KISA 한국인터넷진흥원
KOREA INTERNET & SECURITY AGENCY

2021 KISA REPORT

volume 12

www.kisa.or.kr

CONTENTS

주제 제안 및 정기 메일 신청 | kisareport@kisa.or.kr

인터넷 정보보호 관련 이슈, 현안 등 궁금한 내용을 보내주시면 선별 후 보고서 주제로 선정됩니다.

또한, KISA Report 온라인 서비스 제공을 원하실 경우 신청해주시면 매월 받아보실 수 있습니다.

소프트웨어 보안 관점에서 본 미국 사이버보안 행정명령과 우리의 대응방안

이태준 (tjlee@korea.ac.kr)

고려대학교 컴퓨터보안연구실 연구원

* 공동저자:
이희조 (고려대학교 교수, heejo@korea.ac.kr)
박춘식 (고려대학교 연구교수, cspark14@hanmail.net)

제1장 서론

2021년 5월 12일, 미국 바이든 행정부는 『국가의 사이버보안 향상에 관한 행정명령(EO 14028) (2021)』을 내렸다. 이는 최근 콜로니얼 파이프라인 공격이나 솔라윈즈 공급망 공격과 같은 **대규모 사이버 공격 사례가 등장하여 사이버보안의 필요성이 증가**했기 때문이다. 이러한 악성 사이버 공격이 공공부문, 민간 부문을 넘어 개개인의 보안과 프라이버시까지 위협한다고 미국 정부는 판단한 것이다.

우리나라 또한 소프트웨어 보안 문제에서 자유롭지 않다. 소프트웨어에 국경이 없는 만큼 해외소프트웨어의 취약점이 국내에 영향을 끼칠 수 있으며 반대의 경우도 마찬가지이다. 소프트웨어 공급망을 통해 전세계가 연결되어 있기 때문에 **미국의 행정명령을 통해 나온 규제에서 국내 기업도 자유로울 수 없는 것**이다.

많은 국내 소프트웨어 기업들은 소프트웨어 제품의 구현을 중심으로 하는 프로세스를 가지고 있다. 근래의 사이버보안 요구사항은 소프트웨어 개발부터 유통까지 공급망의 모든 과정을 대상으로 하기 때문에 이에 대비하기 위해서는 많은 비용과 시간을 들여 근본적인 체질 개선을 해야 한다. 이번 행정명령과 같은 **변화를 미리 파악하고 대비하지 않으면 이후에 더 큰 비용**으로 돌아올 수 있다.

본 문서는 소프트웨어 제품을 개발하는 기업에서 행정명령의 요구사항에 맞추기 위해 무엇을 준비해야 할지 살펴보기 위해 작성되었다. 행정명령 중 소프트웨어 보안에 대한 부분인 4절의 내용과 연관된 문서들, 그리고 국내의 소프트웨어 보안 관련 정책을 분석하여 앞으로 **국내 기업에서 소프트웨어 제품 보안 요구사항에 어떻게 대비해야 할지 점검**하고자 한다.

제2장 미국 사이버보안 행정명령

이번 행정명령은 앞으로의 사이버보안 위협에 맞서기 위해 시행되는 조치로 총 11절로 구성되어 있다.

- 1절 방침(Policy)
  - ■ 행정명령의 배경과 목적에 대해 설명
- 2절 위협 정보 공유 과정의 장애물 제거
- 3절 연방 정부 사이버보안의 현대화
- 4절 소프트웨어 공급망 보안의 강화
- 5절 사이버안전심의회(Cyber Safety Review Board)의 설치
- 6절 연방 정부의 사이버보안 취약점과 사고 대응 전략 표준화
- 7절 연방 정부 네트워크에서 사이버보안 취약점과 사고 검출 개선
- 8절 연방 정부의 조사 및 복구 능력 향상
- 9절 국가안보시스템
- 10절 용어 정의
- 11절 일반적인 규정(General Provisions)

행정명령의 1절에서는 **연방 정부와 민간 부문의 협력을 강조**하고 있다. 안전한 사이버 공간을 만들기 위해 연방 정부는 사이버 공격을 방어, 탐지 및 대응하기 위한 노력을, 민간 기업은 끊임없이 변화하는 위협 환경에 적응하며 자사 제품이 안전하게 구축 및 운용되고 있는지 확인해야 한다는 것이다. 최종적으로 디지털 인프라에 대한 신뢰를 구축하여 국가 및 경제의 안전을 보장하는 것을 목표로 한다.

행정명령 4절 '소프트웨어 공급망 보안의 강화'는 소프트웨어 개발 및 유통 과정에서의 보안 강화에 대

해 다룬다. 내용을 보면 보안 강화를 위한 지침 발행, 가이드라인 공표 등 어떤 기관이 언제까지 무엇을 수행해야 하는지를 지시하는데, 그 안에서 **소프트웨어 제품 개발 기업에 요구되는 사항**들을 볼 수 있다.

