EQST insight

오픈소스SW를 안전하게 활용하기 위한 보안관리 방안

금융컨설팅담당 최경호, 김희승 수석

1. 오픈소스 SW 현황

오늘날 오픈소스 소프트웨어(OSS: Open Source Software, 이하 오픈소스 SW)는 적은 비용 부담으로 서비스를 혁신할 수 있는 개발 수단으로 인식되어 활용 빈도와 시장가치가 급속도로 높아지고 있다. 2021 년 국내 기업 및 기관의 오픈소스 SW 활용률은 61.5%를 기록하고 있으며, 사업의 규모가 클수록 오픈소스 SW 활용률은 높아지고 있다.[1]

〈표 1〉 오픈소스 SW 활용 현황 및 시장가치(2021 년 기준)

구분	국내 오픈소스 활용 수준(%)			국내 오픈소스 SW 시장(억 원)	
	부분 활용	전사적 활용	합계	규모	가치
내용	35.1	26.4	61.5	3,302	70,000

오픈소스 SW 는 인터넷 등에 무상으로 공개된 소스코드 또는 소프트웨어로 손쉬운 개량, 기능 활용 및 재배포가 가능하여 사용이 확산되고 있다. 그러나 라이선스 위반으로 인한 법적 분쟁, 공개된 소스코드에 대한 취약점 공격, 오류 수정 또는 추가 개발이 어려운 점 등의 문제점들이 존재하므로, 오픈소스 SW 를 활용하기 전 이를 효과적으로 관리할 수 있는 체계가 필요하다.

1 정보통신산업진흥원, 2021 오픈소스 SW(OSS) 실태조사 보고서, 2021. 11

〈표 2〉 오픈소스 SW 의 라이선스 및 취약점 문제 사례

[라이선스 위반 사례][2]		[오픈소스 SW 취약점 공격 사례①]	
open source initiative	2018 년, 국내 H 기업, PDF 변환 라이브러리 오픈소스 코드 사용에 대한 라이선스 분쟁으로 205 만 달러(약 23 억 원) 합의금 발생	LOG4J	JAVA 서버의 로깅 프레임워크 Log4j 취약점을 악용해 관리자 권한을 탈취하는 사상 최악의 보안 결함으로, 전세계 거의 모든 서버가 위협에 노출되었음
[오픈소스 SW 취약점 공격 사례②]		**[오픈소스 SW 취약점 공격 사례③]**	
	Octopus Scanner 라는 멀웨어는 개발자들의 NetBeans 프로젝트를 감염시켜 사용자들에게 유포되는 오픈소스 SW 공급망 공격 실행	MySQL	MySQL 서버의 기본 포트 통해 암호화되어 있지 않은 서버 확인 후 랜섬웨어 공격

2. 오픈소스 SW 관리체계

오픈소스 SW 는 각각의 라이선스를 지니고 있고[3] 소스코드를 누구나 사용할 수 있기 때문에, 무분별하게 사용 시 자신도 모르는 사이 저작권을 침해하거나 활용한 SW 가 보안 취약점에 노출될 수 있다. 따라서 오픈소스 SW 활용 전략을 수립하고, 이와 관련된 활동들을 적극적으로 관리하여 라이선스 위반과 보안 취약점으로부터 발생할 수 있는 리스크를 최소화해야 한다.

글로벌 ICT 기업은 오픈소스 SW 활용 정책과 프로세스를 OSP(Open Source Program)로 정의하고, 오픈소스 프로그램 매니저, 컴플라이언스 담당, 법무 담당, IT 담당의 협력체계로 구성된 OSPO(Open Source Program Office)라는 조직을 만들어 운영하고 있다[4]. OSPO 는 오픈소스 SW 의 조직 내 사용, 외부 프로젝트에의 기여, 조직 내 프로젝트의 소스코드 공개 등 오픈소스 SW 활용과 관련된 정책의 수립, 배포 및 이행을 담당한다[5].

