관 인 생 략

출 원 번 호 통 지 서

출 원 일 자	2024.02.19
특 기 사 항	심사청구(유) 공개신청(무) 참조번호(W23P0340KR)
출 원 번 호	10-2024-0023236 (접수번호 1-1-2024-0186379-15) (DAS접근코드F210)
출원인 명칭	고려대학교 산학협력단(2-2004-017068-0)
대리인 성명	특허법인 위솔(9-2022-100001-9)
발명자 성명	우승훈 이희조
발명의 명칭	타겟 소프트웨어의 구성 요소에 내재된 오픈소스 소프트웨어의 보안 취약점 전파 탐지 방법 및 장치

특 허 청 장

<< 안내 >>

1. 귀하의 출원은 위와 같이 정상적으로 접수되었으며, 이후의 심사 진행상황은 출원번호를 이용하여 특허로 홈페이지(www.patent.go.kr)에서 확인하실 수 있습니다.
2. 출원에 따른 수수료는 접수일로부터 다음날까지 동봉된 납입영수증에 성명, 납부자번호 등을 기재하여 가까운 은행 또는 우체국에 납부하여야 합니다.
※ 납부자번호 : 0131(기관코드) + 접수번호
3. 귀하의 주소, 연락처 등의 변경사항이 있을 경우, 즉시 [특허고객번호 정보변경(경정), 정정신고서]를 제출하여야 출원 이후의 각종 통지서를 정상적으로 받을 수 있습니다.
4. 기타 심사 절차(제도)에 관한 사항은 특허청 홈페이지를 참고하시거나 특허고객상담센터(☎ 1544-8080)에 문의하여 주시기 바랍니다.
※ 심사제도 안내 : https://www.kipo.go.kr-지식재산제도

【서지사항】

【서류명】 특허출원서

【참조번호】 W23P0340KR

【출원구분】 특허출원

【출원인】

【명칭】 고려대학교산학협력단

【특허고객번호】 2-2004-017068-0

【대리인】

【명칭】 특허법인 위솔

【대리인번호】 9-2022-100001-9

【지정된변리사】 나강은, 김경용, 강현모, 편진호

【발명의 국문명칭】 타겟 소프트웨어의 구성 요소에 내재된 오픈소스 소프트웨어의 보안 취약점 전파 탐지 방법 및 장치

【발명의 영문명칭】 METHOD AND APPARATUS FOR DETECTING PROPAGATION OF SECURITY VULNERABILITIES OF OPEN SOURCE SOFTWARE INHERENT IN COMPONENTS OF TARGET SOFTWARE

【발명자】

【성명】 우승훈

【성명의 영문표기】 WOO, Seung Hoon

【주민등록번호】

【우편번호】

【주소】

【발명자】

【성명】 이희조

【성명의 영문표기】 LEE, Hee Jo

【주민등록번호】

【우편번호】

【주소】

【출원언어】 국어

【심사청구】 청구

【공지예외적용대상증명서류의 내용】

【공개형태】 논문발표 (USENIX Security 2023)

【공개일자】 2023.08.11

【이 발명을 지원한 국가연구개발사업】

【과제고유번호】 1711193663

【과제번호】 2020-0-01819-004

【부처명】 과학기술정보통신부

【과제관리(전문)기관명】 정보통신기획평가원

【연구사업명】 정보통신방송혁신인재양성

【연구과제명】 ICT명품인재양성(고려대학교)

【과제수행기관명】 고려대학교산학협력단

【연구기간】 2024.01.01 ~ 2024.12.31

【이 발명을 지원한 국가연구개발사업】

【과제고유번호】 1711193387

【과제번호】 2022-0-01198-002

【부처명】 과학기술정보통신부

【과제관리(전문)기관명】 정보통신기획평가원

【연구사업명】 정보통신방송혁신인재양성

【연구과제명】 융합보안대학원(고려대학교)

【과제수행기관명】 고려대학교산학협력단

【연구기간】 2024.01.01 ~ 2024.12.31

【이 발명을 지원한 국가연구개발사업】

【과제고유번호】 1711193841

【과제번호】 2022-0-00277-002

【부처명】 과학기술정보통신부

【과제관리(전문)기관명】 정보통신기획평가원

【연구사업명】 정보보호핵심원천기술개발(R&D)

【연구과제명】 SW공급망 보안을 위한 SBOM 자동생성 및 무결성 검증기술 개발

【과제수행기관명】 고려대학교산학협력단

【연구기간】 2024.01.01 ~ 2024.12.31

【취지】 위와 같이 특허청장에게 제출합니다.

대리인 특허법인 위솔 (서명 또는 인)

【수수료】

【출원료】 0 면 원

【가산출원료】 40 면 원

【우선권주장료】 0 건 원

【심사청구료】 10 항 원

【합계】 원

【감면사유】 전담조직(50%감면)[1]

【감면후 수수료】 원

【첨부서류】 1.공지예외적용대상(신규성상실의예외, 출원시의특례)규정을 적용받기 위한 증명서류_1통 2.기타첨부서류_1통

【발명의 설명】

【발명의 명칭】

타겟 소프트웨어의 구성 요소에 내재된 오픈소스 소프트웨어의 보안 취약점 전파 탐지 방법 및 장치{METHOD AND APPARATUS FOR DETECTING PROPAGATION OF SECURITY VULNERABILITIES OF OPEN SOURCE SOFTWARE INHERENT IN COMPONENTS OF TARGET SOFTWARE}

【기술분야】

【0001】본 발명은 보안 취약점을 가진 오픈소스 소프트웨어의 소스코드를 재사용하여 구성된 타겟 소프트웨어에 원본 오픈소스 소프트웨어에 내재된 보안 취약점이 전파되었는지 여부를 탐지하는 기술에 관한 것이다.

【발명의 배경이 되는 기술】

【0002】오픈소스 소프트웨어란 소스코드가 공개되어 있으면서 라이선스를 준수한다면 누구나 재사용, 수정 및 재배포가 가능한 소프트웨어를 말한다. 이때 보안 취약점이 포함된 오픈소스 소프트웨어의 소스코드가 다른 소프트웨어에서 재사용되면 원본 오픈소스 소프트웨어에 내재된 보안 취약점이 그대로 전파되게 되는데, 이러한 현상을 취약점 전파라 한다.

【0003】한편, 소프트웨어의 배포는 매우 빠르고 간편하게 일어난다는 자체적인 특성에 따라 보안 취약점이 광범위의 단말에 전파되면 전체적인 소프트웨어 생태계의 보안성을 저해한다는 문제가 발생한다.

【0004】현재는 이런 문제를 완화하고자 배포된 오픈소스 소프트웨어의 버전을 기준으로 알려진 취약점을 탐지하는 버전 기반 탐지 기술과, 소스코드를 기반으로 취약점을 탐지하는 코드 기반 탐지 기술이 주로 활용되고 있다.

【0005】버전 기반 탐지 기술은 재사용중인 오픈소스 소프트웨어에 대한 구성 요소의 이름과 버전 정보를 기반으로 다른 소프트웨어에 전파된 취약점을 탐지하는 방식이다. 일반적으로 오픈소스 소프트웨어를 재사용한 소프트웨어는 재사용한 오픈소스 소프트웨어의 이름과 버전 정보를 명시하고 있다. 이에 따라, 버전 기반 탐지 기술은 특정 소프트웨어가 재사용한 오픈소스 소프트웨어의 버전을 기반으로 특정 소프트웨어에 내재된 취약점을 탐지할 수 있다.

【0006】다만, 오픈소스 소프트웨어의 코드는 필요한 부분만 재사용되거나, 재사용 과정에서 원본 코드에 대한 수정이 발생하는 경우가 대부분이다. 즉, 오픈소스 소프트웨어에서 특정 취약점이 내재된 부분이 재사용되지 않을 경우가 있고, 오픈소스 소프트웨어에서 취약점이 있는 부분을 보완하여 수정된 채로 재사용되는 경우가 있기 때문에, 버전 기반의 취약점 탐지 기술은 취약점이 내재된 부분이 재사용 되지 않음에도 취약점이라고 탐지하는 오탐지 문제를 발생시킨다.

【0007】코드 기반 탐지 기술은 실제 재사용된 오픈소스 소프트웨어의 소스코드를 분석하여, 알려진 취약점과 유사한 코드가 포함되어 있는지를 식별함으로써 전파된 취약점을 탐지하는 방식이다. 한편, 오픈소스 소프트웨어의 소스코드가 재사용되는 과정에서 원본 코드는 수정될 수 있다. 이로 인해, 코드 기반 탐지 기술은 취약점이 내재된 채로 코드가 일부 수정되어 사용되는 경우 취약점을 탐지하지

못하는 미탐지 문제를 발생시킨다.

【0008】 이에 따라, 오픈소스 소프트웨어의 코드를 필요한 부분만 재사용하거나, 재사용 과정에서 원본 코드에 대한 수정이 발생할 수 있다는 재사용의 특성을 고려하여 타겟 소프트웨어에 전파된 취약점을 정확하게 탐지할 필요가 있다.

【선행기술문헌】

【특허문헌】

【0009】 **(특허문헌 0001)** 대한민국 등록특허공보 제10-1792631호

【발명의 내용】

【해결하고자 하는 과제】

【0010】 본 발명이 해결하고자 하는 과제는 오픈소스 소프트웨어의 소스코드 중 일부분만 재사용하거나, 재사용 과정에서 원본 코드에 대한 수정이 발생할 수 있다는 오픈소스 소프트웨어의 재사용 특성을 고려하여 타겟 소프트웨어에 전파된 취약점과 전파되지 않은 취약점을 정확하게 구분하여 탐지할 수 있게 하여, 보안 취약점 탐지 시 오탐지와 미탐지 문제를 방지하는 기술을 제공하는 것이다.

