관 인 생 략

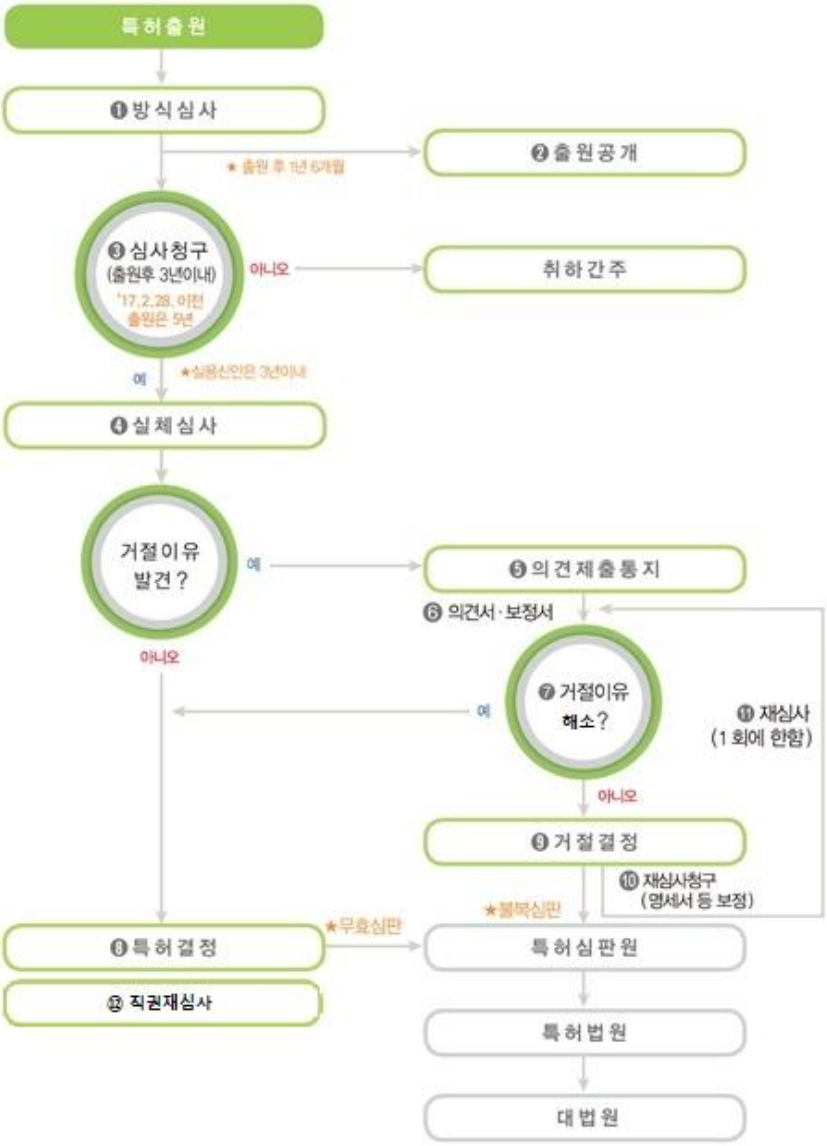
출 원 번 호 통 지 서

출 원 일 자	2022.10.25
특 기 사 항	심사청구(유) 공개신청(무) 참조번호(DPE22188)
출 원 번 호	10-2022-0138414 (접수번호 1-1-2022-1126830-76) (DAS접근코드0C23)
출원인 명칭	고려대학교 산학협력단(2-2004-017068-0)
대리인 성명	이대호(9-2009-003763-5)
발명자 성명	이희조 홍현지
발명의 명칭	취약점 데이터베이스를 구축하는 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


특허출원
❶방식심사
★출원 후 1년 6개월
❷출원공개
❸심사청구
(출원후 3년이내)
'17.2.28. 이전
출원은 5년
아니오
취하간주
예
★실용신안은 3년이내
❹실체심사
거절이유
발견?
예
❺의견제출통지
❻의견서·보정서
아니오
❼거절이유
해소?
예
⓫재심사
(1회에 한함)
아니오
❾거절결정
❿재심사청구
(명세서 등 보정)
★불복심판
★무효심판
❽특허결정
⓬직권재심사
특허심판원
특허법원
대법원

【발명의 설명】

【발명의 명칭】

취약점 데이터베이스를 구축하는 방법 및 장치{METHOD AND DEVICE FOR BUILDING VULNERABILITY DATABASE}

【기술분야】

【0001】 본 발명은 취약점 데이터베이스를 구축하는 방법 및 장치에 관한 것이다.

【발명의 배경이 되는 기술】

【0002】 오픈소스 소프트웨어(OSS)는 소스코드가 공개되어 있어 누구나 라이선스를 준수하는 범위 내에서 사용할 수 있는 소프트웨어를 지칭한다. 오픈소스 소프트웨어(OSS)는 소스코드를 재사용, 재배포, 개조를 자유롭게 허용함으로써 빠른 소프트웨어 개발에 기여할 수 있지만, OSS를 재사용 시 적절한 관리가 이루어지지 않으면 취약점 전파와 같은 문제를 일으킬 수 있다.

【0003】 이러한 문제를 해결하고자, 개발자들은 보통 공개된 보안 취약점 및 노출 리스트인 CVE 정보를 활용하여, 취약한 버전을 업데이트하거나 취약한 소스코드를 수정한다. 특히, 취약한 소스코드를 수정하기 위해서 보안 패치 및 취약 소스코드를 적재하고 있는 취약점 데이터베이스(Vulnerability Database)를 구축하는 것이 필요하다.

【0004】이러한 취약점 데이터베이스의 중요성에도 불구하고, 기존 연구들은 방대한 양의 보안 패치를 수집하는 것에 있어 두 가지 한계점을 가지고 있다. 구체적으로, 첫번째 한계점은 보안 패치가 수집되는 데이터소스가 한정적이라는 것이다. 두번째 한계점은 얕은 스캐닝을 통해서만 보안 패치가 수집된다는 점이다. 취약점 정보는 다양한 데이터 소스에 산재해 있지만 (예를 들어, 저장소(Repository) 및 이슈 트랙커(issue tracker) 등), 대부분의 기존 연구들은 저장소인'GitHub'에서만 보안 패치를 수집하는 방법만을 고려했다. 또한, 대부분의 기존 연구들은 취약점 공개 데이터베이스인 'NVD(National Vulnerability Database)'가 제공하는 참고 URL에서 직접 가져올 수 있는 패치만을 스캐닝하여 수집하기 때문에, 많은 보안 패치들이 수집되지 못했다.

【0005】따라서, 다양한 데이터소스를 고려하여 'NVD'에서 직접 가져올 수 없는 보안 패치까지도 수집할 수 있는 기술에 대한 수요가 존재한다.

【선행기술문헌】

【0006】(특허문헌 1) US 20200327237 A1

【0007】(비특허문헌 1) PatchDB (DSN, 2021): Xinda Wnag et.al. “PatchDB: A Large-Scale Security Patch Dataset“, In 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), IEEE, 2021

【발명의 내용】

【해결하고자 하는 과제】

【0008】 본 개시는 전술한 배경기술에 대응하여 안출 된 것으로, 직접 패치 링크, 간접 패치 링크, 및 비가시 패치 링크에 기반하여 보안 패치를 수집함으로써 취약점 데이터베이스를 구축하기 위한 방법 및 장치를 제공하기 위함이다.