2 https://blog.naver.com/skinfosec2000/221797384835

3 오픈소스 SW 라이선스들은 저작권 표시, 소스코드 배포, 라이선스 정보 제공 등의 의무사항들을 준수해야 하며 각각의 오픈소스 SW 라이선스 별로 요구사항들이 다른 탓에 개별적인 검토가 필요하다. 이를 통해 오픈소스 SW 라이선스 위반으로 인한 지적재산권 분쟁을 사전에 방지할 수 있다. 오픈소스 SW 라이선스에 대한 자세한 내용은 한국저작권위원회의 '오픈소스 소프트웨어 라이선스 가이드 3.0'(2016) 참조

4 TODO Group(talk openly develop openly)이 정의한 OSPO 는 오픈소스 SW 의 비즈니스적인 활용, 프로젝트 공개 및 외부 기여도 포함하고 있으므로 좀 더 포괄적인 인원 및 역할을 구성하고 있으며, 여기서는 라이선스 및 취약점 관리에 중점을 둔 구성과 역할을 중심으로 살펴본다. 전체적인 OSPO 는 https://todogroup.org/ 참조

5 국내 OSPO 운영 예시는 SK 텔레콤 참조(https://sktelecom.github.io/about/osrb/)

〈표 3〉 OSPO 인원 구성과 역할의 예

인원 구성	역할
오픈소스 프로그램 매니저	- 조직 내 오픈소스 SW 의 효과적 활용과 위험 제거를 위한 정책 수립, 프로세스 구축 및 운영
컴플라이언스 담당	- 외부 배포 SW 또는 서비스*를 대상으로, 여기에 포함된 오픈소스 SW 현황 파악 및 해당 라이선스 의무사항 준수
법무 담당	- 저작권 등 오픈소스 SW 라이선스와 관련된 법적 자문
IT 담당	- 도구 등을 활용하여 오픈소스 SW 컴플라이언스 및 보안취약점 점검

* 오픈소스 SW 라이선스는 외부 배포 SW 또는 서비스에 대해 적용 받고, 내부 사용 시에는 관계없음. 단, 내부 사용 시에도 보안 취약점에 대한 이슈는 내부 침입의 경로가 되므로 반드시 관리해야 함

상기 표의 인원 구성과 역할은 예시이므로 상황에 따라 외부 배포 SW 또는 서비스 기획·개발 단계에서의 라이선스 관리, 보안 취약점 점검 등의 핵심적 역할을 토대로 인원 구성과 역할을 변경할 수 있다. 중요한 점은 각 역할을 맡은 담당자들이 서로 협력하여 오픈소스 SW 사용으로 인한 위험(라이선스 위반, 보안 취약점)을 제거해야 한다는 점이다.

따라서 소규모의 조직이라도 오픈소스 SW 관리 및 운영 담당자를 선임하여 핵심적인 역할을 수행하는 것이 필요하며, 이외의 영역은 외부 지원을 받는 것이 효과적이다.

3. 오픈소스 SW 보안관리 방안

OSPO 인원 구성과 역할에서 IT 담당은 도구 활용 등의 방법으로 오픈소스 SW 컴플라이언스 및 보안 취약점 점검을 효과적으로 수행하는 것을 주업무로 한다. 보다 구체적으로, 도구를 이용하여 외부 배포 SW 또는 서비스에 사용된 오픈소스 SW 를 식별하는 것을 기반으로 컴플라이언스 담당자가 라이선스 의무사항을 준수해야 할 대상이 무엇인지 확인시켜 주고, 보안 취약점 진단 대상이 무엇인지 선별하여 진단 및 조치한다. 여기서 보안 담당자의 업무는 도구 활용 등의 방법으로 오픈소스 SW 보안취약점 점검을 수행하는 것으로 정의할 수 있다.[6]

오픈소스 SW 에 대한 보안 위협은 악성코드가 숨겨진 오픈소스 SW 의 배포, 공개된 소스코드의 취약점에 대한 공격, 보안 설정이 미흡한 오픈소스 SW 기반 서비스의 침투 등으로부터 발생하고 있다. 따라서 오픈소스 SW 를 안전하게 활용하기 위해서는 단순한 점검이 아닌 오픈소스 SW 의 도입, 운영, 관리 및 제거 등 활용 기간 전체에 걸쳐 체계적이고 지속적인 관리를 수행해야 한다. 보안 관점에서 안전한 오픈소스 SW 관리를 위한 활용 단계별 보안 고려사항은 다음 표와 같이 제시될 수 있다.