【0011】 한편, 본 발명의 기술적 과제들은 이상에서 언급한 기술적 과제들로 제한되지 않으며, 언급되지 않은 또 다른 기술적 과제들은 아래의 기재로부터 통상의 기술자에게 명확하게 이해될 수 있을 것이다.

【과제의 해결 수단】

【0012】일 실시예에 따른 프로세서에 의해 동작하는 보안 취약점 전파 탐지 장치가 수행하는 동작의 방법에 있어서, 타겟 소프트웨어 및 상기 타겟 소프트웨어에 사용된 오픈소스 소프트웨어의 버전 정보를 획득하는 동작; 상기 버전의 오픈소스 소프트웨어에 포함된 함수 중 상기 타겟 소프트웨어에 동일하게 사용된 제1 함수, 수정되어 사용된 제2 함수, 비사용된 제3 함수를 분류하는 동작; 상기 제2 함수 및 상기 버전의 원본 함수 간의 제1 유사도와, 상기 제2 함수 및 상기 버전에 대한 패치 함수간의 제2 유사도를 기초로 상기 제2 함수의 패치 여부를 판별하는 동작; 상기 제1 함수, 상기 제2 함수 및 상기 제3 함수의 분류와 상기 제2 함수의 패치 여부에 기초하여 상기 버전의 오픈소스 소프트웨어에 내재된 취약점 중 상기 타겟 소프트웨어에 전파된 취약점을 탐지하는 동작을 포함할 수 있다.

【0013】또한, 상기 버전 정보는 상기 버전의 오픈소스 소프트웨어에 포함된 함수, 상기 버전에 따른 오픈소스 소프트웨어에 내재된 취약점과 취약점을 유발하는 함수, 취약점을 유발하는 함수를 보완하는 패치 함수에 대한 정보를 포함할 수 있다.

【0014】또한, 상기 분류하는 동작은 상기 타겟 소프트웨어에 포함된 함수를 해시화한 제1 해시 정보와 상기 오픈소스 소프트웨어에 포함된 함수를 해시화한 제2 해시 정보를 생성하는 동작; 상기 제1 해시 정보와 상기 제2 해시 정보 간의 유사도를 도출하는 동작; 및 상기 유사도에 대하여 기 설정된 범위를 기준으로 상기 오픈소스 소프트웨어에 포함된 함수를 제1 함수, 제2 함수 및 제3 함수 중 하나로 분류하는 동작을 포함할 수 있다.

【0015】또한, 상기 패치 여부를 판별하는 동작은 상기 제1 유사도가 상기 제2 유사도보다 큰 경우 상기 제2 함수는 원본에 내재된 취약점이 패치되지 않은 함수인 것으로 판별하고, 상기 제2 유사도가 상기 제1 유사도보다 큰 경우 상기 제2 함수는 원본에 내재된 취약점이 패치된 함수인 것으로 판별할 수 있다.

【0016】또한, 상기 취약점을 탐지하는 동작은 상기 오픈소스 소프트웨어에 내재된 취약점 중 상기 제1 함수에 대응하는 취약점이 전파되고, 상기 제2 함수 중 패치된 것으로 판별된 제2 함수의 취약점이 비전파되고, 상기 제2 함수 중 패치되지 않은 것으로 판별된 제2 함수의 취약점이 전파되고, 상기 제3 함수에 대응하는 취약점이 비전파된 것으로 탐지하는 동작을 포함할 수 있다.

【0017】또한, 상기 방법은 상기 오픈소스 소프트웨어에 내재된 취약점을 유발하는 코드의 패치 정보를 매크로, 변수, 함수, 구조체 별로 상기 타겟 소프트웨어와 비교하여 코드 기반으로 전파된 취약점을 탐지하는 동작을 더 포함할 수 있다.

【0018】또한, 상기 코드 기반으로 전파된 취약점을 탐지하는 동작은 상기 오픈소스 소프트웨어에 내재된 취약점을 유발하는 코드의 패치 정보를 매크로, 변수, 함수, 구조체 별로 구분하는 동작; 상기 타겟 소프트웨어에 포함된 함수 또는 구조체를 각각 해시화한 제3 해시 정보와 상기 패치 정보에 포함된 함수 또는 구조체를 각각 해시화한 제4 해시 정보를 생성하는 동작; 상기 패치 정보에 따른 매크로 또는 변수의 코드와 상기 타겟 소프트웨어에 포함된 매크로 또는 변수의 코드를 비교하여, 상기 타겟 소프트웨어에 포함된 매크로 또는 변수에 전파된 취약점을 탐

지하는 동작; 및 상기 제3 해시 정보와 상기 제4 해시 정보를 비교하여 상기 타겟 소프트웨어에 포함된 함수 또는 구조체에 전파된 취약점을 탐지하는 동작을 포함할 수 있다.

【0019】 또한, 상기 매크로 또는 변수에 전파된 취약점을 탐지하는 동작은 상기 패치 정보에 따른 매크로 또는 변수의 코드와 상기 타겟 소프트웨어에 포함된 코드가 동일한 경우 취약점이 비전파된 것으로 탐지하고, 상기 패치 정보에 따른 매크로 또는 변수의 코드와 상기 타겟 소프트웨어에 포함된 코드가 동일하지 않은 경우 재사용된 버전의 오픈소스 소프트웨어에 포함된 취약점이 전파된 것으로 탐지하는 동작을 포함할 수 있다.

【0020】 또한, 상기 함수 또는 구조체에 전파된 취약점을 탐지하는 동작은 상기 제3 해시 정보와 상기 제4 해시 정보의 유사도가 동일한 것으로 판별된 경우 상기 타겟 소프트웨어에 포함된 함수 또는 구조체에 취약점이 비전파된 것으로 탐지하고, 상기 제3 해시 정보와 상기 제4 해시 정보의 유사도가 동일하지 않은 경우 재사용된 버전의 오픈소스 소프트웨어에 포함된 함수 또는 구조체에 취약점이 전파된 것으로 탐지하는 동작을 포함할 수 있다.

【0021】 일 실시예에 따른 보안 취약점 전파 탐지 장치는 명령어를 포함하는 메모리; 및 상기 명령어를 기초로 소정의 동작을 수행하는 프로세서를 포함하고, 상기 프로세서의 동작은 타겟 소프트웨어 및 상기 타겟 소프트웨어에 사용된 오픈소스 소프트웨어의 버전 정보를 획득하는 동작; 상기 버전의 오픈소스 소프트웨어에 포함된 함수 중 상기 타겟 소프트웨어에 동일하게 사용된 제1 함수, 수정되어

사용된 제2 함수, 비사용된 제3 함수를 분류하는 동작; 상기 제2 함수 및 상기 버전의 원본 함수 간의 제1 유사도와, 상기 제2 함수 및 상기 버전에 대한 패치 함수 간의 제2 유사도를 기초로 상기 제2 함수의 패치 여부를 판별하는 동작; 및 상기 제1 함수, 상기 제2 함수 및 상기 제3 함수의 분류와 상기 제2 함수의 패치 여부에 기초하여 상기 버전의 오픈소스 소프트웨어에 내재된 취약점 중 상기 타겟 소프트웨어에 전파된 취약점을 탐지하는 동작을 포함할 수 있다.

【0022】 또한, 상기 버전 정보는 상기 버전의 오픈소스 소프트웨어에 포함된 함수, 상기 버전에 따른 오픈소스 소프트웨어에 내재된 취약점과 취약점을 유발하는 함수, 취약점을 유발하는 함수를 보완하는 패치 함수에 대한 정보를 포함할 수 있다.

【0023】 또한, 상기 분류하는 동작은 상기 타겟 소프트웨어에 포함된 함수를 해시화한 제1 해시 정보와 상기 오픈소스 소프트웨어에 포함된 함수를 해시화한 제2 해시 정보를 생성하는 동작; 상기 제1 해시 정보와 상기 제2 해시 정보 간의 유사도를 도출하는 동작; 및 상기 유사도에 대하여 기 설정된 범위를 기준으로 상기 오픈소스 소프트웨어에 포함된 함수를 제1 함수, 제2 함수 및 제3 함수 중 하나로 분류하는 동작을 포함할 수 있다.

【0024】 또한, 상기 패치 여부를 판별하는 동작은 상기 제1 유사도가 상기 제2 유사도보다 큰 경우 상기 제2 함수는 원본에 내재된 취약점이 패치되지 않은 함수인 것으로 판별하고, 상기 제2 유사도가 상기 제1 유사도보다 큰 경우 상기 제2 함수는 원본에 내재된 취약점이 패치된 함수인 것으로 판별할 수 있다.

【0025】또한, 상기 취약점을 탐지하는 동작은 상기 오픈소스 소프트웨어에 내재된 취약점 중 상기 제1 함수에 대응하는 취약점이 전파되고, 상기 제2 함수 중 패치된 것으로 판별된 제2 함수의 취약점이 비전파되고, 상기 제2 함수 중 패치되지 않은 것으로 판별된 제2 함수의 취약점이 전파되고, 상기 제3 함수에 대응하는 취약점이 비전파된 것으로 탐지하는 동작을 포함할 수 있다.

【0026】또한, 상기 프로세서의 동작은 상기 오픈소스 소프트웨어에 내재된 취약점을 유발하는 코드의 패치 정보를 매크로, 변수, 함수, 구조체 별로 상기 타겟 소프트웨어와 비교하여 코드 기반으로 전파된 취약점을 탐지하는 동작을 더 포함할 수 있다.