【과제의 해결 수단】

【0009】 전술한 바와 같은 과제를 실현하기 위한 컴퓨팅 장치에 의해 수행되는 취약점 데이터베이스를 구축하는 방법이 개시된다. 상기 방법은: 직접 패치 링크(direct patch link)에 기초하여 데이터 소스로부터 보안 패치를 수집하는 단계; 간접 패치 링크(indirect patch link)에 기초하여 상기 데이터 소스로부터 보안 패치를 수집하는 단계; 및 비가시 패치 링크(invisible patch link)에 기초하여 상기 데이터 소스로부터 보안 패치를 수집하는 단계; 를 포함할 수 있다.

【0010】 대안적으로, 상기 직접 패치 링크에 기초하여 데이터 소스로부터 보안 패치를 수집하는 단계는: 취약점 정보 페이지 상에서 사전 결정된 패턴을 가지는 보안 패치 링크를 식별하는 단계; 및 상기 보안 패치 링크를 통해 연결되는 보안 패치 페이지로부터 보안 패치를 수집할 수 있다.

【0011】 대안적으로, 상기 사전 결정된 패턴은 취약점 데이터 소스 도메인 네임 정보 및 보안 패치 식별 문자열 정보를 포함할 수 있다.

【0012】 대안적으로, 상기 간접 패치 링크(indirect patch link)에 기초하여 상기 데이터 소스로부터 보안 패치를 수집하는 단계는: 취약점 정보 페이지 상에서

식별되는 웹사이트 주소를 크롤링(crawling)하는 단계; 사전 결정된 패턴을 가지는 보안 패치 링크 또는 사전 결정된 힌트 정보를 획득하는 단계; 및 상기 보안 패치 링크 또는 상기 사전 결정된 힌트 정보에 기초하여 보안 패치를 수집하는 단계;를 포함할 수 있다.

【0013】 대안적으로, 상기 보안 패치 링크 또는 상기 사전 결정된 힌트 정보에 기초하여 보안 패치를 수집하는 단계는: 상기 보안 패치 링크를 통해 연결되는 보안 패치 페이지로부터 보안 패치를 수집하는 단계; 를 포함할 수 있다.

【0014】 대안적으로, 상기 사전 결정된 패턴은 취약점 데이터 소스 도메인 네임 정보 및 보안 패치 식별 문자열 정보를 포함할 수 있다.

【0015】 대안적으로, 상기 보안 패치 링크 또는 상기 사전 결정된 힌트 정보에 기초하여 보안 패치를 수집하는 단계는: 상기 사전 결정된 힌트 정보에 대응하는 패치 커밋을 수집하는 단계를 포함할 수 있다.

【0016】 대안적으로, 상기 사전 결정된 힌트 정보는 커밋 ID 정보 또는 버그 ID 정보를 포함할 수 있다.

【0017】 대안적으로, 상기 비가시 패치 링크에 기초하여 상기 데이터 소스로부터 보안 패치를 수집하는 단계는: 질답 사이트로부터 질답 포스트를 수집하는 단계; 상기 수집된 질답 포스트의 변경 이력을 추출하는 단계; 상기 추출된 변경 이력로부터 사전 결정된 특징에 대응하는 변경 정보를 식별하는 단계; 및 상기 식별된 변경 정보에 기초하여 인시큐어 코드 스니펫(insecure code snippet)을 획득하

는 단계;를 포함할 수 있다.

【0018】 대안적으로, 상기 사전 결정된 특징은 보안-민감 API(security-sensitive API), 보안-관련 키워드(security-related keyword), 및 컨트롤 플로우(control flow)에서의 변경을 포함할 수 잇다.

【0019】 대안적으로, 상기 비가시 패치 링크에 기초하여 상기 데이터 소스로부터 보안 패치를 수집하는 단계는: 저장소 또는 이슈 트랙커에서 CVE ID 정보를 포함하는 커밋 메시지를 검색하는 단계;를 포함할 수 있다.

【0020】 대안적으로, 사전 결정된 특징에 기초하여 상기 검색된 커밋 메시지를 분석함으로써 상기 검색된 커밋 메시지로부터 보안 패치를 수집하는 단계;를 포함할 수 있다.

【0021】 전술한 바와 같은 과제를 실현하기 위한 컴퓨터 판독가능 매체에 저장된 컴퓨터 프로그램으로서, 상기 컴퓨터 프로그램은 하나 이상의 프로세서로 하여금 취약점 데이터베이스를 구축하는 방법을 수행하게 하기 위한 명령들을 포함하며, 상기 방법은: 직접 패치 링크(direct patch link)에 기초하여 데이터 소스로부터 보안 패치를 수집하는 단계; 간접 패치 링크(indirect patch link)에 기초하여 상기 데이터 소스로부터 보안 패치를 수집하는 단계; 및 비가시 패치 링크(invisible patch link)에 기초하여 상기 데이터 소스로부터 보안 패치를 수집하는 단계;를 포함할 수 있다.

【0022】 전술한 바와 같은 과제를 실현하기 위한 취약점 데이터베이스를 구축하는 방법을 수행하는 컴퓨팅 장치가 개시된다. 상기 컴퓨팅 장치는: 컴퓨터 실행가능한 컴포넌트들을 포함하는 메모리; 메모리에 저장된 이하의 컴퓨터 실행가능한 컴포넌트들을 실행하는 프로세서; 를 포함하고, 상기 프로세서는, 직접 패치 링크(direct patch link)에 기초하여 데이터 소스로부터 보안 패치를 수집하고, 간접 패치 링크(indirect patch link)에 기초하여 상기 데이터 소스로부터 보안 패치를 수집하고, 비가시 패치 링크(invisible patch link)에 기초하여 상기 데이터 소스로부터 보안 패치를 수집할 수 있다.

【발명의 효과】

【0023】 본 개시는 직접 패치 링크, 간접 패치 링크, 및 비가시 패치 링크에 기반하여 보안 패치를 수집함으로써 취약점 데이터베이스를 구축하기 위한 방법 및 장치를 제공할 수 있다.

【도면의 간단한 설명】

【0024】 도 1은 본 개시의 몇몇 실시예에 따른 취약점 데이터베이스를 구축하는 방법을 제공하기 위한 동작을 수행하는 컴퓨팅 장치의 블록 구성도를 도시한 도면이다.

도 2는 본 개시의 몇몇 실시예에 따른 취약점 데이터베이스를 구축하는 방법에 대한 개략도이다.

도 3은 본 개시의 몇몇 실시예에 따른 보안 패치 수집부의 동작을 설명하기

위한 개념도이다.

도 4는 본 개시의 몇몇 실시예에 따른 직접 패치 링크(direct patch link)를 이용하여 보안 패치를 수집하는 실시예를 설명하기 위한 도면이다.

도 5는 본 개시의 몇몇 실시예에 따른 간접 패치 링크(indirect patch link)를 이용하여 보안 패치를 수집하는 실시예를 설명하기 위한 도면이다.

도 6은 본 개시의 몇몇 실시예에 따른 간접 패치 링크를 이용하여 보안 패치를 수집하는 실시예를 설명하기 위한 다른 도면이다.

도 7은 본 개시의 몇몇 실시예에 따른 비가시 패치 링크(invisible patch link)를 이용하여 보안 패치를 수집하는 실시예를 설명하기 위한 도면이다.

도 8은 본 개시의 몇몇 실시예에 따른 취약점 데이터베이스를 구축하는 방법의 순서도이다.

도 9는 본 개시의 몇몇 실시예에 따른 컴퓨팅 장치의 블록 구성도이다.