제3장 소프트웨어 공급망 보안의 강화 (행정명령 4절)

행정명령 4절은 총 24항으로 구성되어 있으며, 각 항은 어떤 기관이 언제까지 무엇을 해야 하는지를 지시한다. NIST 등 각 부처와 기관은 행정명령의 내용에 따라 보안 강화를 위한 지침 발행, 주요 소프트웨어의 정의 공표 및 보안 지침 발행, 소프트웨어 라벨링 등 필요한 조치를 수행해야 한다. 각 항의 세부 내용과 기한은 Appendix A를 통해 확인할 수 있다.

[그림 1]은 행정명령 4절 중 소프트웨어 제품을 개발하는 기업에서 대응해야 할 항목들을 시간 순으로 정리한 것이다. 크게 주요 소프트웨어 관련, SBOM 관련, 소프트웨어 테스트 관련, 가이드라인 및 지침 관련으로 나눌 수 있다.

[표 1] 소프트웨어 제품 개발 기업 관점 타임라인

가) '주요 소프트웨어(critical software)' 관련

'주요 소프트웨어'는 연방 정부에서 사용하는 주요 소프트웨어 제품에 대한 보안 기준을 개발하기 위해 행정명령에서 도입한 개념으로, 일반적인 의미의 주요 소프트웨어와 구분하기 위해 EO-주요(EO-Critical) 소프트웨어라 한다. **EO-주요 소프트웨어로 지정된 소프트웨어는 연방 정부가 소프트웨어를 구매 및 관리하는 방법을 포함하여 추가적인 활동**을 필요로 한다. 따라서 서비스하고 있는 소프트웨어 제품이 EO-주요 소프트웨어에 해당되는지 알아보고, EO-주요 소프트웨어를 위한 보안 조치를 적용해야 한다.

NIST에서 공표한 EO-주요 소프트웨어의 정의에 따르면, EO-주요 소프트웨어란 다음 속성들 중 하나 이상을 포함하는 소프트웨어 혹은 그러한 소프트웨어를 직접적으로 포함하는(direct dependencies) 소

프트웨어를 의미한다.

- 상승된 권한으로 작동되거나 권한을 관리하는 소프트웨어
- 네트워크 및 컴퓨팅 자원에 직접적으로, 혹은 권한을 부여받아 접근할 수 있는 소프트웨어
- 운용 기술이나 데이터로의 접근을 제어하기 위해 설계된 소프트웨어
- 신뢰와 직결되는 기능을 수행하는 소프트웨어
- 권한 있는 접근(Priviliged Access)으로 신뢰 경계 밖에서 운용되는 소프트웨어

NIST는 소프트웨어 시장의 규모, 범위 및 복잡성과 정부 내에서 필요한 인프라를 감안하여 EO-주요 소프트웨어의 공급망 보안 강화를 단계적으로 접근하기로 하였다. 행정명령을 처음 적용하는 단계에서는 스탠드얼론 및 온프레미스 소프트웨어를 중심으로 하고, 그 이후에 클라우드 기반 소프트웨어나 펌웨어, OT 분야의 소프트웨어 등을 포함시키는 것을 권장하였다. 첫 적용 범위에 해당되는 소프트웨어의 예비 리스트를 보면 **운영체제, 웹 브라우저, 방화벽, 원격 취약점 검사 소프트웨어 등 광범위한 소프트웨어가 EO-주요 소프트웨어에 해당**된다.

배포되는 방식과 무관하게 EO-주요 소프트웨어에 정의에 해당되는 제품은 모두 EO-주요 소프트웨어이다. 오픈소스 소프트웨어 또한 예외가 아니다. EO-주요 소프트웨어의 정의에 해당되는 오픈소스 소프트웨어를 컴포넌트로 사용하는 경우 이를 개발 프로세스에 포함시켜 요구사항을 준수해야 한다. 임베디드 소프트웨어나 펌웨어 또한 EO-주요 소프트웨어에 해당될 수 있다.

EO-주요 소프트웨어에 해당되는 소프트웨어 제품을 개발하는 기업은 **EO-주요 소프트웨어에 대한 보안 조치를 제품에 적용**해야 한다. 보안 조치에 대한 지침은 기존에 발행된 여러 보안 관련 문서를 기반으로 하는데, 그 중 NIST의 Cybersecurity Framework와 SP 800-53 Revision 5, Security and Privacy Controls for Information Systems and Organizations를 우선적으로 참조하여 만들어졌다.

지침은 다섯 가지 목표와 그에 맞는 조치로 구성되어 있으며, 간략한 내용은 다음과 같다.

1. EO-주요 소프트웨어를 인증되지 않은 접근 및 사용으로부터 보호

(1) 다중 요소 인증 사용
(2) EO-주요 소프트웨어 및 플랫폼에 접근하는 각 서비스를 고유하게 식별 및 인증
(3) 네트워크 기반 관리를 위한 권한 있는 접근(privileged access) 관리 원칙 준수
(4) 적절한 경계(boundary) 보호 기법 사용