【0027】또한, 상기 코드 기반으로 전파된 취약점을 탐지하는 동작은 상기 오픈소스 소프트웨어에 내재된 취약점을 유발하는 코드의 패치 정보를 매크로, 변수, 함수, 구조체 별로 구분하는 동작; 상기 타겟 소프트웨어에 포함된 함수 또는 구조체를 각각 해시화한 제3 해시 정보와 상기 패치 정보에 포함된 함수 또는 구조체를 각각 해시화한 제4 해시 정보를 생성하는 동작; 상기 패치 정보에 따른 매크로 또는 변수의 코드와 상기 타겟 소프트웨어에 포함된 매크로 또는 변수의 코드를 비교하여, 상기 타겟 소프트웨어에 포함된 매크로 또는 변수에 전파된 취약점을 탐지하는 동작; 및 상기 제3 해시 정보와 상기 제4 해시 정보를 비교하여 상기 타겟 소프트웨어에 포함된 함수 또는 구조체에 전파된 취약점을 탐지하는 동작을 포함할 수 있다.

【0028】또한, 상기 매크로 또는 변수에 전파된 취약점을 탐지하는 동작은 상기 패치 정보에 따른 매크로 또는 변수의 코드와 상기 타겟 소프트웨어에 포함된 코드가 동일한 경우 취약점이 비전파된 것으로 탐지하고, 상기 패치 정보에 따른 매크로 또는 변수의 코드와 상기 타겟 소프트웨어에 포함된 코드가 동일하지 않은 경우 재사용된 버전의 오픈소스 소프트웨어에 포함된 취약점이 전파된 것으로 탐지하는 동작을 포함할 수 있다.

【0029】또한, 상기 함수 또는 구조체에 전파된 취약점을 탐지하는 동작은 상기 제3 해시 정보와 상기 제4 해시 정보의 유사도가 동일한 것으로 판별된 경우 상기 타겟 소프트웨어에 포함된 함수 또는 구조체에 취약점이 비전파된 것으로 탐지하고, 상기 제3 해시 정보와 상기 제4 해시 정보의 유사도가 동일하지 않은 경우 재사용된 버전의 오픈소스 소프트웨어에 포함된 함수 또는 구조체에 취약점이 전파된 것으로 탐지하는 동작을 포함할 수 있다.

【발명의 효과】

【0030】본 발명은 오픈소스 소프트웨어로부터 전파된 취약점을 탐지할 대상인 타겟 소프트웨어와 오픈소스 소프트웨어에 포함된 함수를 비교하여, 오픈소스 소프트웨어의 함수 중 타겟 소프트웨어에 동일하게 사용된 제1 함수, 수정되어 사용된 제2 함수, 비사용된 제3 함수를 분류할 수 있다. 즉, 본 발명은 타겟 소프트웨어가 오픈소스 소프트웨어의 코드 중 재사용한 부분과 재사용하지 않은 부분을 특정할 수 있고, 오픈소스 소프트웨어의 함수에 대한 수정이 발생한 부분을 명확히

특정할 수 있다. 이에 따라, 본 발명은 오픈소스 소프트웨어로부터 타겟 소프트웨어에 실제로 전파된 취약점을 탐지하여, 보안 취약점 탐지 시 오탐지 및 미탐지의 문제를 현저하게 감소시킬 수 있다.

【0031】 더하여, 본 발명은 오픈소스 소프트웨어에 내재된 취약점을 유발하는 코드의 패치 정보를 매크로, 변수, 함수, 구조체 별로 구분하여 비교할 수 있고, 나아가 함수 또는 구조체를 각각 해시화한 해시 정보를 비교함에 따라 코드 기반 취약점 탐지를 효율적으로 수행할 수 있다.

【0032】 이로써, 본 발명은 보안 취약점 탐지에 효과적으로 활용될 수 있을 뿐 아니라 소프트웨어의 공급망 보안에 응용될 수 있으며, 궁극적으로 안전한 소프트웨어 생태계를 구축할 수 있다.

【0033】 한편, 본 발명에 의한 효과는 이상에서 언급한 것들로 제한되지 않으며, 언급되지 않은 또 다른 기술적 효과들은 아래의 기재로부터 통상의 기술자에게 명확하게 이해될 수 있을 것이다.

【도면의 간단한 설명】

【0034】 도 1은 일 실시예에 따른 보안 취약점 전파 탐지 장치의 구성도이다.

도 2는 일 실시예에 따른 보안 취약점의 전파 탐지 장치가 수행하는 동작의 단계를 나타낸 흐름도이다.

도 3은 일 실시예에 따라 타겟 소프트웨어가 오픈소스 소프트웨어의 함수를

사용하는 형태의 예시도이다.

도 4는 일 실시예에 따라 오픈소스 소프트웨어에 포함된 함수가 타겟 소프트웨어에 사용된 형태를 분류하는 예시도이다.

도 5는 일 실시예에 따라 특정 버전에 대해 배포되는 오픈소스 소프트웨어의 취약점 정보로서, 특정 버전에서 취약점을 가진 원본 함수와 해당 원본 함수의 취약점을 개선할 수 있는 패치 함수의 예시를 도시한 도면이다.

도 6은 일 실시예에 따라 제2 함수를 원본 함수 및 패치 함수와 각각 비교하여 제2 함수의 패치 여부를 판별하는 동작의 예시도이다.

도 7은 일 실시예에 따라 S1050의 동작을 구체화한 흐름도이다.

도 8은 오픈소스 소프트웨어에 내재된 취약점을 유발하는 코드의 패치 정보(a)를 매크로, 변수, 함수, 구조체 별로 구분하여 전처리(b)하고, 구분된 정보의 특성 별로 타겟 소프트웨어에 대한 취약점 전파 여부를 탐지하는 흐름도이다.

【발명을 실시하기 위한 구체적인 내용】

【0035】 본 발명의 목적과 기술적 구성 및 그에 따른 작용 효과에 관한 자세한 사항은 본 발명의 명세서에 첨부된 도면에 의거한 이하의 상세한 설명에 의해 보다 명확하게 이해될 것이다. 첨부된 도면을 참조하여 본 발명에 따른 실시예를 상세하게 설명한다.

【0036】 본 명세서에서 개시되는 실시예들은 본 발명의 범위를 한정하는 것으로 해석되거나 이용되지 않아야 할 것이다. 이 분야의 통상의 기술자에게 본 명

세서의 실시예를 포함한 설명은 다양한 응용을 갖는다는 것이 당연하다. 따라서, 본 발명의 상세한 설명에 기재된 임의의 실시예들은 본 발명을 보다 잘 설명하기 위한 예시적인 것이며 본 발명의 범위가 실시예들로 한정되는 것을 의도하지 않는다.

【0037】 도면에 표시되고 아래에 설명되는 기능 블록들은 가능한 구현의 예들일 뿐이다. 다른 구현들에서는 상세한 설명의 사상 및 범위를 벗어나지 않는 범위에서 다른 기능 블록들이 사용될 수 있다. 또한, 본 발명의 하나 이상의 기능 블록이 개별 블록들로 표시되지만, 본 발명의 기능 블록들 중 하나 이상은 동일 기능을 실행하는 다양한 하드웨어 및 소프트웨어 구성들의 조합일 수 있다.

【0038】 또한, 어떤 구성요소들을 포함한다는 표현은 “개방형”의 표현으로서 해당 구성요소들이 존재하는 것을 단순히 지칭할 뿐이며, 추가적인 구성요소들을 배제하는 것으로 이해되어서는 안 된다.

【0039】 나아가 어떤 구성요소가 다른 구성요소에 “연결되어” 있다거나 “접속되어” 있다고 언급될 때에는, 그 다른 구성요소에 직접적으로 연결 또는 접속되어 있을 수도 있지만, 중간에 다른 구성요소가 존재할 수도 있다고 이해되어야 한다.

【0040】 이하, 본 발명의 다양한 실시예가 첨부된 도면을 참조하여 기재된다. 그러나, 이는 본 발명을 특정한 실시 형태에 대해 한정하려는 것이 아니며, 본 발명의 실시예의 다양한 변경(modification), 균등물(equivalent), 및/또는 대체물(alternative)을 포함하는 것으로 이해되어야 한다.

【0041】본 발명은 보안 취약점을 가진 오픈소스 소프트웨어의 소스코드를 재사용하여 구성된 타겟 소프트웨어에 원본 오픈소스 소프트웨어에 내재된 보안 취약점이 전파되었는지 여부를 탐지할 수 있다.

【0042】본 발명을 설명하기에 앞서, 오픈소스 소프트웨어란 소스코드가 공개되어 있으면서 라이선스를 준수한다면 누구나 재사용, 수정 및 재배포가 가능한 소프트웨어를 말한다. 이때 보안 취약점이 포함된 오픈소스 소프트웨어의 소스코드가 다른 소프트웨어에서 재사용되면 원본 오픈소스 소프트웨어에 내재된 보안 취약점이 그대로 전파되게 되는데, 이러한 현상을 취약점 전파라 한다.

【0043】한편, 소프트웨어의 배포는 매우 빠르고 간편하게 일어난다는 자체적인 특성에 따라 보안 취약점이 광범위의 단말에 전파되면 전체적인 소프트웨어 생태계의 보안성을 저해한다는 문제가 발생한다.

【0044】현재는 이런 문제를 완화하고자 배포된 오픈소스 소프트웨어의 버전을 기준으로 알려진 취약점을 탐지하는 버전 기반 탐지 기술과, 소스코드를 기반으로 취약점을 탐지하는 코드 기반 탐지 기술이 주로 활용되고 있다.

