발생하는 경우도 있다. 이런 이슈로 인해서 업그레이드, 때로는 다운 그레이드를 권장해 성능 안정적인 버전을 가이드까지 하게 된다. 예를 들어 Apache 의 dbcp와 pool 라이브러리 버전을 적절한 버전으로 패치하기도 했고 안정적인 버전이 나오면 테스트 후 적용할 수 있도록 가이드 할 수 있다. 각 개발 부서에서 경험한 좋은 정보들을 바탕으로 안정적인 버전을 체크할 수 있다. 그렇다고 해서 무조건 안정적인 버전을 쓰라고 강요할 수는 없다. 새로운 도전에 따른 성능 저하는 언제나 생길 수 있고, 관련 정보를 모두 받아 전사에 관련 정보를 전달할 수 있다.

세 번째는 안정성(크리티컬 버그) 이슈이다.

새로 나온 라이브러리를 쓰다가 크리티컬 버그가 발생해 이슈가 되는 경우, 빠른 전파를 진행할 수 있다. 많은 개발부서는 크리티컬 버그 이슈를 모르고 있다가 똑같은 문제가 재현되고 나서 문제 해결을 위해 반복되는 고생이 없도록 크리티컬 버그 버전은 사용하지 않도록 할 수 있다. 예를 들어 새로 나온 JDK 의 버전을 쓰다 GC 부분에서 크리티컬 버그가 발견되었다. Oracle에 버그를 공유함과 동시에 오픈소스 거버넌스 툴에서 해당 버전 사용 시 개발자와 운영자에게 수정하라는 관련 메일을 보내 해당 버전을 쓰지 않도록 독려할 수 있다.

만약 개발자나 운영자가 오픈소스 및 보안, 성능, 안정화를 해결하지 않아도 오픈소스 거버넌스 툴이 매일 체크하여 트래킹이 되도록 한다. 서버의 안정화는 곧 서비스의 안정화를 의미하는 것이기 때문에 작은 이슈라도 정리할 수 있도록 지원한다.

개발 언어	라이브러리 이름	라이브러리 버전	라이센스	사용 가능 여부	이슈	설명
java	common-dbcp	1.2.0	Apache	X	안정성	메모리가 어느 정도 커지면 초기화되는 버그
...						

WEB, WAS 의 오픈소스 라이브러리 정보, JDK와 같은 주요 언어 툴, 운영체제에 연관된 보안과 특정 License를 포함한 오픈소스 라이브러리 정보를 수집하고 이 정보를 바탕으로 이슈를 핸들링 하는 프로세스와 시스템이 필요하게 된다. 모든 개발 부서의 개발자에게 메일을 통해서 매번 새로 나온 정보를 전달하거나 정책을 계속 전달하기가 어려웠다. 그래서 오픈소스 포털을 개발하여 중요 오픈소스 라이브러리나 오픈소스는 아니지만 중요한 라이브러리의 정보를 모으고 카테고리화 하여 개발자들이 언제든지 살펴볼 수 있도록 한다.

오픈소스 라이센스 검증 뿐 아니라 성능, 보안, 안정성 등을 잘 이해할 수 있어야 하므로 오픈소스 거버넌스 조직은 기술과 법적 리스크를 고려할 수 있으며, 빠르고 정확한 피드백을 줄 수 있는 구성원으로 조직화 해야 한다. 모든 일을 다 할 수 없기 때문에 보안팀과 개발팀과의 긴밀한 협조로 시스템을 성숙시켜야 한다.

〈그림 5〉는 지금까지의 내용을 도식화한 것이다. 오픈소스 라이브러리 검증, 보안 검증, 성능 검증, 안정성을 검증하는 역할을 하는 오픈소스 거버넌스 툴이다. 오픈소스 거버넌스 툴은 오픈소스 라이브러리 에 수집된 정보가 저장된 오픈소스 관리툴, 호스트에서 동작하는 수집 Agent, 개발자들에게 정보를 제공하고 소통할 수 있는 오픈소스 포털을 바탕으로 오픈소스 라이브러리 사용을 원칙적으로 적용하고자 했던 필자의 그림을 소개한다.

그림 5_오픈소스 거버넌스 툴 도식화

Ⅳ. 결론

표준화된 JDK/WEB/WAS 라이브러리에 어떤 버전의 오픈소스 라이센스를 쓰는지 검토할 수 있고, 그 밖의 라이브러리의 성능, 안정성, 보안을 검토하여 지속적으로 JDK/WEB/WAS 표준환경을 계속 보완할 수 있다. 또한 클라이언트가 사용하는 라이브러리의 오픈소스 라이센스와 서버에서 사용하는 라이브러리의 라이센스 정책을 잘 저장하면 신규 서비스나 프로젝트에서 개발 초기 또는 중간에 이슈가 있는 라이브러리를 정리할 수 있다. 그리고 개발하면서 얻은 오픈소스 활용에 대한 성능, 보안, 시행착오의 내용을 오픈소스 관리 툴에 내용을 담아 같이 공유할 수 있다.

오픈소스는 소스 코드가 공개되어 빠른 기술 습득과 개발을 가능하게 한다. 그래서 개발 원가를 줄일 수 있는 큰 장점이 있지만, 라이센스를 위배하면 법적 책임뿐 아니라 모든 소스를 공개해야 할 책임을 질 수 있다.

현실상 개발자가 프로그래밍 하는 부분에 있어서 오픈소스 사용의 권리와 책임을 다 알고 개발하는 경우는 흔치 않다. 배포한 후에나 이슈가 생길 때에야 비로소 발견하는 경우가 많다. 따라서 오픈소스 거버넌스 시스템 구축을 통해서 빠른 문제 해결을 처리함으로써 회사가 감내해야 할 리스크를 줄일 수 있다.

본 원고를 통한 필자의 경험 공유가 오픈소스 거버넌스를 보다 쉽게 이해하고 효과적이고 효율적인 시스템을 구축하는데 도움이 되기를 바란다.

참고 자료

1. http://opensource.org
2. https://wiki.kldp.org/wiki.php/OpenSourceLicenseGuide
3. http://www.oss.kr/