2. EO-주요 소프트웨어에서 사용되는 데이터의 CIA 보호

(1) 데이터 인벤토리 설정 및 유지 관리
(2) 데이터 및 자원에 대한 세분화된 접근 제어 사용

(3) 민감한 데이터에 대한 암호화를 통해 저장된 데이터 보호

(4) 상호 인증 및 민감한 데이터 통신 암호화를 통해 전송 중인 데이터 보호

(5) 데이터 백업, 백업 복구 실행 및 데이터 복구 준비

3. EO-주요 소프트웨어 플랫폼과 해당 플랫폼에 배포된 소프트웨어를 식별하고 관리하여 악용 방지

(1) 소프트웨어 인벤토리 설정 및 유지 관리

(2) 패치 관리 방법 적용

(3) 구성 관리 방법 적용

4. EO-주요 소프트웨어와 관련된 위협과 사고에 대한 빠른 발견, 대응, 회복

(1) 보안 관련 이벤트를 기록하기 위한 로깅 설정

(2) 지속적인 보안 모니터링

(3) 엔드포인트 보안 보호 기법 적용

(4) 네트워크 보안 보호 기법 적용

(5) 모든 보안 운용 담당자 및 사고 대응 팀 구성원 교육

5. EO-주요 소프트웨어의 보안을 담당하는 사람의 이해와 능력을 강화

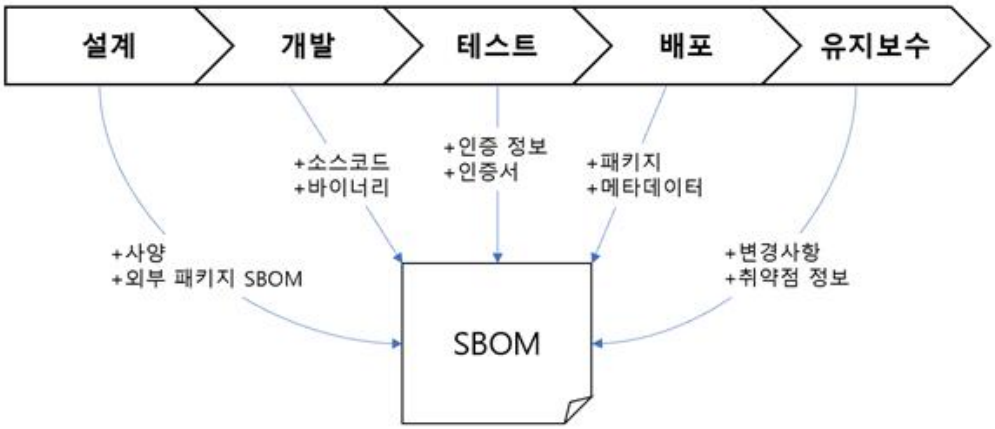
(1) 모든 EO-주요 소프트웨어 사용자 교육

(2) 모든 EO-주요 소프트웨어 및 플랫폼 관리자 교육

(3) 보안 인식 활동 자주 수행

나) SBOM 관련

SBOM은 Software Bill of Materials의 준말로 **소프트웨어를 빌드하기 위해 사용된 여러 컴포넌트에 대한 내용과 공급망 관계를 형식에 맞춰 기록하는 것**을 의미한다. 식품의 성분표와 비슷한 역할을 한다고 볼 수 있다. SBOM은 소프트웨어 제품을 생산, 소비, 운용하는 자가 공급망에 대해 더 깊이 이해하고, 알려진 혹은 새로운 취약점과 위협을 추적할 수 있게 한다. 특히 알려진 취약점의 확산을 저지하는 데 효과적이다.



[그림 1] 소프트웨어 개발 주기에 따른 SBOM 정보

SBOM은 소프트웨어가 어떤 패키지를 포함하는지, 어떤 코드와 파일로 구성이 되는지, 어떤 인증을 받았는지 등 소프트웨어에 대한 자세한 정보를 포함해야 한다. [그림 1]은 소프트웨어가 개발되는 주기의 각 단계가 SBOM에 어떻게 반영되는지를 나타낸 그림이다. SBOM은 소프트웨어를 설계하는 단계부터 배포되기까지 고려되어야 하며, 배포 이후 유지보수 단계에서도 소프트웨어의 변경사항과 취약점 등 위험 관련 정보가 지속적으로 반영, 배포되어야 한다.

행정명령에 의해 NTIA에서 SBOM의 최소 요소를 공표하였다. SBOM의 최소 요소란 취약점 관리, 라이선스 등 기본적인 SBOM의 활용에 필요한 최소 요소를 의미한다. 공표된 최소 요소는 크게 세 부분으로 필수적인 항목, 자동화, 관행과 절차로 구성되어 있다. 필수적인 항목은 SBOM이 어떤 정보를 담아야 하는지를, 자동화는 SBOM 생성이 자동화되어야 한다는 것을, 관행과 절차는 SBOM을 어떻게 관리하고 운영해야 하는지를 담고 있다.

필수적인 항목의 경우 각 컴포넌트에 대해 추적하고 관리해야 하는 기본 정보를 지정한다. 기본 정보는 컴포넌트 공급자명, 컴포넌트명, 컴포넌트 버전, 컴포넌트 식별을 위한 기타 고유 식별자, 의존 관계, SBOM 데이터 작성자, 타임스탬프로 총 7가지 항목으로 구성되었다.