【0045】버전 기반 탐지 기술은 재사용중인 오픈소스 소프트웨어에 대한 구성 요소의 이름과 버전 정보를 기반으로 다른 소프트웨어에 전파된 취약점을 탐지하는 방식이다. 일반적으로 오픈소스 소프트웨어를 재사용한 소프트웨어는 재사용한 오픈소스 소프트웨어의 이름과 버전 정보를 명시하고 있다. 이에 따라, 버전 기반 탐지 기술은 특정 소프트웨어가 재사용한 오픈소스 소프트웨어의 버전을 기반으

로 특정 소프트웨어에 내재된 취약점을 탐지할 수 있다.

【0046】 다만, 오픈소스 소프트웨어의 코드는 필요한 부분만 재사용되거나, 재사용 과정에서 원본 코드에 대한 수정이 발생하는 경우가 대부분이다. 즉, 오픈소스 소프트웨어에서 특정 취약점이 내재된 부분이 재사용되지 않을 경우가 있고, 오픈소스 소프트웨어에서 취약점이 있는 부분을 보완하여 수정된 채로 재사용되는 경우가 있기 때문에, 버전 기반의 취약점 탐지 기술은 취약점이 내재된 부분이 재사용 되지 않음에도 취약점이라고 탐지하는 오탐지 문제를 발생시킨다.

【0047】 코드 기반 탐지 기술은 실제 재사용된 오픈소스 소프트웨어의 소스코드를 분석하여, 알려진 취약점과 유사한 코드가 포함되어 있는지를 식별함으로써 전파된 취약점을 탐지하는 방식이다. 한편, 오픈소스 소프트웨어의 소스코드가 재사용되는 과정에서 원본 코드는 수정될 수 있다. 이로 인해, 코드 기반 탐지 기술은 취약점이 내재된 채로 코드가 일부 수정되어 사용되는 경우 취약점을 탐지하지 못하는 미탐지 문제를 발생시킨다.

【0048】 본 발명은 오픈소스 소프트웨어의 소스코드 중 일부분만 재사용하거나, 재사용 과정에서 원본 코드에 대한 수정이 발생할 수 있다는 오픈소스 소프트웨어의 재사용 특성을 고려하여, 다음 도 1 내지 도 8의 실시예에 따라 타겟 소프트웨어에 전파된 취약점과 전파되지 않은 취약점을 정확하게 구분하여 탐지할 수 있게 하여, 보안 취약점 탐지 시 오탐지와 미탐지 문제를 방지하는 기술을 제안한다.

【0049】 도 1은 일 실시예에 따른 보안 취약점 전파 탐지 장치(100)(이하,

'장치(100)'로 지칭)의 구성도이다.

【0050】 도 1을 참조하면, 일 실시예에 따른 장치(100)는 각각 메모리(110), 프로세서(120), 입출력 인터페이스(130) 및 통신 인터페이스(140)를 포함할 수 있다.

【0051】 메모리(110)는 외부 장치로부터 획득한 데이터 또는 스스로 생성한 데이터를 저장할 수 있다. 메모리(110)는 프로세서(120)의 동작을 수행시킬 수 있는 명령어들을 저장할 수 있다. 또한, 메모리(110)는 후술할 타겟 소프트웨어에 대한 정보 및 타겟 소프트웨어에 사용된 오픈소스 소프트웨어에 대한 정보를 저장할 수 있다.

【0052】 프로세서(120)는 전반적인 동작을 제어하는 연산 장치이다. 프로세서(120)는 메모리(110)에 저장된 명령어들을 실행할 수 있다. 본 문서의 실시예에 따라 후술할 도 3의 장치(100) 동작은 프로세서(120)에 의해 수행되는 동작으로 이해될 수 있다.

【0053】 입출력 인터페이스(130)는 정보를 입력하거나 출력하는 하드웨어 인터페이스 또는 소프트웨어 인터페이스를 포함할 수 있다.

【0054】 통신 인터페이스(140)는 통신망을 통해 정보를 송수신 할 수 있게 한다. 이를 위해, 통신 인터페이스(140)는 무선 통신모듈 또는 유선 통신모듈을 포함할 수 있다.

【0055】장치(100)는 프로세서(120)를 통해 연산을 수행하고 네트워크를 통해 정보를 송수신할 수 있는 다양한 형태의 장치로 구현될 수 있다. 예를 들면, 서버, 컴퓨터 장치, 휴대용 통신 장치, 스마트 폰, 휴대용 멀티미디어 장치, 노트북, 태블릿 PC 등의 형태로 구현될 수 있으나, 이러한 예시에 한정되는 것은 아니다.

【0056】도 2는 일 실시예에 따른 장치(100)가 수행하는 동작의 흐름도이다. 도 2의 실시예에 따른 장치(100)의 동작은 프로세서(120)에 의해 수행되는 동작으로 이해될 수 있다.

【0057】도 2에 개시된 각 단계는 본 발명의 목적을 달성함에 있어서 바람직한 실시예일 뿐이며, 필요에 따라 일부 단계가 추가 또는 삭제될 수 있음은 물론이고, 어느 한 단계가 다른 단계에 포함되어 수행될 수도 있다. 도 3에 개시된 각 동작의 순서는 이해의 편의를 위해 배치된 순서일 뿐, 이러한 순서가 시계열적인 순서로 한정되는 것이 아니며, 설계자의 선택에 따라 순서가 다르게 변경되어 동작될 수 있다.

【0058】S1010 단계에서, 장치(100)는 타겟 소프트웨어 및 타겟 소프트웨어에 사용된 오픈소스 소프트웨어의 버전 정보를 획득할 수 있다. 타겟 소프트웨어란 오픈소스 소프트웨어의 소스코드를 재사용하여 만든 소프트웨어로서, 오픈소스 소프트웨어를 재사용함에 따라 전파된 취약점을 탐지할 대상인 소프트웨어이다.

【0059】장치(100)는 타겟 소프트웨어에 명시된 오픈소스 소프트웨어의 이름과 버전 정보로부터 오픈소스 소프트웨어의 버전 정보를 획득할 수 있다. 또한, 장

치(100)는 타겟 소프트웨어의 소스코드를 기초로 타겟 소프트웨어가 사용하고 있는 오픈소스 소프트웨어의 버전 정보를 획득할 수 있다.

【0060】 오픈소스 소프트웨어는 누구나 재사용, 수정 및 재배포할 수 있도록 소스코드를 공개할 뿐만 아니라, 특정 버전에 대해 알려진 취약점 정보와 그에 대한 취약점을 개선하기 위한 패치 정보를 함께 공개한다. 이에 따라, 장치(100)는 오픈소스 소프트웨어의 버전 정보를 기초로 해당 버전의 오픈소스 소프트웨어에 대해 알려진 취약점 정보를 획득할 수 있다. 즉, 장치(100)는 오픈소스 소프트웨어의 버전 정보를 기초로 해당 버전의 오픈소스 소프트웨어에 포함된 함수, 해당 버전에 따른 오픈소스 소프트웨어에 내재된 취약점과 취약점을 유발하는 함수, 취약점을 유발하는 함수를 보완하는 패치 함수에 대한 정보를 획득할 수 있다.

【0061】 도 3은 일 실시예에 따라 타겟 소프트웨어가 오픈소스 소프트웨어의 함수를 사용하는 형태의 예시도이다.

【0062】 도 3을 참조하면, 타겟 소프트웨어는 오픈소스 소프트웨어의 함수를 재사용하는 경우, 1) 필요한 부분만 그대로 재사용하거나(Exactly reused functions), 2) 원본 코드에 대한 수정을 가하여 재사용하거나(Modified functions), 3) 필요하지 않은 부분을 사용하지 않을 수 있다(Unused functions).

【0063】 한편, 오픈소스 소프트웨어의 버전 정보, 즉, 특정 버전의 오픈소스 소프트웨어가 가진 취약점 정보는 오픈소스 소프트웨어에 포함된 함수를 기준으로 취약점 정보를 명시하고 있기 때문에, 특정 버전을 기준으로 버전 기반 탐지 기술을 타겟 소프트웨어에 적용하는 경우, 도 2에 도시된 재사용 특성에 따라 오탐지

및 미탐지 문제가 발생할 수 있다. 예를 들어, 타겟 소프트웨어에서 오픈소스 소프트웨어의 특정 취약점이 내재된 부분이 재사용되지 않음에도 취약점이라고 탐지하는 오탐지 문제와, 타겟 소프트웨어에서 오픈소스 소프트웨어의 취약점이 있는 부분이 일부 수정되어 사용됨에도 수정에 의해 동일한 함수라고 인식하지 못하여 탐지하지 못하는 미탐지 문제가 발생할 수 있다. 이러한 문제를 해결하기 위해, 장치(100)는 다음 도 4와 같이 오픈소스 소프트웨어에 포함된 함수를 분류할 수 있다.

【0064】 도 4는 일 실시예에 따라 오픈소스 소프트웨어에 포함된 함수가 타겟 소프트웨어에 사용된 형태를 분류하는 예시도이다.

【0065】 도 4를 참조하면, S1020 단계에서, 장치(100)는 오픈소스 소프트웨어의 함수를 기준으로, 타겟 소프트웨어에 그대로 재사용된 제1 함수, 타겟 소프트웨어에서 수정된 채로 재사용된 제2 함수, 타겟 소프트웨어에서 사용하지 않은 제3 함수를 분류할 수 있다.

【0066】 일 예로, 장치(100)는 타겟 소프트웨어에 포함된 각 함수에 대한 코드를 해시화한 제1 해시 정보와, 오픈소스 소프트웨어에 포함된 각 함수에 대한 코드를 해시화한 제2 해시 정보를 생성할 수 있다. 예를 들어, 장치(100)는 TLSH 알고리즘을 이용한 fuzzy hash 기반의 해시 정보를 생성할 수 있다.