【발명을 실시하기 위한 구체적인 내용】

【0025】 다양한 실시예들이 이제 도면을 참조하여 설명된다. 본 명세서에서, 다양한 설명들이 본 개시의 이해를 제공하기 위해서 제시된다. 그러나, 이러한 실시예들은 이러한 구체적인 설명 없이도 실행될 수 있음이 명백하다.

【0026】 본 명세서에서 사용되는 용어 "컴포넌트", "모듈", "시스템" 등은 컴퓨터-관련 엔티티, 하드웨어, 펌웨어, 소프트웨어, 소프트웨어 및 하드웨어의 조합, 또는 소프트웨어의 실행을 지칭한다. 예를 들어, 컴포넌트는 프로세서상에서

실행되는 처리과정(procedure), 프로세서, 객체, 실행 스레드, 프로그램, 및/또는 컴퓨터일 수 있지만, 이들로 제한되는 것은 아니다. 예를 들어, 컴퓨팅 장치에서 실행되는 애플리케이션 및 컴퓨팅 장치 모두 컴포넌트일 수 있다. 하나 이상의 컴포넌트는 프로세서 및/또는 실행 스레드 내에 상주할 수 있다. 일 컴포넌트는 하나의 컴퓨터 내에 로컬화 될 수 있다. 일 컴포넌트는 2개 이상의 컴퓨터들 사이에 분배될 수 있다. 또한, 이러한 컴포넌트들은 그 내부에 저장된 다양한 데이터 구조들을 갖는 다양한 컴퓨터 판독가능한 매체로부터 실행할 수 있다. 컴포넌트들은 예를 들어 하나 이상의 데이터 패킷들을 갖는 신호(예를 들면, 로컬 시스템, 분산 시스템에서 다른 컴포넌트와 상호작용하는 하나의 컴포넌트로부터의 데이터 및/또는 신호를 통해 다른 시스템과 인터넷과 같은 네트워크를 통해 전송되는 데이터)에 따라 로컬 및/또는 원격 처리들을 통해 통신할 수 있다.

【0027】 더불어, 용어 "또는"은 배타적 "또는"이 아니라 내포적 "또는"을 의미하는 것으로 의도된다. 즉, 달리 특정되지 않거나 문맥상 명확하지 않은 경우에, "X는 A 또는 B를 이용한다"는 자연적인 내포적 치환 중 하나를 의미하는 것으로 의도된다. 즉, X가 A를 이용하거나; X가 B를 이용하거나; 또는 X가 A 및 B 모두를 이용하는 경우, "X는 A 또는 B를 이용한다"가 이들 경우들 어느 것으로도 적용될 수 있다. 또한, 본 명세서에 사용된 "및/또는"이라는 용어는 열거된 관련 아이템들 중 하나 이상의 아이템의 가능한 모든 조합을 지칭하고 포함하는 것으로 이해되어야 한다.

【0028】 또한, "포함한다" 및/또는 "포함하는"이라는 용어는, 해당 특징 및/또는 구성요소가 존재함을 의미하는 것으로 이해되어야 한다. 다만, "포함한다" 및/또는 "포함하는"이라는 용어는, 하나 이상의 다른 특징, 구성요소 및/또는 이들의 그룹의 존재 또는 추가를 배제하지 않는 것으로 이해되어야 한다. 또한, 달리 특정되지 않거나 단수 형태를 지시하는 것으로 문맥상 명확하지 않은 경우에, 본 명세서와 청구범위에서 단수는 일반적으로 "하나 또는 그 이상"을 의미하는 것으로 해석되어야 한다.

【0029】 그리고, "A 또는 B 중 적어도 하나"이라는 용어는, "A만을 포함하는 경우", "B 만을 포함하는 경우", "A와 B의 구성으로 조합된 경우"를 의미하는 것으로 해석되어야 한다.

【0030】 당업자들은 추가적으로 여기서 개시된 실시예들과 관련되어 설명된 다양한 예시적 논리적 블록들, 구성들, 모듈들, 회로들, 수단들, 로직들, 및 알고리즘 단계들이 전자 하드웨어, 컴퓨터 소프트웨어, 또는 양쪽 모두의 조합들로 구현될 수 있음을 인식해야 한다. 하드웨어 및 소프트웨어의 상호교환성을 명백하게 예시하기 위해, 다양한 예시적 컴포넌트들, 블록들, 구성들, 수단들, 로직들, 모듈들, 회로들, 및 단계들은 그들의 기능성 측면에서 일반적으로 위에서 설명되었다. 그러한 기능성이 하드웨어로 또는 소프트웨어로서 구현되는지 여부는 전반적인 시스템에 부과된 특정 어플리케이션(application) 및 설계 제한들에 달려 있다. 숙련된 기술자들은 각각의 특정 어플리케이션들을 위해 다양한 방법들로 설명된 기능성을 구현할 수 있다. 다만, 그러한 구현의 결정들이 본 개시내용의 영역을 벗어나게

하는 것으로 해석되어서는 안 된다.

【0031】 제시된 실시예들에 대한 설명은 본 개시의 기술 분야에서 통상의 지식을 가진 자가 본 발명을 이용하거나 또는 실시할 수 있도록 제공된다. 이러한 실시예들에 대한 다양한 변형들은 본 개시의 기술 분야에서 통상의 지식을 가진 자에게 명백할 것이다. 여기에 정의된 일반적인 원리들은 본 개시의 범위를 벗어남이 없이 다른 실시예들에 적용될 수 있다. 그리하여, 본 발명은 여기에 제시된 실시예들로 한정되는 것이 아니다. 본 발명은 여기에 제시된 원리들 및 신규한 특징들과 일관되는 최광의의 범위에서 해석되어야 할 것이다.

【0032】 도 1은 본 개시의 몇몇 실시예에 따른 의료 이미지 분석 방법을 수행하기 위한 컴퓨팅 장치의 블록 구성도이다.

【0033】 도 1에서 도시되는 바와 같이, 컴퓨팅 장치(100)는 프로세서(110), 메모리(120), 네트워크부(130)를 포함할 수 있다. 도 1에 도시된 컴퓨팅 장치(100)의 구성은 간략화 하여 나타낸 예시일 뿐이다. 본 개시의 몇몇 실시예에서 컴퓨팅 장치(100)는 컴퓨팅 장치(100)의 컴퓨팅 환경을 수행하기 위한 다른 구성들을 포함할 수 있고, 개시된 구성들 중 일부만이 컴퓨팅 장치(100)를 구성할 수도 있다.

【0034】 프로세서(110)는 하나 이상의 코어로 구성될 수 있으며, 컴퓨팅 장치의 중앙 처리 장치(CPU: central processing unit), 범용 그래픽 처리 장치(GPGPU: general purpose graphics processing unit), 텐서 처리 장치(TPU: tensor processing unit) 등의 데이터 분석 및 처리를 위한 프로세서를 포함할 수 있다.

프로세서(110)는 메모리(120)에 저장된 컴퓨터 프로그램을 판독하여 본 개시의 몇몇 실시예에 따라 프로세서(110)는 취약점 데이터베이스를 구축하는 방법을 수행하기 위한 보안 패치 수집부 및 이의 모듈들을 구현할 수 있다. 또한, 프로세서(110)는 본 개시의 몇몇 실시예에 따른 취약점 데이터베이스를 구축하는 방법을 수행하기 위한 데이터 변환, 연산, 생성 등을 수행할 수 있다. 또한, 본 개시의 몇몇 실시예에 따른 컴퓨팅 장치에서 수행되는 컴퓨터 프로그램은 CPU, GPGPU 또는 TPU 실행가능 프로그램일 수 있다.