자동화의 경우 SBOM을 생성하고 처리하기 위한 **데이터 형식을 SPDX, CycloneDX, SWID의 세 가지로 제한**하고 있다. 리눅스 재단의 SPDX는 Software Package Data Exchange의 준말로, 오픈소스 라이선스 관리를 중심으로 컴포넌트 관련 정보를 교환할 수 있는 SBOM 표준이다. OWASP의 CycloneDX는 오픈소스 컴포넌트에 대한 취약점 식별, 라이선스 관리 등을 위해 개발된 보안 취약점 중심의 SBOM 표준이다. NIST의 SWID는 Software Identification의 준말로, 소프트웨어 수명 주기 전반에 걸쳐 소프트웨어 인벤토리 자동화 및 관리를 하기 위해 개발된 SBOM 표준이다.

*P1, P2, P3는 소프트웨어에 포함된 패키지를 의미함

[그림 2] SBOM 자동 생성 과정

예시로 SPDX는 SBOM Generator를 제공하여 SBOM을 자동 생성할 수 있게 한다. [그림 2]는 패키지 P1, P2, P3를 포함하고 있는 소프트웨어에 SBOM Generator를 사용했을 때 SBOM 파일이 생성되는 과정을 나타낸 것이다. SBOM Generator는 Composer, Maven, NPM, PIP 등의 패키지 매니저를 지원하며 입력된 소프트웨어가 어떤 하위 패키지를 포함하고 있는지 정보를 추출하여 이를 SBOM으로 만든다.

```
SPDXVersion: SPDX-2.2
DataLicense: CC0-1.0
SPDXID: SPDXRef-DOCUMENT
DocumentName: spdx-sbom-generator
DocumentNamespace: http://spdx.org/spdxpackages/spdx-sbom-generator--57918521-3212-4369-a8ed-3d681ec1d7a1
Creator: Tool: spdx-sbom-generator-XXXXX
Created: 2021-05-23 11:25:29.1672276 -0400 -04 m=+0.538283001

##### Package representing the Go distribution

PackageName: go
SPDXID: SPDXRef-Package-go
PackageVersion: v0.46.3
PackageSupplier: NOASSERTION
PackageDownloadLocation: pkg:golang/cloud.google.com/go@v0.46.3
FilesAnalyzed: false
PackageChecksum: TEST: SHA-1 224ffa55932c22cef869e85aa33e2ada43f0fb8d
PackageHomePage: pkg:golang/cloud.google.com/go@v0.46.3
PackageLicenseConcluded: NOASSERTION
PackageLicenseDeclared: NOASSERTION
PackageCopyrightText: NOASSERTION
PackageLicenseComments: NOASSERTION
PackageComment: NOASSERTION

Relationship: SPDXRef-DOCUMENT DESCRIBES SPDXRef-Package-go
```

[그림 3] SPDX SBOM Generator 사용 결과 예시

[그림 3]은 SBOM Generator를 통해 생성된 SPDX 형식의 실제 SBOM 파일 예시이다. SPDX의

SBOM은 spdx와 json 형식으로 사용할 수 있다. 그림을 자세히 보면 사용된 SPDX의 버전, 라이선스, 포함된 패키지의 이름과 버전 등 자세한 정보가 담겨있다. SPDX 포맷은 문서 생성 정보, 패키지 정보, 파일 정보, 스니펫 정보, SPDX 외 라이선스 정보, (SPDX 문서 간) 관계 정보, 주석으로 이루어지는데, 이를 알려진 취약점을 발견하거나 라이선스를 관리하기 위해 활용할 수 있다.

다) 소프트웨어 테스트 관련

소프트웨어의 제품이 안전하다는 것을 검증하기 위해서는 소프트웨어 테스트가 불가결하다. 현재 사용되는 테스트 방법은 위협 모델링, 자동화 검사, 정적 및 동적 분석 등으로 다양하기 때문에 어느 정도의 테스트를 거쳐야 소프트웨어가 충분히 안전한지 판단하기가 어렵다. 이번 행정명령에서는 **소프트웨어 검증에 권장되는 최소 기준을 마련**하여 소프트웨어 생산자가 자체적인 검증 프로세스를 만들 수 있도록 돕고자 한다.

행정명령으로부터 60일 이내, NIST에서 소스 코드 테스트의 최저 기준에 대한 가이드라인을 공표하게 된다. 자발적인 가이드라인이기 때문에 의무적으로 준수할 필요는 없지만 추후 의무 사항이 될 가능성이 있으므로 미리 대비해두는 것이 좋을 것이다. 공표된 테스트 종류에는 위협 모델링, 자동화 검사, 코드 정적 분석, 동적 분석, 구성요소 검사, 버그 수정이 있다.