【0067】 이후, 장치(100)는 제1 해시 정보와 제2 해시 정보 간의 유사도를 도출할 수 있다. 예를 들어, 장치(100)는 ssdeep 알고리즘을 이용하여 유사도를 판별할 수 있고, 즉, 장치(100)는 오픈소스 소프트웨어에 포함된 각 함수의 제2 해시 정보를 타겟 소프트웨어에 포함된 모든 함수의 제1 해시 정보 비교할 수 있다. 이

에 따라, 특정 제2 해시 정보와 특정 제1 해시 정보의 유사도 점수가 '0' 이면 각 해시 정보가 '동일'인 것으로 판별하여, 해당 해당 제1 해시 정보와 제2 해시 정보의 함수를 제1 함수로 분류하고, 0 초과 30 이하이면 '유사'인 것으로 판별하여 해당 제1 해시 정보와 제2 해시 정보의 함수를 제2 함수로 분류하고, 30 초과이면 '비유사'인 것으로 판별하여 해당 제2 해시 정보의 함수를 제3 함수로 분류할 수 있다. 한편, 유사도 점수에 따라 동일/유사/비유사로 판별하는 범위는 설계에 따라 변경될 수 있다.

【0068】 이때, '동일'인 것으로 판별된 제1 함수로 분류된 함수는 그대로 사용된 것이기에, 해당 함수에 오픈소스 소프트웨어의 취약점이 존재하는 함수라면, 오픈소스의 취약점이 타겟 소프트웨어에 전파된 것으로 볼 수 있다. 또한, '비유사'인 것으로 판별된 제3 함수로 분류된 함수는 타겟 소프트웨어에 사용되지 않은 것이기에, 해당 함수에 오픈소스 소프트웨어의 취약점이 존재하는 함수라면, 오픈소스의 취약점이 타겟 소프트웨어에 전파되지 않은 것으로 볼 수 있다.

【0069】 한편, '유사'인 것으로 판별된 제2 함수로 분류된 함수는 오픈소스 소프트웨어의 취약점이 전파된 것인지 판단하기 위해서는, 단순히 타겟 소프트웨어의 설정에 맞도록 수정된 것인지, 취약점 정보가 개선되도록 수정된 것인지에 대한 판단이 필요하다. 단순히 타겟 소프트웨어의 설정에 맞도록 수정되어 사용된 것이라면 오픈소스의 취약점이 전파된 것으로 볼 수 있지만, 취약점 정보가 개선되도록 수정되어 사용된 것이라면 취약점이 전파되지 않은 것으로 볼 수 있기 때문이다.

【0070】도 5는 일 실시예에 따라 특정 버전에 대해 배포되는 오픈소스 소프트웨어의 취약점 정보로서, 특정 버전에서 취약점을 가진 원본 함수와 해당 원본 함수의 취약점을 개선할 수 있는 패치 함수의 예시를 도시한 도면이다.

【0071】도 5를 참조하면, 장치(100)는 제2 함수가 원본 함수와 더 유사한지 또는 패치 함수와 더 유사한 지를 판별하여, 제2 함수가 단순히 타겟 소프트웨어의 설정에 맞도록 수정된 것인지, 취약점 정보가 개선되도록 패치된 것인지에 대해 판별할 수 있다.

【0072】도 6은 일 실시예에 따라 제2 함수를 원본 함수 및 패치 함수와 각각 비교하여 제2 함수의 패치 여부를 판별하는 동작의 예시도이다.

【0073】도 6을 참조하면, S1030 단계에서, 장치(100)는 제2 함수 및 오픈소스 버전의 원본 함수 간의 제1 유사도를 도출하고, 제2 함수 및 오픈소스 버전의 원본 함수를 개선한 패치 함수 간의 제2 유사도를 도출할 수 있다. 장치(100)는 제2 함수가 패치 함수에 비해 원본 함수와의 유사도가 더 높은 경우, 해당 제2 함수는 패치되지 않은 것으로 판별할 수 있다. 장치(100)는 제2 함수가 원본 함수에 비해 패치 함수와의 유사도가 더 높은 경우, 해당 제2 함수는 패치되어 취약점이 개선된 함수인 것으로 판별할 수 있다. 한편, 함수 간의 유사도를 판별하는 동작은 공지된 다양한 비교 알고리즘을 사용할 수 있다.

【0074】S1040 단계에서, 장치(100)는 제1 함수, 제2 함수 및 제3 함수의 분류와, 제2 함수의 패치 여부에 기초하여 재사용된 버전의 오픈소스 소프트웨어에

내재된 취약점 중 타겟 소프트웨어에 전파된 취약점을 탐지할 수 있다. 예를 들어, 장치(100)는 오픈소스 소프트웨어에 내재된 취약점 중 제1 함수에 대응하는 취약점이 전파되고, 제2 함수 중 패치된 것으로 판별된 제2 함수의 취약점이 비전파되고, 제2 함수 중 패치되지 않은 것으로 판별된 제2 함수의 취약점이 전파되고, 제3 함수에 대응하는 취약점이 비전파된 것으로 탐지할 수 있다.

【0075】 S1050 단계에서, 장치(100)는 오픈소스 소프트웨어에 내재된 취약점을 유발하는 코드의 패치 정보를 매크로, 변수, 함수, 구조체 별로 상기 타겟 소프트웨어와 비교하여 코드 기반으로 전파된 취약점을 탐지할 수 있다.

【0076】 도 7은 일 실시예에 따라 S1050의 동작을 구체화한 흐름도이고, 도 8은 오픈소스 소프트웨어에 내재된 취약점을 유발하는 코드의 패치 정보(a)를 매크로, 변수, 함수, 구조체 별로 구분하여 전처리(b)하고, 구분된 정보의 특성 별로 타겟 소프트웨어에 대한 취약점 전파 여부를 탐지하는 흐름도이다.

【0077】 도 7 및 도 8을 참조하면, S1051 단계에서, 장치(100)는 특정 버전의 오픈소스 소프트웨어에 내재된 취약점을 유발하는 코드에 대한 패치 정보를, 매크로/변수/함수/구조체 별로 구분하여 패치 정보를 전처리할 수 있다. 즉, 도 8(a)와 같이, 패치 정보는 취약점을 가진 매크로/변수/함수/구조체 별로 추가해야 할 코드나 삭제해야 할 코드에 대한 정보를 포함하고 있다. 이에 따라, 장치(100)는 도 8(b)와 같이, 매크로/변수/함수/구조체 별로 구분하여, 각 구분에 따라 특정 매크로/변수/함수/구조체 별로 추가해야 할 코드(+)와 삭제해야 할 코드(-)를 추출하여 분류할 수 있다.

【0078】S1052 단계에서, 장치(100)는 타겟 소프트웨어에 포함된 함수 또는 구조체를 각각 해시화한 제3 해시 정보와, 패치 정보에 포함된 함수 또는 구조체를 각각 해시화한 제4 해시 정보를 생성할 수 있다.

【0079】S1053 단계에서, 장치(100)는 패치 정보에 따른 매크로 또는 변수의 코드와 타겟 소프트웨어에 포함된 매크로 또는 변수의 코드를 비교하여, 타겟 소프트웨어에 포함된 매크로 또는 변수에 전파된 취약점을 탐지할 수 있다. 예를 들어, 장치(100)는 패치 정보에 따른 매크로 또는 변수의 코드와 타겟 소프트웨어에 포함된 코드가 동일한 경우, 타겟 소프트웨어에 사용된 버전이 패치 정보와 동일하므로 취약점이 비전파된 것으로 탐지하고, 패치 정보에 따른 매크로 또는 변수의 코드와 타겟 소프트웨어에 포함된 코드가 동일하지 않은 경우 재사용된 버전의 오픈소스 소프트웨어에 포함된 취약점이 전파된 것으로 탐지할 수 있다.

【0080】S1054 단계에서, 장치(100)는 제3 해시 정보와 제4 해시 정보를 비교하여 타겟 소프트웨어에 포함된 함수 또는 구조체에 전파된 취약점을 탐지할 수 있다. 예를 들어, 장치(100)는 제3 해시 정보와 제4 해시 정보의 유사도가 동일한 것으로 판별된 경우, 타겟 소프트웨어에 사용된 버전이 패치 정보와 동일하므로 타겟 소프트웨어에 포함된 함수 또는 구조체에 취약점이 비전파된 것으로 탐지하고, 제3 해시 정보와 제4 해시 정보의 유사도가 동일하지 않은 경우 재사용된 버전의 오픈소스 소프트웨어에 포함된 함수 또는 구조체에 취약점이 전파된 것으로 탐지할 수 있다.

【0081】즉, 도 7 및 도 8의 실시예에서, 장치(100)는 매크로/변수에 비하여, 함수/구조체의 코드 길이가 길기 때문에, 모든 코드의 비교 시 연산 시간이 길 수 있다는 점에 착안하여, 함수 또는 구조체에 대해서는 타겟 소프트웨어의 코드와 패치 정보의 코드에 대해 각각 해시 정보를 생성하고 해시 정보를 기준으로 비교하여 패치 여부를 판별함으로써, 취약점의 전파 여부를 신속하게 수행할 수 있다.