【0035】본 개시의 몇몇 실시예에 따르면, 메모리(120)는 프로세서(110)가 생성하거나 결정한 임의의 형태의 정보 및 네트워크부(130)가 수신한 임의의 형태의 정보를 저장할 수 있다. 메모리(120)는 프로세서(110)에 의해 취약점 데이터베이스를 구축하는 방법을 수행하는 과정에서 생성되는 데이터들을 저장할 수 있다. 예를 들어, 메모리(120)는 본 개시의 몇몇 실시예에 따라 취약점 데이터베이스를 구축하는 방법에 따라 생성되는 보안 패치를 저장할 수 있다. 또한, 메모리(120)는 프로세서(110)에 의해 취약점 데이터베이스를 구축하는 방법을 수행하는 과정에서 외부에서 수신되는 데이터들을 저장할 수 있다. 예를 들어, 메모리(120)는 프로세서(110)에 의해 취약점 데이터베이스를 구축하는 방법을 수행하는 과정에서 데이터 소스(data source)로부터 수신한 데이터를 저장할 수 있다. 다만, 이에 한정되지 않고, 메모리(120)는 본 개시의 몇몇 실시예에 따른 의료 이미지 분석 방법을 수행하기 위한 다양한 정보를 저장할 수 있다.

【0036】본 개시의 몇몇 실시예에 따르면, 메모리(120)는 플래시 메모리 타입(flash memory type), 하드디스크 타입(hard disk type), 멀티미디어 카드 마이크로 타입(multimedia card micro type), 카드 타입의 메모리(예를 들어 SD 또는 XD 메모리 등), 램(Random Access Memory, RAM), SRAM(Static Random Access Memory), 롬(Read-Only Memory, ROM), EEPROM(Electrically Erasable Programmable Read-Only Memory), PROM(Programmable Read-Only Memory), 자기 메모리, 자기 디스크, 광디스크 중 적어도 하나의 타입의 저장매체를 포함할 수 있다. 컴퓨팅 장치(100)는 인터넷(internet) 상에서 상기 메모리(120)의 저장 기능을 수행하는 웹 스토리지(web storage)와 관련되어 동작할 수도 있다. 전술한 메모리에 대한 기재는 예시일 뿐, 본 개시는 이에 제한되지 않는다.

【0037】 본 개시의 몇몇 실시예에 따른 네트워크부(130)는 임의의 형태의 공지된 유무선 통신 시스템을 사용할 수 있다.

【0038】네트워크부(130)는 프로세서(110)에 의해 처리된 정보, 사용자 인터페이스 등을 타 단말과의 통신을 통해 송수신할 수 있다. 예를 들어, 네트워크부(130)는 프로세서(110)에 의해 생성된 사용자 인터페이스를 클라이언트(e.g. 사용자 단말)로 제공할 수 있다. 또한, 네트워크부(130)는 클라이언트로 인가된 사용자의 외부 입력을 수신하여 프로세서(110)로 전달할 수 있다. 이때, 프로세서(110)는 네트워크부(130)로부터 전달받은 사용자의 외부 입력을 기초로 사용자 인터페이스를 통해 제공되는 정보의 출력, 수정, 변경, 추가 등의 동작을 처리할 수 있다.

【0039】구체적으로 예를 들면, 네트워크부(130)는 본 개시의 몇몇 실시예에 따른 취약점 데이터베이스를 구축하는 방법을 수행하기 위한 다양한 정보를 송수신할 수 있다. 예를 들어, 네트워크부(130)는 다양한 저장소, 이슈 트랙커, 질답 사이트를 포함하는 데이터 소스 상에 저장된 보안 이슈 관련 데이터를 수신할 수 있다. 또한, 네트워크부(130)는 본 개시의 몇몇 실시예에 따른 취약점 데이터베이스를 구축하는 방법을 수행하는 과정에서 생성되는 몇몇의 데이터를 취약점 데이터베이스 상에 저장하기 위해 외부로 전송할 수 있다.

【0040】한편, 본 개시의 몇몇 실시예에 따른 컴퓨팅 장치(100)는 클라이언트와 통신을 통해 정보를 송수신하는 컴퓨팅 시스템으로서 서버를 포함할 수 있다. 이때, 클라이언트는 서버에 엑세스할 수 있는 임의의 형태의 단말일 수 있다. 예를 들어, 서버인 컴퓨팅 장치(100)는 사용자 단말로부터 쿼리를 수신하여 쿼리에 대응하는 단일 정보 처리 결과를 생성할 수 있다. 이 경우에, 서버인 컴퓨팅 장치(100)는 처리 결과를 포함하는 사용자 인터페이스를 사용자 단말로 제공할 수 있다. 이때, 사용자 단말은 서버인 컴퓨팅 장치(100)로부터 수신한 사용자 인터페이스를 출력하고, 사용자와의 상호 작용을 통해 정보를 입력받거나 처리할 수 있다.

【0041】추가적인 실시예에서, 컴퓨팅 장치(100)는 임의의 서버에서 생성된 데이터 리소스를 전달받아 추가적인 정보 처리를 수행하는 임의의 형태의 단말을 포함할 수도 있다.

【0042】본 개시는 다양한 데이터 소스로부터 보안 패치를 수집함으로써 취약점 데이터베이스가 하나의 데이터 소스만을 고려함으로써 발생하는 편향되고 불

충분 데이터셋을 가지는 문제점을 해소할 수 있는 방법 및 장치를 제공할 수 있다. 본 개시는 취약점 정보를 명시적으로 제공하지 않는 데이터소스로부터 보안 패치를 수집할 수 있다. 본 개시는 패치 커밋 URL과 대응하는 CVE 취약점 간에 직접적인 연결성이 없는 경우에(invisible link) 보안 패치를 수집할 수 있다. 결과적으로, 본 개시는 다양한 데이터 소스로부터 자동화된 방법론으로 보안 패치를 수집할 수 있는 방법 및 장치를 제공할 수 있다.

【0043】 도 2는 본 개시의 몇몇 실시예에 따른 취약점 데이터베이스를 구축하는 방법에 대한 개략도이다.

【0044】 프로세서(110)는 데이터 소스(200) 상에서 알려지거나 또는 알려지지 않은 취약점에 대한 보안 패치(security patch)를 수집할 수 있다. 여기서 알려진 취약점(known vulnerability)은 CVE ID가 할당됨으로써 공용 취약점 데이터베이스에 의해 관리되는 취약점을 포함할 수 있다. 알려지지 않은 취약점(unknown vulnerability)은 CVE에 의해 관리되지 않는 취약점을 포함할 수 있다. 예를 들어, 알려지지 않은 취약점은 비밀리(secretly)에 패치되며 공용 취약점 데이터베이스에 의해 관리되지 않는 취약점을 포함할 수 있다.

【0045】 본 개시의 몇몇 실시예에 따라, 제한적인 데이터 소스에 의한 제약을 피하기 위해, 프로세서(110)는 3개의 유형의 데이터 소스(200)로부터 보안 패치(400)를 수집할 수 있다. 예를 들어, 데이터 소스(200)는 저장소(repository)들, 이슈 트랙커(issue tracker)들, 및 질답 사이트(Q&A site)를 포함할 수 있다.

【0046】 몇몇 예에서, 저장소는 소프트웨어 프로그램들의 다양한 버전들의

파일들의 컬렉션을 저장하기 위한 보관 장소일 수 있다. 저장소는 소프트웨어 코드들을 관리하기 위한 사이트일 수 있다. 예를 들어, 저장소는 'GitHub'를 포함할 수 있다.