테스트 종류	설명
위협 모델링	시스템을 추상화하여 잠재적인 공격자와 그들의 목표 및 방법, 잠재적인 위협을 설계 단계에서 미리 발견하는 방법
자동화 검사	소프트웨어 테스트를 자동화하여 워크플로우나 이슈 추적 과정에 통합하는 방법
정적 분석	코드 스캐너를 사용하여 버그를 발견하거나 하드코딩된 비밀 정보가 있는지 검토하는 방법
동적 분석	블랙박스 테스팅을 하거나 퍼징, 웹앱 스캐너를 사용하는 등 프로그램이 실행되는 동안 분석하는 방법
구성요소 검사	소프트웨어에 포함된 라이브러리, 패키지 등에 존재하는 알려진 취약점을 탐지하는 방법
버그 수정	발견된 치명적인 버그를 최대한 빨리 수정하는 방법

위협 모델링이란 시스템을 분석하여 잠재적인 위협을 식별하는 것을 의미한다. 자동화 검사는 자동화된 검사를 지속적으로 수행하는 것이다. 정적 분석은 소스코드를 직접 분석하는 것이고, 동적 분석은 코드가 실행되는 환경에서 분석하는 것이다. 구성요소 검사는 소프트웨어에 포함된 라이브러리, 패키지, 서비스 등 컴포넌트가 가지고 있는 알려진 취약점을 검사하는 것이다. 버그 수정은 치명적인 버그를 최대

한 빨리 수정하는 것을 의미한다. 각 테스트의 세부 내용은 Appendix B를 통해 확인할 수 있다.

라) 가이드라인 및 지침 관련

행정명령 4절의 가장 핵심적인 산출물은 공급망 보안 강화를 위한 가이드라인과 실행 지침이다. 가이드라인은 4절의 요구사항들을 만족시킬 수 있게끔 만들어지고, 지침은 가이드라인을 기반으로 만들어진다. 지침에는 안전한 소프트웨어 개발 환경, 코드의 완전성을 확보하기 위한 자동화 도구 도입, SBOM 제공, 안전한 소프트웨어 개발 방법에 대한 적합성 증명 등 공급망 보안을 강화하기 위한 내용이 포함된다.

예비 가이드라인과 관련된 내용의 경우 NIST에서 개발 중인 SP 800-161 Rev. 1 (2nd Draft)의 Appendix F에 수록되었다. SP 800-161은 시스템과 조직을 위한 사이버보안 공급망 위험 관리 실행 지침에 대한 문서이다. SP 800-161 문서의 Appendix F의 내용을 보면 행정명령을 통해 나온 '주요 소프트웨어의 정의' 등 산출물들의 내용에 대한 정리와 그 내용이 SP 800-161과 어떻게 연결이 되어있는지 확인할 수 있다.

실행 지침의 경우 기존에 NIST에서 개발한 SSDF (Secure Software Development Framework)를 행정명령의 요구사항에 맞게 업데이트하는 것으로 대신한다. SSDF는 SDLC에 통합할 수 있는 고수준의 안전한 소프트웨어 개발 지침 모음으로 총 43개의 보안 조치로 구성되어 있다. 보안 조치는 조직에서 준비해야 하는 조치, 소프트웨어 보호를 위한 조치, 안전한 소프트웨어 생산을 위한 조치, 취약점 대응을 위한 조치의 네 종류로 구성되어 있다. 행정명령 내용을 반영한 SSDF 1.1 버전이 배포되었으며, 행정명령의 요구사항과의 연결은 SSDF 문서의 Appendix A에 수록되어있다.

제4장 우리나라의 대응방안

국내의 경우 2021년 2월 발표된 'K-사이버방역 추진전략'에서 공급망 보안을 강화하기 위한 정책을 제시한 바 있다. 해당 정책에 따르면 주요 SW업체에 단계별 보안진단 및 조치 지원, 중소 SW업체에 자가 보안 진단 도구 및 가이드 보급, 주요 SW서비스 기업에 개발환경 보안점검 지원 및 공급망 검증 체계 구축, 공급망 보안점검 기준 및 점검도구 개발이 이루어진다고 한다.

위 내용에 따라 올해 8월 과학기술정보통신부 주관으로 '소프트웨어 개발 보안 허브'가 개소된 바 있다. 중소기업에 소프트웨어 개발보안을 적용하여 공급망 공격을 예방하겠다는 것이다. 소프트웨어 개발 보안 활성화를 위해 보안약점 진단, 개발자 대상 교육 등의 사업이 추진된다. 아쉬운 점은 보안약점 진단이 정적분석에 한정되어 있고, **국내 개발보안이 시큐어코딩으로 한정**된 경우가 많다는 것이다.

[표 2] 미 사이버보안 행정명령과 K-사이버방역 추진전략 비교

사이버보안 행정명령	내용	K-사이버방역 추진전략
'주요 소프트웨어' 대상 보안 조치 지침 발행,	**보안 조치**	다중이용공공서비스 분야 주요 SW 업체 대상 단계별 보안진단·조치 지원
SBOM 최소 요소 공표	**SBOM**	해당 없음
소프트웨어 검증 최소 기준 공표	**소프트웨어 테스트**	중소 SW 업체 대상 자가 보안 진단 도구 및 가이드 보급
공급망 보안 강화를 위한 가이드라인 및 지침 발행	**가이드라인 및 지침**	융합산업 분야별 보안 가이드라인 및 표준 보안모델 마련

[표 2]를 보면 미 사이버보안 행정명령과 K-사이버방역 추진전략이 공급망 보안 문제에 접근하는 방식의 차이를 볼 수 있다. 행정명령은 공급망 보안에 필요한 부분을 체계적으로 문서화하여 기업에서 직접 적용할 수 있게끔 한다. K-사이버방역 추진전략은 정부기관의 개입을 통해 주요 SW업체나 중소 SW업체를 대상으로 보안 관련 지원을 한다. 즉 **행정명령이 기업의 근본적인 체질 개선을 통해 보안을 강화하려 한다면 K-사이버방역 추진전략은 대응 체계 강화를 통해 보안을 강화**하려 한다는 것이다.