【0082】한편, 현재 GitHub에서 널리 활용되는 10개의 소프트웨어(Turicreate, ReactOS, FreeBSD, MongoDB, Filament, TizenRT, Aseprite, MAME, Godot, 및 ArangoDB)에서 전파된 보안 취약점을 탐지하고자 시도하였을 때, 본 발명은 96%의 정밀도와 91%의 재현율을 보이며 총 137개의 전파된 보안 취약점을 10개의 타겟 소프트웨어로부터 탐지해냈다. 동일한 데이터셋에 대해 기존의 버전 기반 취약점 탐지 기술(CENTRIS)와 코드 기반 취약점 탐지 기술(VUDDY, MOVERY, VOFinder)를 접목해 보았을 때, CENTRIS는 23%의 정밀도와 68%의 재현율을, VUDDY는 62%의 정밀도와 51%의 재현율을 나타내었으며, MOVERY는 84%의 정밀도와 60%의 재현율을 나타내었고, VOFinder는 78%의 정밀도와 45%의 재현율을 나타내었다. 또한, 속도 및 확장성을 평가하기 위해, GitHub의 4,434개 C/C++ 소프트웨어를 대상으로 본 발명에서 제안하는 기술을 접목해 보았을 때, 99% 이상의 타겟 소프트웨어에 대해서 20초 이내에 전파된 보안 취약점을 탐지하였다.

【0083】상술한 실시예에 따르면, 본 발명은 오픈소스 소프트웨어로부터 전파된 취약점을 탐지할 대상인 타겟 소프트웨어와 오픈소스 소프트웨어에 포함된 함

수를 비교하여, 오픈소스 소프트웨어의 함수 중 타겟 소프트웨어에 동일하게 사용된 제1 함수, 수정되어 사용된 제2 함수, 비사용된 제3 함수를 분류할 수 있다. 즉, 본 발명은 타겟 소프트웨어가 오픈소스 소프트웨어의 코드 중 재사용한 부분과 재사용하지 않은 부분을 특정할 수 있고, 오픈소스 소프트웨어의 함수에 대한 수정이 발생한 부분을 명확히 특정할 수 있다. 이에 따라, 본 발명은 오픈소스 소프트웨어로부터 타겟 소프트웨어에 실제로 전파된 취약점을 탐지하여, 보안 취약점 탐지 시 오탐지 및 미탐지의 문제를 현저하게 감소시킬 수 있다.

【0084】 더하여, 본 발명은 오픈소스 소프트웨어에 내재된 취약점을 유발하는 코드의 패치 정보를 매크로, 변수, 함수, 구조체 별로 구분하여 비교할 수 있고, 나아가 함수 또는 구조체를 각각 해시화한 해시 정보를 비교함에 따라 코드 기반 취약점 탐지를 효율적으로 수행할 수 있다.

【0085】 이로써, 본 발명은 보안 취약점 탐지에 효과적으로 활용될 수 있을 뿐 아니라 소프트웨어의 공급망 보안에 응용될 수 있으며, 궁극적으로 안전한 소프트웨어 생태계를 구축할 수 있다.

【0086】 본 문서의 다양한 실시예들 및 이에 사용된 용어들은 본 문서에 기재된 기술적 특징들을 특정한 실시예들로 한정하려는 것이 아니며, 해당 실시예의 다양한 변경, 균등물, 또는 대체물을 포함하는 것으로 이해되어야 한다. 도면의 설명과 관련하여, 유사한 또는 관련된 구성요소에 대해서는 유사한 참조 부호가 사용될 수 있다. 아이템에 대응하는 명사의 단수 형은 관련된 문맥상 명백하게 다르게 지시하지 않는 한, 아이템 한 개 또는 복수 개를 포함할 수 있다.

【0087】본 문서에서, "A 또는 B", "A 및 B 중 적어도 하나", “A 또는 B 중 적어도 하나,” "A, B 또는 C," "A, B 및 C 중 적어도 하나,” 및 “A, B, 또는 C 중 적어도 하나"와 같은 문구들 각각은 그 문구들 중 해당하는 문구에 함께 나열된 항목들의 모든 가능한 조합을 포함할 수 있다. " 1", "제2", 또는 "첫째" 또는 "둘째"와 같은 용어들은 단순히 해당 구성요소를 다른 해당 구성요소와 구분하기 위해 사용될 수 있으며, 해당 구성요소들을 다른 측면(예: 중요성 또는 순서)에서 한정하지 않는다. 어떤(예: 제1) 구성요소가 다른(예: 제2) 구성요소에, “기능적으로” 또는 “통신적으로”라는 용어와 함께 또는 이런 용어 없이, “커플드” 또는 “커넥티드”라고 언급된 경우, 그것은 어떤 구성요소가 다른 구성요소에 직접적으로(예: 유선으로), 무선으로, 또는 제3 구성요소를 통하여 연결될 수 있다는 것을 의미한다.

【0088】본 문서에서 사용된 용어 "모듈"은 하드웨어, 소프트웨어 또는 펌웨어로 구현된 유닛을 포함할 수 있으며, 예를 들면, 로직, 논리 블록, 부품, 또는 회로 등의 용어와 상호 호환적으로 사용될 수 있다. 모듈은, 일체로 구성된 부품 또는 하나 또는 그 이상의 기능을 수행하는, 부품의 최소 단위 또는 그 일부가 될 수 있다. 예를 들면, 일 실시예에 따르면, 모듈은 ASIC(application-specific integrated circuit)의 형태로 구현될 수 있다.

【0089】본 문서의 다양한 실시예들은 기기(예: 전자 장치)에 의해 읽을 수 있는 저장 매체(예: 메모리)에 저장된 하나 이상의 명령어들을 포함하는 소프트웨어(예: 프로그램)로서 구현될 수 있다. 저장 매체는 RAM(random access memory),

메모리 버퍼, 하드 드라이브, 데이터베이스, EPROM(erasable programmable read-only memory), EEPROM(electrically erasable read-only memory), ROM(read-only memory) 및/또는 등등을 포함할 수 있다.

【0090】 또한, 본 문서의 실시예들의 프로세서는, 저장 매체로부터 저장된 하나 이상의 명령어들 중 적어도 하나의 명령을 호출하고, 그것을 실행할 수 있다. 이것은 기기가 호출된 적어도 하나의 명령어에 따라 적어도 하나의 기능을 수행하도록 운영되는 것을 가능하게 한다. 이러한 하나 이상의 명령어들은 컴파일러에 의해 생성된 코드 또는 인터프리터에 의해 실행될 수 있는 코드를 포함할 수 있다. 프로세서는 범용 프로세서, FPGA(Field Programmable Gate Array), ASIC(Application Specific Integrated Circuit), DSP(Digital Signal Processor) 및/또는 등등 일 수 있다.

【0091】 기기로 읽을 수 있는 저장매체는, 비일시적(non-transitory) 저장매체의 형태로 제공될 수 있다. 여기서, '비일시적' 은 저장매체가 실재(tangible)하는 장치이고, 신호(예: 전자기파)를 포함하지 않는다는 것을 의미할 뿐이며, 이 용어는 데이터가 저장매체에 반영구적으로 저장되는 경우와 임시적으로 저장되는 경우를 구분하지 않는다.

【0092】 본 문서에 개시된 다양한 실시예들에 따른 방법은 컴퓨터 프로그램 제품(computer program product)에 포함되어 제공될 수 있다. 컴퓨터 프로그램 제품은 상품으로서 판매자 및 구매자 간에 거래될 수 있다. 컴퓨터 프로그램 제품은 기기로 읽을 수 있는 저장 매체(예: compact disc read only memory (CD-ROM))의

형태로 배포되거나, 또는 어플리케이션 스토어(예: 플레이 스토어)를 통해 또는 두 개의 사용자 장치들(예: 스마트폰들) 간에 직접, 온라인으로 배포(예: 다운로드 또는 업로드)될 수 있다. 온라인 배포의 경우에, 컴퓨터 프로그램 제품의 적어도 일부는 제조사의 서버, 어플리케이션 스토어의 서버, 또는 서버의 메모리와 같은 기기로 읽을 수 있는 저장 매체에 적어도 일시 저장되거나, 임시적으로 생성될 수 있다.

【0093】 다양한 실시예들에 따르면, 기술한 구성요소들의 각각의 구성요소(예: 모듈 또는 프로그램)는 단수 또는 복수의 개체를 포함할 수 있다. 다양한 실시예들에 따르면, 전술한 해당 구성요소들 중 하나 이상의 구성요소들 또는 동작들이 생략되거나, 또는 하나 이상의 다른 구성요소들 또는 동작들이 추가될 수 있다. 대체적으로 또는 추가적으로, 복수의 구성요소들(예: 모듈 또는 프로그램)은 하나의 구성요소로 통합될 수 있다. 이런 경우, 통합된 구성요소는 복수의 구성요소들 각각의 구성요소의 하나 이상의 기능들을 통합 이전에 복수의 구성요소들 중 해당 구성요소에 의해 수행되는 것과 동일 또는 유사하게 수행할 수 있다. 다양한 실시예들에 따르면, 모듈, 프로그램 또는 다른 구성요소에 의해 수행되는 동작들은 순차적으로, 병렬적으로, 반복적으로, 또는 휴리스틱하게 실행되거나, 동작들 중 하나 이상이 다른 순서로 실행되거나, 생략되거나, 또는 하나 이상의 다른 동작들이 추가될 수 있다.

【부호의 설명】

【0094】100: 장치

110: 메모리

120: 프로세서

130: 입출력 인터페이스

140: 통신 인터페이스

【청구범위】

【청구항 1】

프로세서에 의해 동작하는 보안 취약점 전파 탐지 장치가 수행하는 동작의 방법에 있어서,

타겟 소프트웨어 및 상기 타겟 소프트웨어에 사용된 오픈소스 소프트웨어의 버전 정보를 획득하는 동작;

상기 버전의 오픈소스 소프트웨어에 포함된 함수 중 상기 타겟 소프트웨어에 동일하게 사용된 제1 함수, 수정되어 사용된 제2 함수, 비사용된 제3 함수를 분류하는 동작;

상기 제2 함수 및 상기 버전의 원본 함수 간의 제1 유사도와, 상기 제2 함수 및 상기 버전에 대한 패치 함수간의 제2 유사도를 기초로 상기 제2 함수의 패치 여부를 판별하는 동작; 및

상기 제1 함수, 상기 제2 함수 및 상기 제3 함수의 분류와 상기 제2 함수의 패치 여부에 기초하여 상기 버전의 오픈소스 소프트웨어에 내재된 취약점 중 상기 타겟 소프트웨어에 전파된 취약점을 탐지하는 동작을 포함하는,

방법.