〈표 4〉 오픈소스 SW 활용 단계별 보안 고려사항*

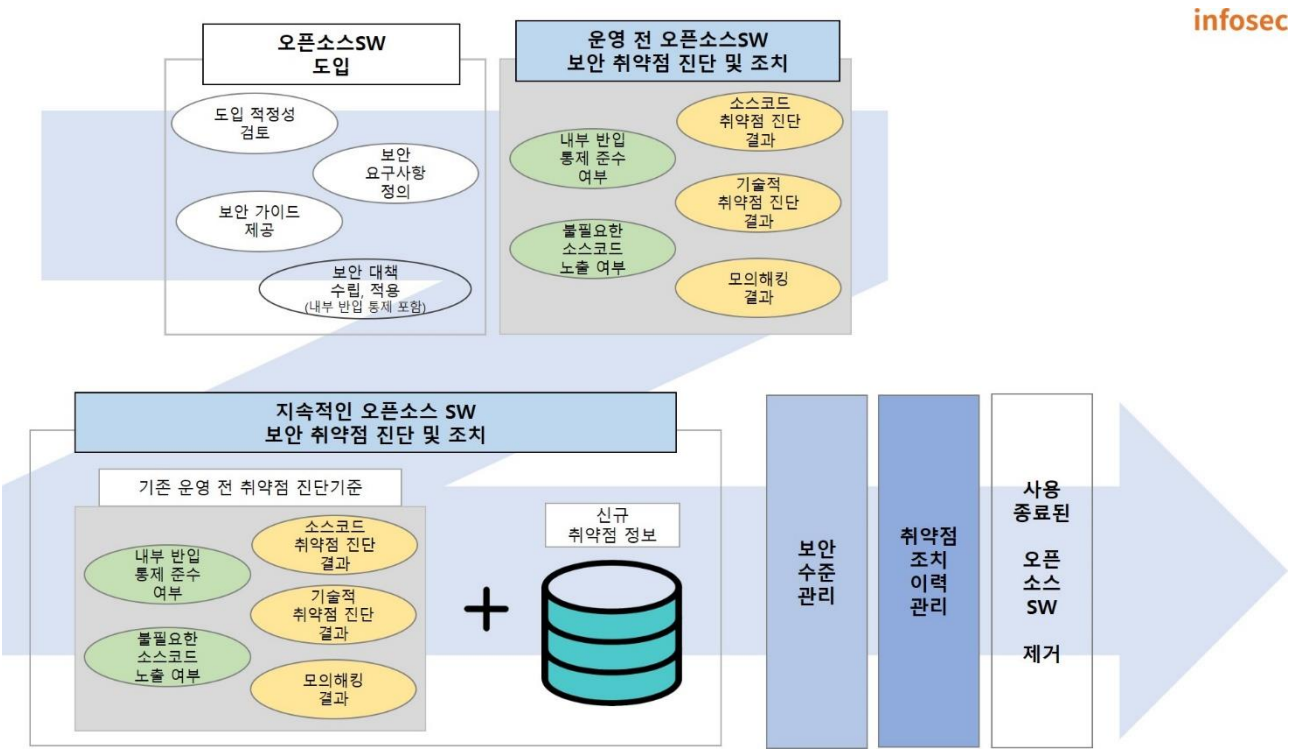
단계		보안 고려사항**
도입	개발	- 오픈소스 SW 보안성 검토
	운영 전 취약점 진단 및 조치	- 취약점 진단 항목 선별, 진단 및 조치
운영	지속적 취약점 진단 및 조치	- 오픈소스 SW 신규 취약점 정보 모니터링 결과 및 기존 취약점 현황 관리를 기반으로 주기적인 취약점 진단 및 조치 수행 - 오픈소스 SW 별 보안 수준, 취약점 진단 및 조치 결과 이력관리
관리 및 제거		- 오픈소스 SW 의 내부 반입 통제
		- 사용 종료된 오픈소스 SW 제거 및 해당 오픈소스 SW 의 신규 취약점 모니터링 종료

* 도입 – 활용 – 관리 – 폐기 등의 단계로 오픈소스 SW 활용 라이프사이클을 통합 관리하는 오픈소스 SW 거버넌스 측면에서의 단계 구분을 기준으로 함. 금융보안원(2022)은 개발 전, 개발 중 및 개발 후로 구분하기도 하며, 이는 여기서의 도입과 운영 단계로 포괄할 수 있음

** 여기서의 보안 관점은 개발, IT 자산 관리 단계에서의 고려사항 등은 포함하지 않음을 의미함

[6] 일반적인 IT 담당의 역할은 정보자산 관리, SW 라이선스 관리, 정보시스템 운영 및 보안관리 등으로 나누어 볼 수 있으며, 조직 규모에 따라 1 명 또는 여러 명이 업무를 나누어 수행하기도 한다. 이러한 관점에서 오픈소스 SW 에 대한 업무도 대상 식별, 컴플라이언스 담당자와의 협력, 보안 취약점 진단 등의 업무로 분류하고 1 명 또는 여러 명이 담당할 수도 있으나, 여기서는 보안 취약점 관리라는 한 분야만 세부적으로 살펴보기로 한다.