행정명령을 통해 연방 정부에 납품되는 모든 소프트웨어와 그 컴포넌트의 보안에 요구사항이 생기는 만큼 국내 기업에 영향이 존재할 수밖에 없고, 현재 국내의 보안 정책만으로는 새롭게 만들어지는 국제적 보안 기준을 만족하기 어려운 것이 사실이다. 또한 공급망 보안은 최종 소프트웨어가 만들어지는 모든 과정에 대한 보안을 의미하기 때문에 국제적 요구사항에 직면했을 때 반영하기에는 시간이 부족할 수 있다. 따라서 행정명령에 따라 공표되는 **가이드라인, 지침 등을 잘 파악하여 미리 국제적 요구사항에 맞춰 준비하는 것이 필수적**이라고 볼 수 있다.

가) '주요 소프트웨어(critical software)' 관련

우선 서비스 중이거나 개발 중인 소프트웨어 제품이 **EO-주요 소프트웨어에 해당되는지 확인**해야 한다. 공표된 EO-주요 소프트웨어의 정의가 광범위한 종류의 소프트웨어를 포함하고 중요한 소프트웨어를 직접적으로 포함하고 있는 소프트웨어도 EO-주요 소프트웨어가 되는 만큼, 소프트웨어 제품이 EO-주요 소프트웨어에 해당될 가능성이 높다고 볼 수 있다. 만약 **제품이 EO-주요 소프트웨어에 해당된다면 EO-주요 소프트웨어를 위한 보안 조치를 제품에 적용**해야 한다.

NIST에서 제시한 EO-주요 소프트웨어의 예시에 의하면 인증 관련 소프트웨어, 운영체제, 브라우저, 보안 관련 소프트웨어, 네트워크 관련 소프트웨어 등이 EO-주요 소프트웨어에 해당될 수 있다. 물론 미국 정부에 납품하거나 납품하는 소프트웨어에 관련될 일이 없다면 이를 EO-주요 소프트웨어에 해당되지 않으므로 EO-주요 소프트웨어를 위한 보안 조치를 꼭 적용하지 않아도 된다.

나) SBOM 관련

그리고 SBOM을 도입해야 한다. SBOM은 알려진 취약점을 제거할 뿐 아니라 라이선스 리스크를 방지하는 데에도 도움이 된다. 안전한 소프트웨어 생태계 조성을 위해서도 SBOM은 필수적이다. SBOM의 최소 요소를 공표하면서 사용 가능한 SBOM의 데이터 형식이 **SPDX, CycloneDX, SWID tags의 세 가지**로 명시되었으므로 이 중 **하나를 채택하는 것으로부터 SBOM 생성 및 관리를 시작**할 수 있을 것이다.

다) 소프트웨어 테스트 관련

소프트웨어 제품을 적절히 테스트하는 것도 중요한 부분이다. 공표된 소프트웨어 테스트 최소 기준에서는 총 12가지 방식의 테스트 방식을 제안하고 있는데, 기업마다 보유한 보안 역량에 차이가 있으므로 모든 방법을 적용하기 어려울 수 있다. 그런 경우 **자동화 검사나 코드 스캐너, 구성요소 검사와 같이 비교적 적용하기 용이한 방법부터 단계적으로 적용**해보는 방식으로 접근할 수 있을 것이다. 참고로 최소 기준 가이드라인 문서를 보면 각 방법과 관련된 보충 자료가 있으므로 이를 참고해볼 수 있다.

라) 가이드라인 및 지침 관련

무엇보다도 공급망 보안 강화를 위한 **가이드라인과 실행 지침을 준수**해야 한다. NIST의 SSDF를우선적으로 적용할 수 있다. 개발 환경을 안전하게 만들어야 하고, 코드의 무결성 유지와 취약점 발견을 위한 자동화 도구를 도입해야 하며, SBOM을 제공해야 하고, 취약점 공개 프로그램에 참여해야 하는 등 행정명령의 여러 요구 사항들이 SSDF에 포함되었다. 문서에서 제안하고 있는 보안 조치가 43개 정도로 많기 때문에 **어떤 조치가 이미 적용되어 있는지, 어떤 조치부터 적용하면 좋을지 분석하여 단계적으로 적용**하는 것을 권한다.

제5장 결론

이번 사이버보안 행정명령이 여러 방면으로 많은 내용을 담고 있지만, 자세히 살펴보면 기존에 이미 존재하던 관행, 기준, 표준을 활용하여 재구성한 부분이 많다. 다만 기존에 **권장 사항 정도로 존재하던 개념들이 행정명령을 통해 필수적으로 요구되는 사항이 될 수 있다**는 점에서 의의가 있다고 볼 수 있다.