【청구항 2】

제1항에 있어서,

상기 버전 정보는

상기 버전의 오픈소스 소프트웨어에 포함된 함수, 상기 버전에 따른 오픈소스 소프트웨어에 내재된 취약점과 취약점을 유발하는 함수, 취약점을 유발하는 함수를 보완하는 패치 함수에 대한 정보를 포함하는,

방법.

【청구항 3】

제1항에 있어서,

상기 분류하는 동작은

상기 타겟 소프트웨어에 포함된 함수를 해시화한 제1 해시 정보와 상기 오픈소스 소프트웨어에 포함된 함수를 해시화한 제2 해시 정보를 생성하는 동작;

상기 제1 해시 정보와 상기 제2 해시 정보 간의 유사도를 도출하는 동작; 및

상기 유사도에 대하여 기 설정된 범위를 기준으로 상기 오픈소스 소프트웨어에 포함된 함수를 제1 함수, 제2 함수 및 제3 함수 중 하나로 분류하는 동작을 포함하는

방법.

【청구항 4】

제1항에 있어서,

상기 패치 여부를 판별하는 동작은

상기 제1 유사도가 상기 제2 유사도보다 큰 경우 상기 제2 함수는 원본에 내재된 취약점이 패치되지 않은 함수인 것으로 판별하고, 상기 제2 유사도가 상기 제

1 유사도보다 큰 경우 상기 제2 함수는 원본에 내재된 취약점이 패치된 함수인 것으로 판별하는,

방법.

【청구항 5】

제4항에 있어서,

상기 취약점을 탐지하는 동작은

상기 오픈소스 소프트웨어에 내재된 취약점 중 상기 제1 함수에 대응하는 취약점이 전파되고, 상기 제2 함수 중 패치된 것으로 판별된 제2 함수의 취약점이 비전파되고, 상기 제2 함수 중 패치되지 않은 것으로 판별된 제2 함수의 취약점이 전파되고, 상기 제3 함수에 대응하는 취약점이 비전파된 것으로 탐지하는 동작을 포함하는,

방법.

【청구항 6】

제1항에 있어서,

상기 방법은

상기 오픈소스 소프트웨어에 내재된 취약점을 유발하는 코드의 패치 정보를 매크로, 변수, 함수, 구조체 별로 상기 타겟 소프트웨어와 비교하여 코드 기반으로 전파된 취약점을 탐지하는 동작을 더 포함하는,

방법.

【청구항 7】

제6항에 있어서,

상기 코드 기반으로 전파된 취약점을 탐지하는 동작은

상기 오픈소스 소프트웨어에 내재된 취약점을 유발하는 코드의 패치 정보를 매크로, 변수, 함수, 구조체 별로 구분하는 동작;

상기 타겟 소프트웨어에 포함된 함수 또는 구조체를 각각 해시화한 제3 해시 정보와 상기 패치 정보에 포함된 함수 또는 구조체를 각각 해시화한 제4 해시 정보를 생성하는 동작;

상기 패치 정보에 따른 매크로 또는 변수의 코드와 상기 타겟 소프트웨어에 포함된 매크로 또는 변수의 코드를 비교하여, 상기 타겟 소프트웨어에 포함된 매크로 또는 변수에 전파된 취약점을 탐지하는 동작; 및

상기 제3 해시 정보와 상기 제4 해시 정보를 비교하여 상기 타겟 소프트웨어에 포함된 함수 또는 구조체에 전파된 취약점을 탐지하는 동작을 포함하는

방법.

【청구항 8】

제7항에 있어서,

상기 매크로 또는 변수에 전파된 취약점을 탐지하는 동작은

상기 패치 정보에 따른 매크로 또는 변수의 코드와 상기 타겟 소프트웨어에 포함된 코드가 동일한 경우 취약점이 비전파된 것으로 탐지하고, 상기 패치 정보에

따른 매크로 또는 변수의 코드와 상기 타겟 소프트웨어에 포함된 코드가 동일하지 않은 경우 재사용된 버전의 오픈소스 소프트웨어에 포함된 취약점이 전파된 것으로 탐지하는 동작을 포함하는

방법.

【청구항 9】

제7항에 있어서,

상기 함수 또는 구조체에 전파된 취약점을 탐지하는 동작은

상기 제3 해시 정보와 상기 제4 해시 정보의 유사도가 동일한 것으로 판별된 경우 상기 타겟 소프트웨어에 포함된 함수 또는 구조체에 취약점이 비전파된 것으로 탐지하고, 상기 제3 해시 정보와 상기 제4 해시 정보의 유사도가 동일하지 않은 경우 재사용된 버전의 오픈소스 소프트웨어에 포함된 함수 또는 구조체에 취약점이 전파된 것으로 탐지하는 동작을 포함하는

방법.

【청구항 10】

명령어를 포함하는 메모리; 및

상기 명령어를 기초로 소정의 동작을 수행하는 프로세서를 포함하고,

상기 프로세서의 동작은,

타겟 소프트웨어 및 상기 타겟 소프트웨어에 사용된 오픈소스 소프트웨어의 버전 정보를 획득하는 동작;

상기 버전의 오픈소스 소프트웨어에 포함된 함수 중 상기 타겟 소프트웨어에 동일하게 사용된 제1 함수, 수정되어 사용된 제2 함수, 비사용된 제3 함수를 분류하는 동작;

상기 제2 함수 및 상기 버전의 원본 함수 간의 제1 유사도와, 상기 제2 함수 및 상기 버전에 대한 패치 함수간의 제2 유사도를 기초로 상기 제2 함수의 패치 여부를 판별하는 동작; 및

상기 제1 함수, 상기 제2 함수 및 상기 제3 함수의 분류와 상기 제2 함수의 패치 여부에 기초하여 상기 버전의 오픈소스 소프트웨어에 내재된 취약점 중 상기 타겟 소프트웨어에 전파된 취약점을 탐지하는 동작을 포함하는,

보안 취약점 전파 탐지 장치.

【요약서】

【요약】

일 실시예에 따른 보안 취약점 전파 탐지 장치는 타겟 소프트웨어 및 상기 타겟 소프트웨어에 사용된 오픈소스 소프트웨어의 버전 정보를 획득하는 동작; 상기 버전의 오픈소스 소프트웨어에 포함된 함수 중 상기 타겟 소프트웨어에 동일하게 사용된 제1 함수, 수정되어 사용된 제2 함수, 비사용된 제3 함수를 분류하는 동작; 상기 제2 함수 및 상기 버전의 원본 함수 간의 제1 유사도와, 상기 제2 함수 및 상기 버전에 대한 패치 함수간의 제2 유사도를 기초로 상기 제2 함수의 패치 여부를 판별하는 동작; 상기 제1 함수, 상기 제2 함수 및 상기 제3 함수의 분류와 상기 제2 함수의 패치 여부에 기초하여 상기 버전의 오픈소스 소프트웨어에 내재된 취약점 중 상기 타겟 소프트웨어에 전파된 취약점을 탐지하는 동작을 수행할 수 있다.