위 표에서 제시된 보안 고려사항들을 반영한 오픈소스 SW 보안관리 절차는 다음 그림과 같이 표현할 수 있으며, 이하에서는 각 단계별 수행 내용을 살펴보도록 한다.



[그림 1] 오픈소스 SW 보안관리 방안

🗁 도입: 개발 단계

개발자는 외부 배포 SW 또는 서비스 개발 시 필요한 기능을 갖춘 오픈소스 SW 를 선택하여 활용할 수 있다. 이때 개발자는 다음의 요소들을 검토해야 하며, 보안 이슈에 대해 보안 담당자의 자문을 받을 수 있다.

- ✓ 낮은 품질(보안이 고려되지 않은 코드)로 인한 보안취약점 발생 가능성

- ✓ 발생한 보안 취약점에 대한 조치 수행 여부
  - 오픈소스 SW 홈페이지, 커뮤니티 등에서 제시된 보안 이슈 대응 결과 확인
  - 가장 최근의 릴리즈가 오래되었을 경우 업데이트가 안되고 있을 수 있음

- ✓ 사용하려는 오픈소스 SW 의 소스코드 악성코드 검사 및 안전한 내부 반입 절차 이행
  - 내부 오픈소스 SW 저장소를 운영하여 악성코드 유입을 방지하고, 오픈소스 SW 사용 승인 등의 운영 관리, 보안 취약점 모니터링 대상 관리 등의 현황 관리에 활용

보안 담당자는 상기 요소들을 포함한 보안 요구사항을 정의, 개발 시 준수해야할 보안가이드를 제공하고, 보안 요구사항을 토대로 개발이 진행되고 있는지 확인하는 등의 보안성 검토 프로세스를[7] 수행하여, 도입하고자 하는 오픈소스 SW 로부터 발생 가능한 위험을 식별하고 보호대책을 적용한다.

프로세스의 마지막인 개발 단계에서 운영 단계로 이관하는 시기에는 최종 보안성 검토를 수행하여 보안 요구사항 충족 여부를 확인하고, 필요에 따라 다음의 '도입: 운영 전 취약점 진단 및 조치 단계'와 같이 소스코드 취약점 진단, 기술적 취약점 진단 및 모의해킹 등을 수행하여 미흡 사항 식별 및 개발자가 취약점 조치를 할 수 있도록 지원한다.

7 보안성 검토에 대한 상세 내용은 https://blog.naver.com/adtkorea77/222269193658 참조

🗁 도입: 운영 전 취약점 진단 및 조치 단계

보안 담당자는 개발에 사용된 오픈소스 SW 의 기능 및 알려진 보안 취약점 정보를 토대로 체크리스트를 작성하고, 취약점 진단 및 조치를 수행한다.

오픈소스 SW 의 보안위협 고려 시 안전한 내부 반입 절차 이행 여부, 소스코드 공개 시 상황도 포함해야 하며, 소스코드 취약점 진단 결과, 모의해킹 결과 그리고 계정 관리, 권한 관리, 데이터 관리, 감사/추적 관리 등 SW 기능[8]에 대한 취약점들도 확인해야 한다.

🗁 운영: 지속적 취약점 진단 및 조치 단계

오픈소스 SW 는 공개된 소스코드와 높은 활용도로 인해 공격받기 쉬우며 침해사고의 파급효과도 크다. 따라서, 주기적으로 신규 취약점 정보를 수집하고, 체크리스트를 업데이트하여 취약점 진단 및 조치를 수행해야 한다.

오픈소스 SW 의 신규 취약점 정보는 오픈소스 SW 홈페이지, 커뮤니티 등에 신규로 제기된 보안 이슈가 있는지, 공개된 취약점 DB 등에 신규 등록된 취약점 정보가 있는지 확인하여 수집한다.