【0047】 몇몇 예에서, 이슈 트래커는 소프트웨어 공급업체의 버그를 트랙킹하고 기타 이슈들을 관리하기 위한 툴(tool)들을 포함할 수 있다. 이슈 트래커는 보안 관련 버그 및 취약점(vulnerability)을 관리하는 사이트일 수 있다. 예를 들어, 이슈 트래커는 'Bugzilla'를 포함할 수 있다.

【0048】 몇몇 예에서, 질답 사이트는 코드 문제를 논의하는 플랫폼을 포함할 수 있다. 질답 사이트는 NVD(National Vulnerability Database)와 같은 공용 취약점 데이터베이스에 의해 관리되지 않지만 인시큐어 코드 스니펫(insecure code snippet)들이 생성되고 전파되는 사이트일 수 있다. 예를 들어, 질답 사이트는 'Stack Overflow'를 포함할 수 있다.

【0049】 상술한 데이터 소스(200)의 예는 단지 예시에 불과하고, 데이터 소스(200)는 다양한 저장소, 이슈 트랙커, 및 질답 사이트를 포함할 수 있다.

【0050】 프로세서(110)는 데이터 소스(200)로부터 보안 패치(400)를 수집하기 위한 다양한 동작을 수행하는 보안 패치 수집부(300)를 구현할 수 있다. 몇몇 예에서, 보안 패치 수집부(300)는 데이터 소스(200) 상에 저장된 CVE 정보와 같은 보안 관련 정보를 직접 또는 간접적으로 이용하여 보안 패치를 수집할 수 있다. 또한, 보안 패치 수집부(300)는 CVE 정보로부터 보이지 않지만(invisible) 데이터 소스(200)로부터 획득할 수 있는 보안 패치를 수집할 수 있다. 몇몇 예에서, 보안 패

치 수집부(300)는 직접 패치 링크 기반 수집 모듈(310), 간접 패치 링크 기반 수집 모듈(320), 미가시(invisible) 패치 링크 기반 수집 모듈(330)을 포함할 수 있다. 직접 패치 링크 기반 수집 모듈(310), 간접 패치 링크 기반 수집 모듈(320), 미가시 패치 링크 기반 수집 모듈(330)을 포함하는 보안 패치 수집부(300)의 예시적인 동작들은 도 3 내지 도 8을 참조하여 이하에서 자세히 설명된다.

【0051】프로세서(110)는 보안 패치 수집부(300)에 의해 데이터 소스(200)로부터 보안 패치(400)를 수집할 수 있다. 몇몇 예에서, 보안 패치(400)는 보안 이슈들을 해결하기 위해 적용되는 소스코드-레벨 패치(source code-level patch)를 포함할 수 있다. 예를 들어, 보안 패치(400)는 패치 적용 전후의 코드의 'diff' 형태로 제공될 수 있다. 표 1에 도시되는 예시적인 보안 패치(400)는 텐서플로우에서 힙 버퍼 오버플로우 취약점(heap buffer overow vulnerability)에 관한 취약점표준 코드 'CVE-2021-41216'에 대한 보안 패치 단편을 도시한다.

【0052】**【표 1】**

```
diff --git a/tensorflow/core/ops/array_ops.cc
           b/tensorflow/core/ops/array_ops.cc
index 64bd4f3847854..14c9efae1ddd3 100644
--- a/tensorflow/core/ops/array_ops.cc
+++ b/tensorflow/core/ops/array_ops.cc

@@ -168,7 +168,7 @@ Status TransposeShapeFn(...) {

   for (int32_t i = 0; i < rank; ++i) {
      int64_t in_idx = data[i];
-  if (in_idx >= rank) {
+  if (in_idx >= rank || in_idx <= -rank) {
      return errors::InvalidArgument("perm dim ",
           in_idx, " is out of range of input rank ",
           rank);
}
```

【0053】보안 패치(400)는 효율적인 취약점 관리를 위한 정보를 제공할 수 있다. 예를 들어, 보안 패치(400)는 취약점을 가지고 패치된 소스 파일에 대한 정보를 제공할 수 있다(예를 들어, 표 1에서 “array_ops.cc”). 보안 패치(400)는 패치 적용 전후의 파일 인덱스 값에 대한 정보를 제공할 수 있다(예를 들어, 표 1에서 3번 라인). 또한, 보안 패치(400)는 패치가 적용된 코드 라인 넘버에 대한 정보를 제공할 수 있다(예를 들어, 표 1에서, “array_ops.cc” 파일에서 168번 라인으로부터 7개의 라인). 보안 패치(400)는 보안 패치에서 추가 또는 삭제된 실제 코드 라인들에 대한 정보를 제공할 수 있다(예를 들어, 표 1에서, 11번 라인 및 12번 라인). 다만 이에 한정되지 않고, 보안 패치(400)는 다양한 형태로 다양한 정보를 제공할 수 있다.

【0054】프로세서(110)는 보안 패치 수집부(300)에 의해 수집된 보안 패치(400)를 취약점 데이터베이스 상에 저장할 수 있다. 본 개시의 몇몇 실시예에 따른 방법 및 장치에 따라 구축된 예시적인 취약점 데이터베이스는 'NVD'와 같은 취약점 공개 데이터베이스로부터 수집할 수 있는 보안 패치 뿐만 아니라 취약점 공개 데이터베이스로부터 수집할 수 없던 보안 패치를 보유하여 넓은 커버리지를 가질 수 있다. 본 개시의 몇몇 실시예에 따른 취약점 데이터베이스는 취약점 공개 데이터베이스로부터 직접적으로 획득할 수 없던 다수의 보안 패치를 수집함으로써 기존 방식으로 탐지할 수 없던 취약점 존재 여부를 탐지 가능하게 하여, 소프트웨어 보안성을 향상시키는 효과를 제공할 수 있다.

【0055】 이하에서, 도 3 내지 도 7을 참조하여 본 개시의 몇몇 실시예에 따른 취약점 데이터베이스를 구축하는 예시적인 실시예가 설명된다.

【0056】 도 3은 본 개시의 몇몇 실시예에 따른 보안 패치 수집부의 동작을 설명하기 위한 개념도이다. 도 4는 본 개시의 몇몇 실시예에 따른 직접 패치 링크(direct patch link)를 이용하여 보안 패치를 수집하는 실시예를 설명하기 위한 도면이다. 도 5는 본 개시의 몇몇 실시예에 따른 간접 패치 링크(indirect patch link)를 이용하여 보안 패치를 수집하는 실시예를 설명하기 위한 도면이다. 도 6은 본 개시의 몇몇 실시예에 따른 간접 패치 링크를 이용하여 보안 패치를 수집하는 실시예를 설명하기 위한 다른 도면이다. 도 7은 본 개시의 몇몇 실시예에 따른 비가시 패치 링크(invisible patch link)를 이용하여 보안 패치를 수집하는 실시예를 설명하기 위한 도면이다.

【0057】 보안 패치 수집부(300)는 데이터 소스(200)에서 제공되는 취약점 정보 페이지로부터 보안 패치를 수집할 수 있는 링크 정보를 획득할 수 있다. 보안 패치 수집부(300)는 수집한 링크 정보를 이용하여 다양한 데이터 소스(200)의 보안 패치 페이지 또는 보안 관련 정보 페이지로부터 보안 패치를 수집할 수 있다. 이

【0058】 몇몇 예에서, 취약점 정보 페이지는 NVD, CVE MITRE와 같은 공용 취약점 데이터베이스(public vulnerability database)에서 취약점에 대한 정보를 제공하는 페이지를 포함할 수 있다. 예를 들어, 취약점 정보 페이지는 공용 취약점 데이터베이스에서 각각의 CVE에 대한 정보를 제공하는 페이지를 포함할 수 있다. 본 명세서에서, 취약점 정보 페이지는 CVE 정보 페이지(CVE info page)로도 지칭될

수 있다.