국내의 경우 이번 행정명령의 요구사항을 100% 만족하고 있는 기업은 많지 않을 것으로 예상이 되고 있는 상황이다. 미국에서 나온 행정명령이지만, 소프트웨어의 국경이 모호한 시대인 만큼 **국내 기업에도 영향**을 끼칠 것임은 자명한 일이다. 다행히 행정명령이 발령된 지 오래 되지 않았고, 아직 진행 중이기 때문에 지금부터 대비한다면 **대비할 시간은 충분**하다.

소프트웨어 제품의 구현에 집중하다 보면 보안에 소홀해지곤 한다. 당장 눈에 보이지 않는 위험에 대

한 지불을 꺼리는 것이다. 솔라윈즈 공급망 해킹 사건을 보면 알 수 있듯이 공급망 공격은 큰 파급력을 가지고 있다. 소프트웨어 제품에 있는 **작은 취약점이 돌이킬 수 없는 피해**를 낳을 수 있다는 것이다. 이번 행정명령이 국내에서도 **보안을 중요시하는 문화**가 자리잡는 계기가 될 수 있을 것으로 생각된다.

참고문헌

1. Exec. Order No. 14028, 86 Fed. Reg. 26633 (May 12, 2021). https://www.federalregister.gov/documents/2021/05/17/2021-10460/improving-the-nations-cybersecurity

2. NIST. (2021) Definition of Critical Software Under Executive Order (EO) 14028. https://www.nist.gov/document/white-paper-critical-software-enhancing-security-software-supply-chain

3. NIST. (2021) Security Measures for “EO-Critical Software” Use Under Executive Order (EO) 14028. https://www.nist.gov/document/security-measures-eo-critical-software-use-under-executive-order-eo-14028-pdf-version

4. The United States Department of Commerce. (2021) The Minimum Elements For a Software Bill of Materials (SBOM). https://www.ntia.doc.gov/files/ntia/publications/sbom_minimum_elements_report.pdf

5. NIST. (2021) Guidelines on Minimum Standards for Developer Verification of Software. https://www.nist.gov/document/document-developer-verification-software

6. NIST. (2021) Cybersecurity Supply Chain Risk Management Practices for Systems and Organizations. https://doi.org/10.6028/NIST.SP.800-161r1-draft2

7. NIST. (2021) Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities. https://doi.org/10.6028/NIST.SP.800-218-draft

8. 과학기술정보통신부. (2021) K-사이버방역추진전략 발표. https://www.msit.go.kr/bbs/view.do?sCode=user&mPid=112&mId=113&bbsSeqNo=94&nttSeqNo=3179937

9. 과학기술정보통신부. (2021) 소프트웨어 개발보안 시대 활짝 열렸다. https://www.msit.go.kr/bbs/view.do?sCode=user&mId=113&mPid=112&bbsSeqNo=94&nttSeqNo=3180584

10. 이희조, "오픈소스 SW 공급망 공격 대응기술 동향", 금융보안원-전자금융과 금융보안, Vol. 26, pp. 13, Nov. 23. 2021.

Appendix A

- 사이버보안 행정명령 4절 세부 항목

항	기간	책임	협력	내용
(a)	해당 없음	해당 없음	해당 없음	개요
(b)	30 일 이내	상무부-NIST	해당 없음	(e)에 맞는 기준, 도구 및 모범 사례 특정을 위한 의견 수집
(c)	180 일 이내	NIST	해당 없음	공급망 보안 강화를 위한 예비 가이드라인 공표
(d)	360 일 이내	NIST	해당 없음	(c)의 가이드라인의 검토, 보완 절차를 포함한 추가 가이드라인 공표
(e)	(c)로부터 90 일 이내	상무부-NIST	적절한 기관	공급망 보안 강화를 위한 실행 지침 발행 *지침에 포함될 기준 포함
(f)	60 일 이내	상무부	통신정보담당차관보, 미국통신정보국	SBOM 의 최소 요소 공표
(g)	45 일 이내	상무부-NIST	국방부-NSA, 국토안보부-CISA, OMB, 국가정보장실	'중요한 소프트웨어'의 정의 공표
(h)	(g)로부터 30 일 이내	국토안보부-CISA	상무부-NIST	사용 중 혹은 발주 단계인 소프트웨어 중 '중요한 소프트웨어'에 해당되는 제품 목록 특정 후 각 부처에 제공
(i)	60 일 이내	상무부-NIST	국토안보부-CISA, OMB	'중요한 소프트웨어'의 보안조치에 대한 지침 발행
(j)	(i)로부터 30 일 이내	OMB-전자정부국	해당 없음	각 부처가 (i)의 지침을 준수하기 위한 조치
(k)	(e)로부터 30 일 이내	OMB-전자정부국	해당 없음	명령 이후 조달된 소프트웨어에 대해 각 부처가 (i)의 지침을 준수하도록 요구하기 위한 조치
(l)	(k) 이후	각 기관	해당 없음	(k)의 요구사항 준수 연장 요구
(m)	(k) 이후	각 기관	해당 없음	(k)의 요구사항 면제 요구
(n)	1 년 이내	국토안보부	국방부, 법무부, OMB, OMB-전자정부국	(g)~(k)의 요구사항을 소프트웨어 공급자가 준수하고, 준수하고 있음을