✓ 오픈소스 SW 취약점 정보 확인의 예
- 공개 SW 포털(https://www.oss.kr/): 정보마당 〉 공개 SW 보안취약점에서 검색
- 취약점 DB 에서 검색
  → NVD[9](https://nvd.nist.gov/): Search 〉 공개 SW 명칭으로 검색
  → CVE[10](https://cve.mitre.org/): Search CVE List 〉 공개 SW 명칭으로 검색

이렇게 신규로 수집된 정보는 해당 오픈소스 SW 의 체크리스트에 반영하여 주기적인 취약점 진단 및 조치 수행으로 취약 여부를 확인한다. 단, 기존에 점검하던 항목들과 함께 주기적으로 취약 여부를 확인하는 정기 업무로 수행해도 되긴 하나, 긴급 조치를 요하는 경우에는 해당 항목만 즉시 확인 및 조치하여 위험에 선제 대응하는 것도 효과적인 방법이다.

8 오픈소스 SW 별 보안 설정에 관련된 취약점 진단 및 조치 상세내용들은 EQST 의 '클라우드 보안 가이드(컨테이너 보안) – Docker, Kubernetes(2019)', '오픈 소스 소프트웨어 보안 가이드(2018)' 참조

9 美 국립과학기술표준연구소(NIST: National Institute of Science and Technology)의 취약점 데이터베이스(National Vulnerability Database)

10 마이터(MITRE, 美 정부가 지원하는 연구개발 단체)의 취약점 데이터베이스(Common Vulnerabilities and Exposures)

보다 효율적인 취약점 정보 수집, 진단, 사후관리를 수행하기 위해서는 자동화 도구를 잘 활용해야 한다. 자동화 도구들은 사용된 오픈소스 SW 들을 식별하고, 보안 취약점을 진단하여 발견 시 알림을 제공한다[11]. 또 필요한 취약점 정보만을 수집하는 크롤러(crawler)나 진단할 취약점 항목만을 대상으로 한 스크립트(script)를 제작해 활용하는 것도 좋은 방법이다. 이러한 자동화된 방법들을 자신의 조직에 적합하게 도입 단계부터 활용하여, 조직 내 오픈소스 SW 사용현황을 파악하고, 취약점의 사전 제거 및 위험을 최소화하는 것이 안전한 오픈소스 SW 사용환경을 만들어가는 보안관리 방안이다.

오픈소스 SW 는 지속적으로 보안·관리가 수행되어야 높은 보안수준을 달성 및 유지할 수 있다. 발견된 취약점이 조치되지 않아 보안 구멍(Security Hole)으로 잔류하는 일이 없도록, 주기적으로 조치 계획이 제대로 수행되었는지를 확인하여 미조치 사항을 제로화(0)하고 동일 취약점이 재발하지 않도록 해야 한다. 또한, 보안수준을 정량적으로 평가하여 목표 대비 어느 수준에 머물러 있는지 파악 후 목표 달성을 위한 노력도 해야 한다. 오픈소스 SW 를 안심하고 활용하기 위해서는 이러한 지속적인 관리가 중요하다.