【0059】보안 패치 페이지는 취약점에 대한 보안 패치를 제공하는 페이지를 포함할 수 있다. 예를 들어, 보안 패치 페이지는 CVE 취약점에 대한 보안 패치를 제공하는 페이지를 포함할 수 있다. 본 명세서에서, 보안 패치 페이지는 CVE 패치 페이지(CVE patch page)로도 지칭될 수 있다.

【0060】보안 패치 수집부(300)는 본 개시의 몇몇 실시예에 따라 정의되는 3가지 유형의 링크를 이용하여 데이터 소스(200)로부터 보안 패치(400)를 수집할 수 있다. 도 3을 참조하면, 예시적인 3가지 유형의 링크는 직접 패치 링크, 간접 패치 링크, 미가시 패치 링크를 포함할 수 있다. 몇몇 예에서, 보안 패치 수집부(300)는 취약점 정보 페이지 및 취약점 정보 페이지에 참조되는 웹페이지를 파싱 또는 크롤링함으로써 직접 패치 링크, 간접 패치 링크, 미가시 패치 링크에 대한 정보를 획득할 수 있다. 보안 패치 수집부(300)의 3개의 모듈(310, 320, 330)은 취약점 정보 페이지로부터 직접 패치 링크, 간접 패치 링크, 미가시 패치 링크를 획득함으로써 보안 패치를 수집할 수 있다. 몇몇 예에서, 직접 패치 링크는 취약점 정보 페이지로부터 보안 패치 페이지로 직접 연결되는 링크를 포함할 수 있다. 몇몇 예에서, 간접 패치 링크는 취약점 정보 페이지로부터 연결되는 웹페이지에서 보안 패치 페이지로 연결되는 링크 또는 패치 커밋을 검색할 수 있는 힌트 정보를 포함할 수 있다. 몇몇 예에서, 비가시 패치 링크는 보안 패치 페이지로 연결되는 링크 아닌 보안 패치를 수집할 수 있는 링크 및 정보를 포함할 수 있다. 각각의 패치 링크 유형에 기반한 모듈의 동작이 이하에서 설명된다.

【0061】본 개시의 몇몇 실시예에 따라, 직접 패치 링크 기반 수집 모듈(310)은 직접 패치 링크(direct patch link)에 기초하여 데이터 소스부터 보안 패치를 수집할 수 있다.

【0062】구체적으로 설명하면, 직접 패치 링크 기반 수집 모듈(310)은 취약점 정보 페이지 상에서 사전 결정된 패턴을 가지는 보안 패치 링크를 식별할 수 있다. 도 4를 참조하면, 직접 패치 링크 기반 수집 모듈(310)은 NVD 공용 취약점 데이터베이스의 취약점 정보 페이지(600) 상에서 사전 결정된 패턴을 가지는 보안 패치 링크(610)를 식별할 수 있다. 여기서, 사전 결정된 패턴은 취약점 데이터 소스 도메인 네임 정보 및 보안 패치 식별 문자열 정보를 포함할 수 있다. 보안 패치 링크(610)는 취약점 데이터소스 도메인 네임 정보 'github.com' 및 보안 패치 식별 문자열 정보 'commit'을 포함한다. 이 경우에, 직접 패치 링크 기반 수집 모듈(310)은 보안 패치 링크(610)를 통해 저장소(210)의 보안 패치 페이지(700)로 바로 접속할 수 있다. 직접 패치 링크 기반 수집 모듈(310)은 보안 패치 링크(610)를 통해 접속되는 보안 패치 페이지(700)로부터 보안 패치를 수집할 수 있다. 예를 들어, 직접 패치 링크 기반 수집 모듈(310)은 명령 'git clone repository_url'을 사용하여 컴퓨팅 장치(100)의 로컬 환경으로 클론 저장소를 다운로드할 수 있다. 그리고, 직접 패치 링크 기반 수집 모듈(310)은 클론 저장소에서 명령 'git show com-mit_id'를 사용하여 검색된 커밋에 관련된 'diff'들을 추출할 수 있다. 다만 이에 한정되지 않고, 직접 패치 링크 기반 수집 모듈(310)은 다양한 방식으로 동작할 수 있다.

【0063】본 개시의 몇몇 실시예에 따라, 간접 패치 링크 기반 수집 모듈(320)은 간접 패치 링크(indirect patch link)에 기초하여 데이터 소스로부터 보안 패치를 수집할 수 있다.

【0064】구체적으로 설명하면, 간접 패치 링크 기반 수집 모듈(320)은 취약점 정보 페이지 상에서 식별되는 웹사이트 주소를 크롤링(crawling)할 수 있다. 몇몇 예에서, 취약점 정보 페이지 상에서 식별되는 웹사이트 주소는 'BeautifulSoup'와 같은 크롤러를 이용하여 크롤링될 수 있다. 도 5를 참조하면, 간접 패치 링크 기반 수집 모듈(320)은 'CVE-2020-14323에 대한 취약점 정보 페이지(600) 상에서 식별되는 이슈 트랙커의 웹사이트 주소 'http://bugzilla.redhat.com/show_bug.cgi?id=1891685'를 식별할 수 있다. 이 경우에, 간접 패치 링크 기반 수집 모듈(320)은 식별된 이슈 트랙커의 웹사이트 주소를 크롤러를 이용하여 크롤링할 수 있다.

【0065】크롤링을 통해, 간접 패치 링크 기반 수집 모듈(320)은 사전 결정된 패턴을 가지는 보안 패치 링크 또는 보안 패치 링크에 대한 사전 결정된 힌트 정보를 획득할 수 있다. 예를 들어, 첫번째 경우로, 간접 패치 링크 기반 수집 모듈(320)은 사전 결정된 패턴을 가지는 보안 패치 링크를 식별할 수 있다. 직접 패치 링크 기반 수집 모듈(310)의 실시예와 동일하게, 사전 결정된 패턴은 취약점 데이터소스 도메인 네임 정보 및 보안 패치 식별 문자열 정보를 포함할 수 있다. 사전 결정된 패턴을 가지는 보안 패치 링크를 획득한 경우에, 간접 패치 링크 기반 수집 모듈(320)은 보안 패치 링크를 통해 연결되는 보안 패치 페이지로부터 보안 패치를

수집할 수 있다. 간접 패치 링크 기반 수집 모듈(320)이 보안 패치 링크를 통해 보안 패치를 수집하는 동작은 상술한 직접 패치 링크 기반 수집 모듈(310)의 동작과 동일할 수 있다.

【0066】두번째 경우로, 간접 패치 링크 기반 수집 모듈(320)은 사전 결정된 힌트 정보를 식별할 수 있다. 여기서 사전 결정된 힌트 정보는 커밋 ID 정보 또는 버그 ID 정보를 포함할 수 있다. 도 6을 참조하면, 'CVE-2020-11655에 대한 취약점 정보 페이지(600) 상에서 식별되는 이슈 트랙커의 웹사이트 주소 'https://www3.sqlite.org/'cgi/src/info/4a302b42c7bf5e11'를 크롤링함으로써 간접 패치 링크 기반 수집 모듈(320)은 사전 결정된 힌트 정보로서 SHA3-256 해시값인 버그 ID(82)를 획득할 수 있다. 사전 결정된 힌트 정보를 획득한 경우, 간접 패치 링크 기반 수집 모듈(320)은 사전 결정된 힌트 정보에 대응하는 패치 커밋을 수집할 수 있다. 도 6을 참조하면, 이러한 버그 ID(82)를 가지는 관련된 커밋을 커밋 메시지에서 검색한 예가 도시된다. 그리고, 간접 패치 링크 기반 수집 모듈(320)은 검색된 커밋으로부터 'diff'들을 출력함으로써 보안 패치를 수집할 수 있다. 다만 이에 한정되지 않고, 간접 패치 링크 기반 수집 모듈(320)은 다양한 방식으로 동작할 수 있다.