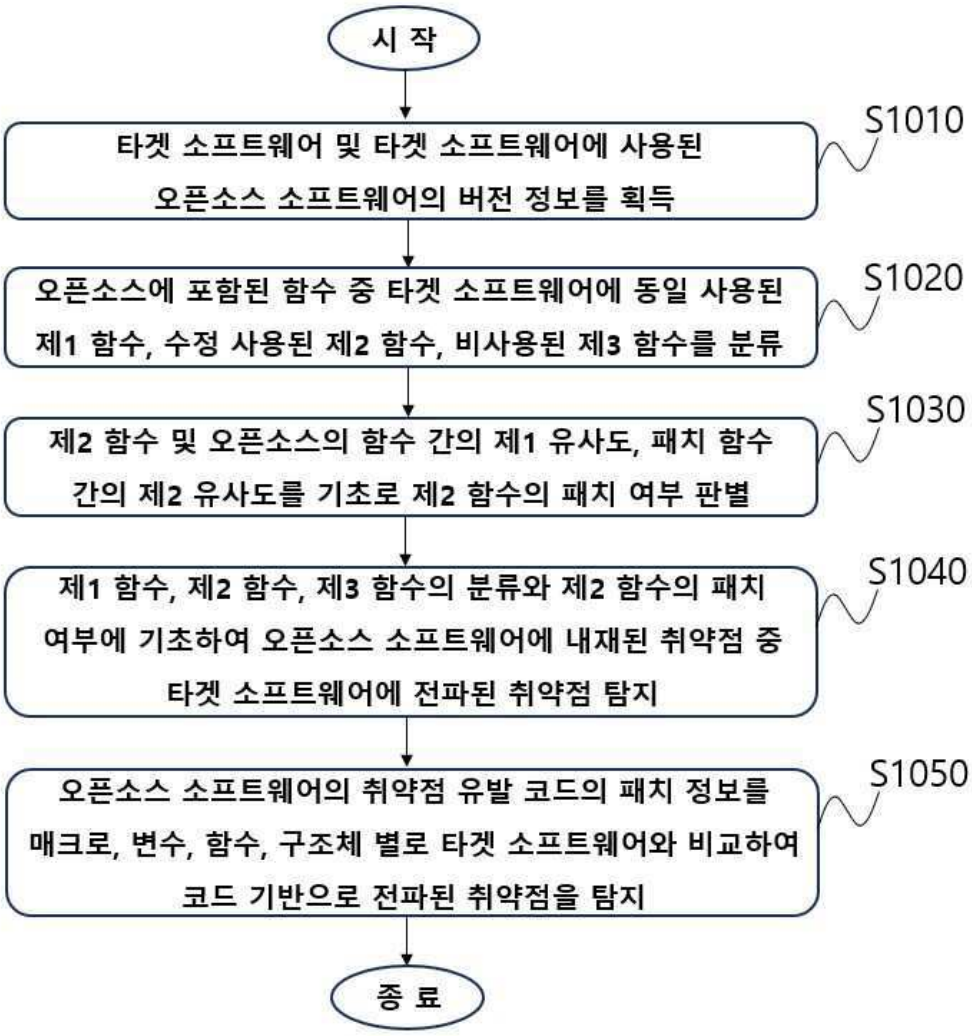
【대표도】

도 2

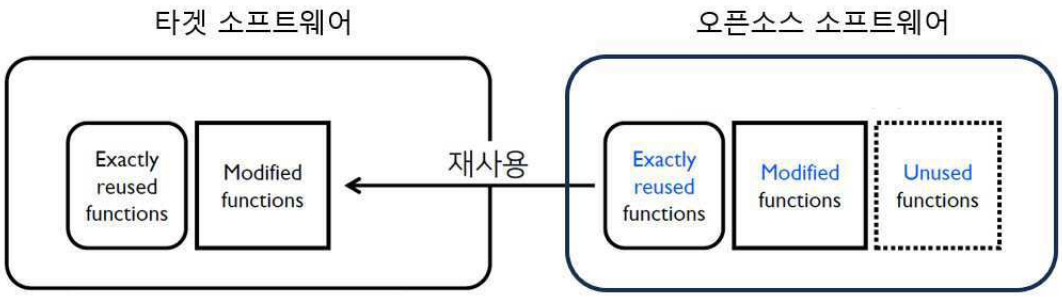
【도면】

【도 1】

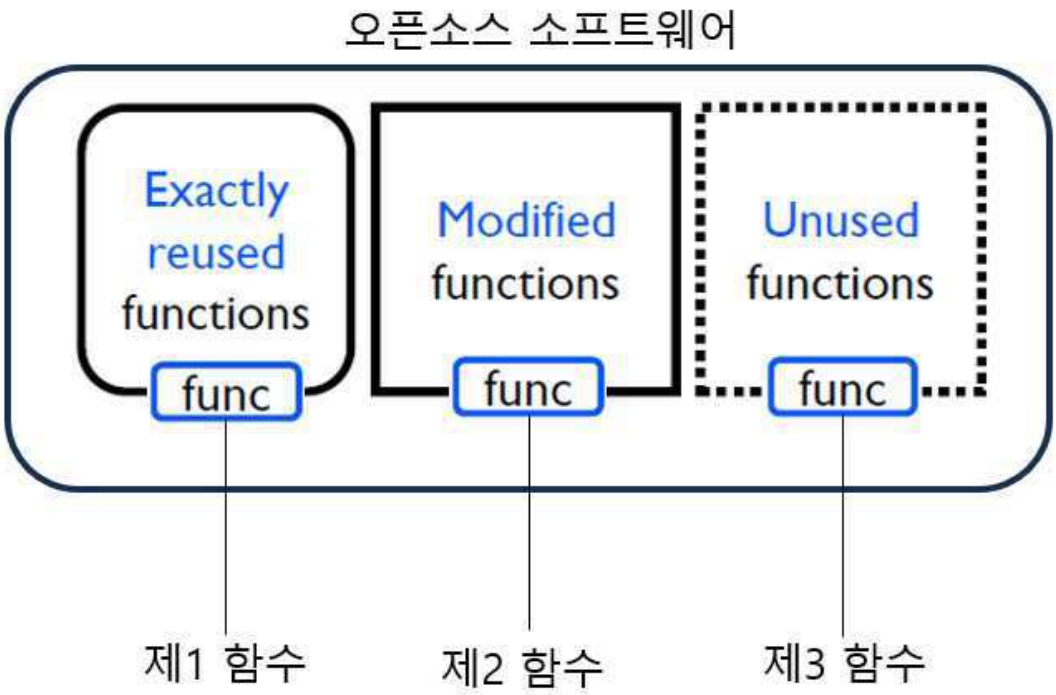
【도 2】


시 작
타겟 소프트웨어 및 타겟 소프트웨어에 사용된 오픈소스 소프트웨어의 버전 정보를 획득
S1010
오픈소스에 포함된 함수 중 타겟 소프트웨어에 동일 사용된 제1 함수, 수정 사용된 제2 함수, 비사용된 제3 함수를 분류
S1020
제2 함수 및 오픈소스의 함수 간의 제1 유사도, 패치 함수 간의 제2 유사도를 기초로 제2 함수의 패치 여부 판별
S1030
제1 함수, 제2 함수, 제3 함수의 분류와 제2 함수의 패치 여부에 기초하여 오픈소스 소프트웨어에 내재된 취약점 중 타겟 소프트웨어에 전파된 취약점 탐지
S1040
오픈소스 소프트웨어의 취약점 유발 코드의 패치 정보를 매크로, 변수, 함수, 구조체 별로 타겟 소프트웨어와 비교하여 코드 기반으로 전파된 취약점을 탐지
S1050
종 료

【도 3】


타겟 소프트웨어
오픈소스 소프트웨어
Exactly reused functions
Modified functions
재사용
Exactly reused functions
Modified functions
Unused functions

【도 4】


오픈소스 소프트웨어
Exactly reused functions
func
Modified functions
func
Unused functions
func
제1 함수
제2 함수
제3 함수

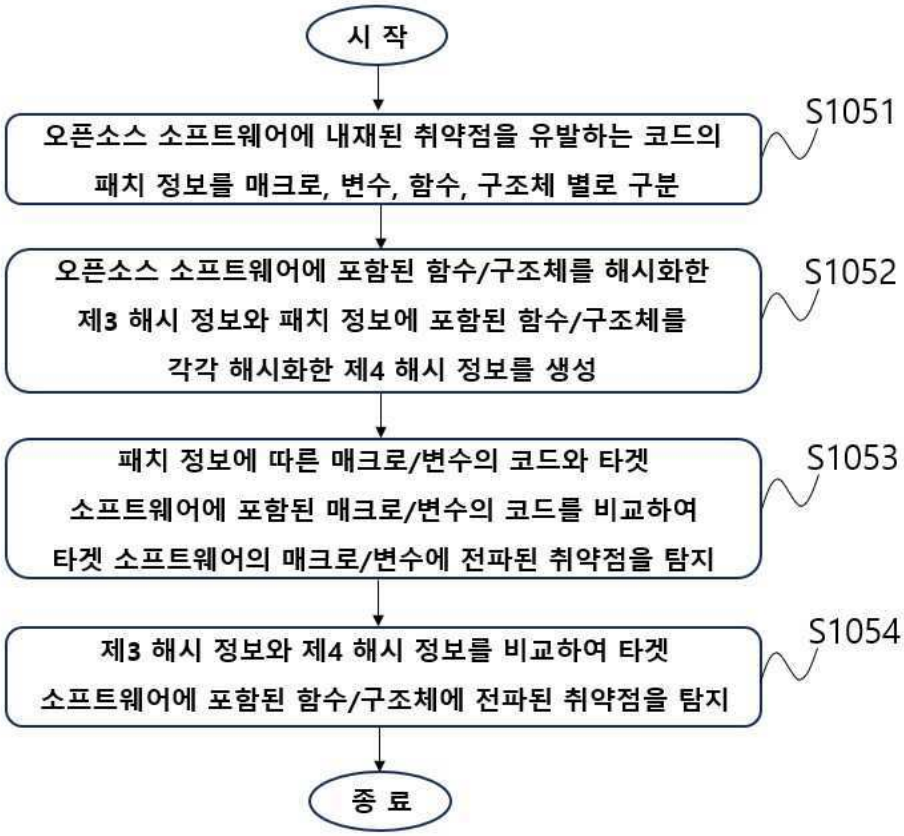
【도 5】

오픈소스 (ver 1.1.2) 취약점 정보	원본 함수	패치 함수
취약점 1	f1	f'1
취약점 2	f2	f'2
취약점 3	f3	f'3
...	...	...
...	...	...

【도 6】

타겟 소프트웨어 내의 제2 함수	원본 함수와의 유사도	패치 함수와의 유사도	제2 함수 패치 여부
t1	f1 - 80%	f'1 - 30%	패치 x
t2	f2 - 20%	f'2 - 90%	패치 o
t3	f3 - 30%	f'3 - 85%	패치 o
...	...	...	...
...	...	...	...

【도 7】


시 작
오픈소스 소프트웨어에 내재된 취약점을 유발하는 코드의 패치 정보를 매크로, 변수, 함수, 구조체 별로 구분
S1051
오픈소스 소프트웨어에 포함된 함수/구조체를 해시화한 제3 해시 정보와 패치 정보에 포함된 함수/구조체를 각각 해시화한 제4 해시 정보를 생성
S1052
패치 정보에 따른 매크로/변수의 코드와 타겟 소프트웨어에 포함된 매크로/변수의 코드를 비교하여 타겟 소프트웨어의 매크로/변수에 전파된 취약점을 탐지
S1053
제3 해시 정보와 제4 해시 정보를 비교하여 타겟 소프트웨어에 포함된 함수/구조체에 전파된 취약점을 탐지
S1054
종 료