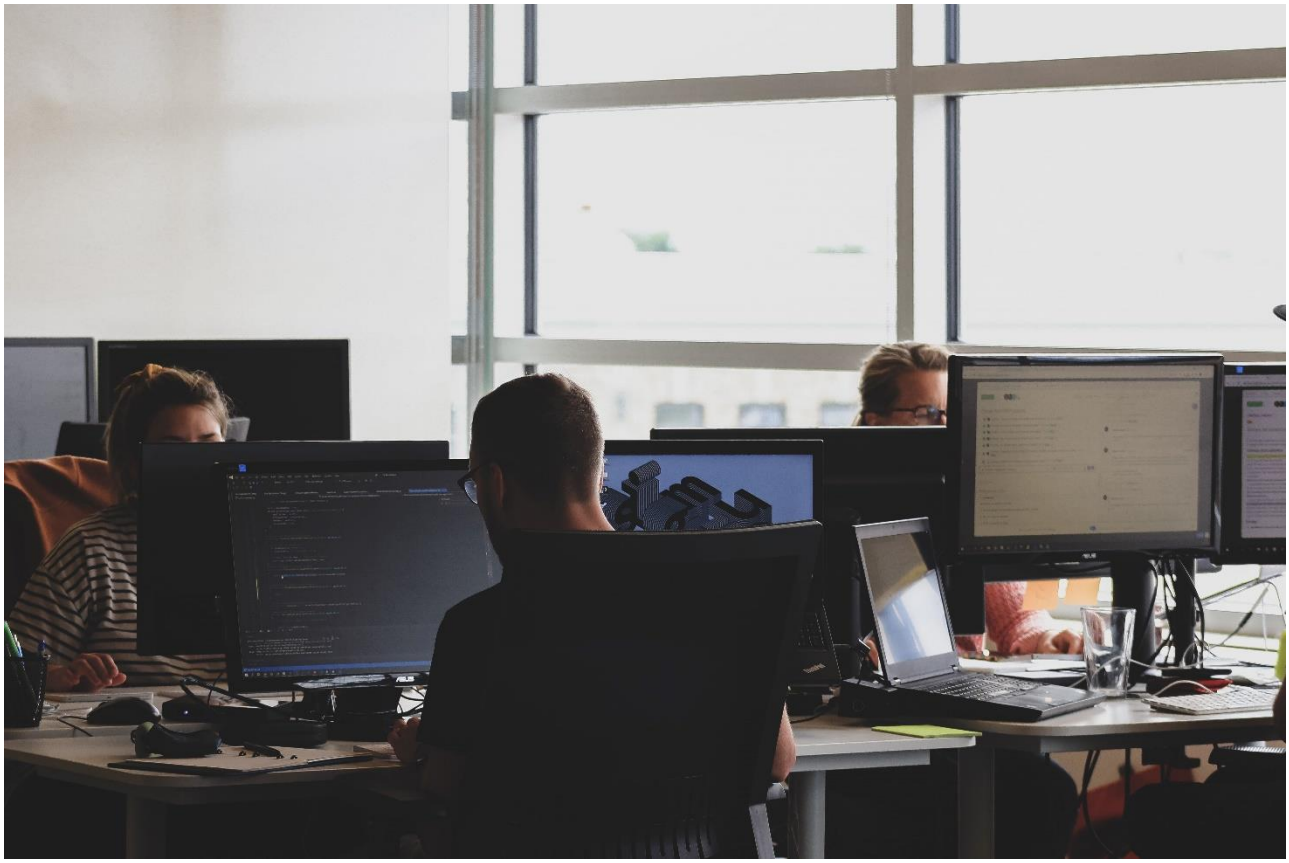
🗁 관리 및 제거 단계

내부의 오픈소스 SW 사용현황 관리, 비인가 및 악성코드가 숨겨진 오픈소스 SW 반입 차단을 위해서는 망연계장치 등을 이용한 인가된 방법으로 내부 반입 여부를 확인하거나, 내부에 별도로 구성한 오픈소스 SW 저장소를 이용해 반입 및 배포 여부를 확인하는 등 내부 통제 방안을 수립하고 실행해야 한다.

마지막으로 제거 단계에서는 외부 배포 또는 서비스가 종료되어 여기에 포함된 오픈소스 SW 에 대한 관리가 불필요하게 되었을 경우, 해당 오픈소스 SW 의 제거를 확인해 사용하지 않는 오픈소스 SW 로부터의 위협 발생 상황을 사전에 차단해야 한다.

11 자동화 도구들의 종류와 기능들은 주요 기관에서 발행한 아래와 같은 문서 등을 참조
금융보안원, 금융분야 오픈소스 소프트웨어 활용관리 안내서, 2022. 12.
정보통신산업진흥원, 기업 공개소프트웨어 거버넌스 가이드, 2021

4. 맺음말



이상과 같이 안전한 오픈소스 SW 사용환경을 만들기 위한 보안관리 방안을 살펴보았다. 오픈소스 SW 는 전세계적인 디지털 전환 흐름과 함께 활용성이 높아지고 있어 관심 받고 있으나, 보안 관련 전담조직이나 대책과 같은 위협에 대한 대응은 미약한 실정이다. 더욱이 여러 상용 SW 및 서비스에도 오픈소스 SW 가 활용되고 있어, 이처럼 보안이 취약한 상황에서 침해사고가 발생할 경우, 거대한 파급효과가 발생하게 된다.

따라서 조직 내 오픈소스 SW 의 안전한 사용환경을 만들어 가기 위해서는 보안 담당자 뿐만 아니라 개발자, 서비스 운영자도 함께 협력하여 오픈소스 SW 의 도입, 운영, 관리 및 제거를 아우르는 활용 기간 전체에 걸친 지속적인 보안관리 방안을 내부 환경에 적합하게 적용하고 계속 발전시켜 나가야 한다.