【0067】본 개시의 몇몇 실시예에 따라, 비가시 패치 링크 기반 수집 모듈(330)은 비가시 패치 링크(invisible patch link)에 기초하여 데이터 소스로부터 보안 패치를 수집할 수 있다.

【0068】구체적으로 설명하면, 비가시 패치 링크 기반 수집 모듈(330)은 질답 사이트로부터 질답 포스트를 수집할 수 있다. 몇몇 예에서, 비가시 패치 링크 기반 수집 모듈(330)은 본 명세서에 그 전체가 참조로서 통합되는 ''Dicos: Discovering insecure code snippets from stack overow posts by leveraging user discussions'' 개시되는 'Dicos'에 기반하여 구현될 수 있다.

【0069】몇몇 예에서, 비가시 패치 링크 기반 수집 모듈(330)은 수집된 질답 포스트의 변경 이력을 추출할 수 있다. 예를 들어, 도 7을 참조하면, 비가시 패치 링크 기반 수집 모듈(330)은 질답 포스트의 에디트 로그(edit log)를 통해 가장 오래된 버전과 최신 버전을 비교함으로써 포스트(900)의 답변란(920)에 포함된 설명(921) 및 코드 스니펫(922)에 대한 'diff'형태의 변경 이력을 추출할 수 있다.

【0070】비가시 패치 링크 기반 수집 모듈(330)은 추출된 변경 이력으로부터 사전 결정된 특징에 대응하는 변경 정보를 식별할 수 있다. 여기서 사전 결정된 특징은 보안-감도 API(security-sensitive API), 보안-관련 키워드(security-related keyword), 및 컨트롤 플로우(control flow)에서의 변경을 포함할 수 있다. 비가시 패치 링크 기반 수집 모듈(330)은 식별된 변경 정보에 기초하여 인시큐어 코드 스니펫(insecure code snippet)을 획득할 수 있다. 예를 들어, 비가시 패치 링크 기반 수집 모듈(330)은 추출된 변경 정보가 사전 결정된 특징에 기초하여 보안 이슈를 해결하기 위한 것인지 분석함으로써 인시큐어 코드 스니펫(insecure code snippet)을 보안 패치로 획득할 수 있다.

【0071】또한, 비가시 패치 링크 기반 수집 모듈(330)은 저장소 또는 이슈 트랙커로부터 CVE ID 정보를 포함하는 커밋 메시지를 검색할 수 있다. 커밋은 상술한 변경 이력과 같은 개념으로 취급될 수 있으므로, 커밋 메시지 및 소스 코드의 diff를 분석함으로써 인시큐어 코드 스니펫 형태의 보안 패치가 추가적으로 수집될 수 있다. 예를 들어, CVE ID 정보는 관련된 커밋을 찾는 힌트를 제공할 수 있으므로, 비가시 패치 링크 기반 수집 모듈(330)은 커밋 메시지가 “CVE-20”를 포함하고 있는지 여부를 분석함으로써 패치 커밋을 검색할 수 있다(예를 들어, 명령 'git log -grep='CVE-20'이 사용될 수 있음). 비가시 패치 링크 기반 수집 모듈(330)은 검색된 커밋이 사전 결정된 특징에 기초하여 보안 이슈를 해결하기 위한 것인지 분석함으로써 검색된 커밋으로부터 보안 패치를 수집할 수 있다. 다만 이에 한정되지 않고, 비가시 패치 링크 기반 수집 모듈(330)은 다양한 방식으로 동작할 수 있다.

【0072】이하의 표 2는 각각의 링크 유형에 따라 보안 패치를 수집할 수 있는 데이터 소스(200)를 표시한다.

【0073】【표 2】

데이터 소스	링크 유형		
	직접	간접	비가시
저장소	✓	✓	✓
이슈 트랙커		✓	✓
질답 사이트			✓

【0074】표 2로부터 알 수 있는 바와 같이, Git-hub와 같은 단일한 저장소로 데이터 소스가 한정되는 기존의 방법론과 달리, 본 개시의 방법은 다양한 데이터 소스와 공용 취약점 데이터베이스 사이의 숨겨진 연결성까지 고려할 수 있는 3가지 유형의 링크를 통해 보안 패치를 수집할 수 있다.

【0075】이하의 표 3은 보안 패치 수집부(300)의 예시적인 알고리즘을 나타낸다.

【0076】 【표 3】

Algorithm 1: Algorithm for Collecting Security Patches

```
Input: V, C, R
// V: Vulnerability, C: CVE info page,
// R: Repository reporting V
Output: P
// P: Security patch for V
procedure ExtractingPatch( V, C, R)
    Ref ← References( V, C)
    for URL in Ref do
        if ("git" in URL) and ("commit" in URL) then
            // Collect P with direct patch links
            P ← Crawl( URL)
        else
            // Collect P with indirect patch links
            if GitURL in Visit( URL) then
                P ← Crawl( GitURL)
            else if H in Visit( URL) then
                // H: Hints for detecting patches (e.g., Commit ID or Bug ID)
                for Cm in R do
                    // Cm: Commit
                    P ← GetPatchCommit( Cm, H)
    // Collect P with invisible patch links
    for Cm in R do
        if "CVE-20" in Cm then
            if (IsControlFlowChanged( Cm) or IsSecurityAPIChanged( Cm)) then
                P ← Cm
    return P
```

【0077】 상술한 본 개시의 몇몇 실시예에 따른 취약점 데이터베이스를 구축하는 방법에 따라, 본 발명자는 'xVDB'라고 명명된 취약점 데이터베이스를 구축하

는 실험을 진행하였다. 본 실험에 대한 결과는 본 명세서에 그 전체가 참조로서 통합되는 'xVDB: A High-Coverage Approach for Constructing a Vulnerability Database'에 자세히 개시된다.

【0078】본 실험은 외부 라이브러리(예를 들어, BeautifulSoup)를 제외하고 대략 1,000 라인의 파이톤 코드로 구현되었다. 오픈소스 툴인 'Dicos'는 질답 사이트에서 비가시 패치 링크를 통해 보안패치를 수집하는데 사용되었다. 'Dicos'의 소스 코드는'https: //github.com/hyunji-hong/DICOS-public'에 공개되어 있다.