				설명하는 것을 요구하는 계약언어 FAR Council 에 권고
(o)	(n) 이후	FAR Council	해당 없음	(n)의 권고 검토 후 FAR 개정
(p)	(o) 이후	각 부처	해당 없음	(o)의 FAR 개정 이후, 개정된 FAR 을 만족하지 않는 소프트웨어를 계약에서 삭제
(q)		OMB-전자정부국	해당 없음	명령 이전에 개발 및 조달된 소프트웨어를 채용하고 있는 기관에 (k)의 요구사항을 준수할 것을 요구
(r)	60 일 이내	상무부-NIST	국방부-NSA	소프트웨어 소스 코드 테스트의 최저 기준에 대한 가이드라인 공표
(s)		상무부-NIST	적절한 기관	기존 소비자용 제품의 라벨링 프로그램을 참고하여 대중에게 IoT 기기와 소프트웨어 개발 관행에 대한 보안을 교육하기 위한 파일럿 프로그램 개시
(t)	270 일 이내	상무부-NIST	연방거래위원회, 적절한 기관	소비자용 라벨링 프로그램을 위한 IoT 사이버보안 기준 특정
(u)	270 일 이내	상무부-NIST	연방거래위원회, 적절한 기관	안전한 소프트웨어 개발 관행이나 소비자용 라벨링 프로그램의 기준 특정
(v)	해당 없음	해당 없음	해당 없음	파일럿 프로그램은 OMB Circular A-119 및 NIST Special Publication 2000-02 에 준거해야 함
(w)	1 년 이내	NIST	해당 없음	파일럿 프로그램 검토 및 유효성 평가, 개선 방안 도출하여 APNSA 에 보고
(x)	1 년 이내	상무부	적절한 기관	진척 상황 검토 및 추가 조치를 설명한 보고서를 대통령에게 제출

Appendix B

- Recommended Minimum Standard for Vendor or Developer Verification of Code The following are recommended minimums for verification of code by developers.

Technique Class	Technique	Description & Reference to Recommended Minimums Document
Threat modeling	Threat modeling helps identify key or potentially overlooked testing targets.	Section 2.1. Threat modeling methods create an abstraction of the system, profiles of potential attackers and their goals and methods, and a catalog of potential threats. Threat modeling can identify design-level security issues and help focus verification.
Automated testing	As testing is automated, it can be repeated often, for instance upon every commit or before an issue is retired.	Section 2.2. Automated testing can run tests consistently, check results accurately, and minimize the need for human effort and expertise. Automated testing can be integrated into the existing workflow or issue tracking system.
Code-based (static) analysis	Use a code scanner to look for top bugs.	Section 2.3. Static analysis tools can check code for many kinds of vulnerabilities and for compliance with the organization's coding standards. For multi-threaded or parallel processing software, use a scanner capable of detecting race conditions.
	Review for hardcoded secrets.	Section 2.4. Heuristic tools can be somewhat effective checking for hardcoded passwords and private encryption keys since functions or services taking these as parameters have specific interfaces.
Dynamic analysis (i.e., run the program on test cases)	Run with built-in checks and protections.	Section 2.5. Programming languages, both compiled and interpreted, provide many built-in checks and protections.
	Create "black box" test cases.	Section 2.6. "Black box" tests can address functional specifications or requirements, negative tests (invalid inputs and testing what the software should not do), denial of service and overload attempts, input boundary analysis, and input combinations.
	Create code-based structural test cases.	Section 2.7. Code-based, or structural, test cases are based on the implementation, that is, the specifics of the code. Code-based test cases may also come from coverage metrics.
	Use test cases created to catch previous bugs.	Section 2.8. Test cases which have been created to specifically show the presence (and later, the absence) of a bug can be used to identify issues in the absence of more general "first principles" assurance approaches for detecting bugs.
	Run a fuzzer.	Section 2.9. Fuzzers can try an immense

		number of inputs with minimal human supervision. The tools can be programmed with inputs that often reveal bugs, such as very long or empty inputs and special characters.
	If the software might be connected to the Internet, run a web app scanner.	Section 2.10. If there is a network interface, use a dynamic security testing tool (e.g., web application scanner) to detect vulnerabilities.
Check included software	Use similar techniques to gain assurance that included libraries, packages, services, etc. are no less secure than your code.	Section 2.11. Use the verification techniques recommended in this section to gain assurance that included code is at least as secure as code developed locally. The components of your software must be continually monitored against databases of known vulnerabilities; a new vulnerability in existing code may be reported at any time.
Fix bugs	Fix critical bugs that are uncovered.	Correct critical bugs as soon as possible and make process improvements necessary to prevent such bugs in the future, or to at least catch them earlier in the development process.

발 행 일	2022년 1월
발 행 처	한국인터넷진흥원 (전라남도 나주시 진흥길 9)
기 획	한국인터넷진흥원 미래정책연구실 정책분석팀
편 집	㈜ 해리