【0079】【표 4】

xVDB와 PatchDB 탐지 비교 결과

Approach	Direct	Indirect		Invisible			Total
	R*	R*	IT†	R*	IT†	QA‡	
PatchDB	4,076	X	X	X	X	X	4,076 CVE
xVDB	6,387	1,644	3,020	2,966	1,766	12,458	12,432 CVE 12,458 Posts

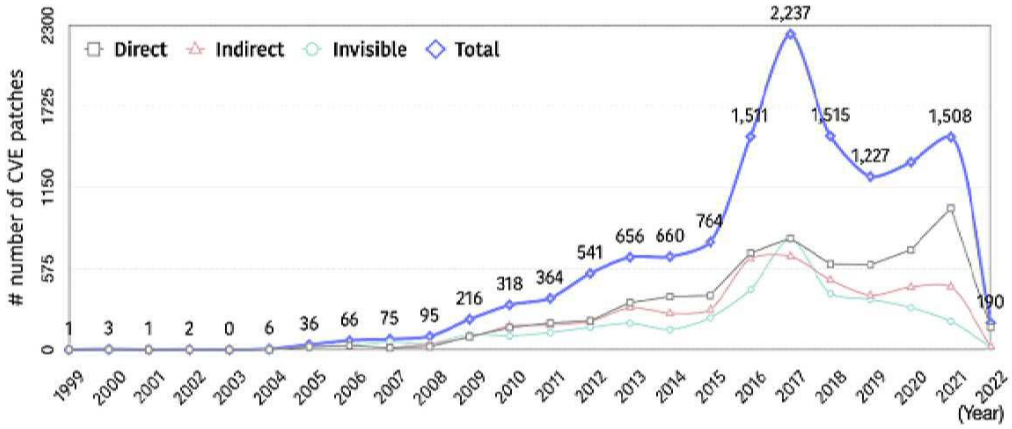
R*: Repositories
IT†: Issue trackers
QA‡: Q&A sites

【0080】표 4를 참조하면, 기존의 'PatchDB'의 경우 취약점 공개 데이터베이스인 'NVD'에서 'Git' 데이터 소스로부터 제공되는 보안 패치만이 수집되었다(C/C++ 보안패치만 고려함). 그 결과, 4,076개의 CVE 패치가 수집되었다.

【0081】본 개시에 따른 'xVDB'의 경우, 저장소, 이슈 트랙커, 질답 사이트 대상으로 'NVD'와 보안 패치의 숨겨진 연결성까지 고려하여 12,432개의 CVE 패치,

그리고 더 나아가 질답 사이트에서 12,458개의 인시큐어 포스트까지 수집되었다(CVE 패치의 경우 C/C++, Java, JavaScript, Go, Python 패치를 고려함, 인시큐어 포스트의 경우 C/C++, Android 포스트를 고려함). 이는 기존 연구와 비교했을 때, CVE 패치 기준 약 3배 정도 확장된 결과를 나타낸다.

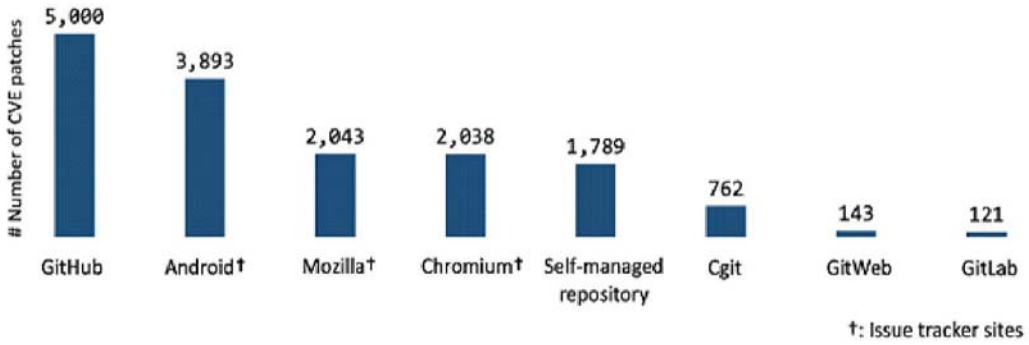
【0082】 **【표 5】**



연도별 수집된 CVE 패치 분포도

【0083】 표 5를 참조하면, 연도별로 수집된 패치 중 절반 이상이 간접 패치 링크 및 비가시 패치 링크에 기반하여 수집되었음을 알 수 있다.

【0084】【표 6】



참고 사이트 별 CVE 패치 분포도

【0085】또한, 표 6를 참조하면, 상당 수의 보안 패치가 이슈 트래커를 통해 수집됨을 알 수 있다.

【0086】도 8은 본 개시의 몇몇 실시예에 따른 취약점 데이터베이스를 구축하는 방법의 순서도이다.

【0087】본 개시의 몇몇 실시예에 따라, 취약점 데이터베이스를 구축하는 방법은 직접 패치 링크(direct patch link)에 기초하여 데이터 소스로부터 보안 패치를 수집하는 단계(s100)를 포함할 수 있다.

【0088】본 개시의 몇몇 실시예에 따라, 취약점 데이터베이스를 구축하는 방법은 간접 패치 링크(indirect patch link)에 기초하여 상기 데이터 소스로부터 보안 패치를 수집하는 단계(s200)

【0089】본 개시의 몇몇 실시예에 따라, 취약점 데이터베이스를 구축하는 방법은 비가시 패치 링크(invisible patch link)에 기초하여 상기 데이터 소스로부터 보안 패치를 수집하는 단계(s300)를 포함할 수 있다.

【0090】본 개시의 몇몇 실시예에 따라, 의료 이미지 분석 방법은 후보 약물 결정 모델에 의해, 상기 추정된 면역 아형 또는 바이오 마커 중 적어도 하나에 기초하여 후보 약물을 결정하는 단계(s300)를 포함할 수 있다.

【0091】전술한 방법의 단계들은 단지 설명을 위해 제시된 것이며, 일부 단계가 생략되거나 별도의 단계가 추가될 수 있다. 또한, 전술한 단계들은 임의의 순서에 따라 수행될 수 있다.

【0092】도 9는 본 개시의 몇몇 실시예에 따른 컴퓨팅 장치의 블록 구성도이다.

【0093】도 9는 본 개시의 몇몇 실시예들이 구현될 수 있는 예시적인 컴퓨팅 환경에 대한 간략하고 일반적인 개략도를 도시한다.

【0094】본 개시가 일반적으로 하나 이상의 컴퓨터 상에서 실행될 수 있는 컴퓨터 실행가능 명령어와 관련하여 전술되었지만, 당업자라면 본 개시가 기타 프로그램 모듈들과 결합되어 및/또는 하드웨어와 소프트웨어의 조합으로써 구현될 수 있다는 것을 잘 알 것이다.

【0095】일반적으로, 프로그램 모듈은 특정의 태스크를 수행하거나 특정의 추상 데이터 유형을 구현하는 루틴, 프로그램, 컴포넌트, 데이터 구조, 기타 등등

을 포함한다. 또한, 당업자라면 본 개시의 방법이 단일-프로세서 또는 멀티프로세서 컴퓨터 시스템, 미니컴퓨터, 메인프레임 컴퓨터는 물론 퍼스널 컴퓨터, 핸드 헬드 컴퓨팅 장치, 마이크로프로세서-기반 또는 프로그램가능 가전 제품, 기타 등등(이들 각각은 하나 이상의 연관된 장치와 연결되어 동작할 수 있음)을 비롯한 다른 컴퓨터 시스템 구성으로 실시될 수 있다는 것을 잘 알 것이다.

【0096】본 개시의 설명된 실시예들은 또한 어떤 태스크들이 통신 네트워크를 통해 연결되어 있는 원격 처리 장치들에 의해 수행되는 분산 컴퓨팅 환경에서 실시될 수 있다. 분산 컴퓨팅 환경에서, 프로그램 모듈은 로컬 및 원격 메모리 저장 장치 둘 다에 위치할 수 있다.

【0097】컴퓨터는 통상적으로 다양한 컴퓨터 판독가능 매체를 포함한다. 컴퓨터에 의해 액세스 가능한 매체는 그 어떤 것이든지 컴퓨터 판독가능 매체가 될 수 있다. 컴퓨터 판독가능 매체는 휘발성 및 비휘발성 매체, 일시적(transitory) 및 비일시적(non-transitory) 매체, 이동식 및 비-이동식 매체를 포함한다. 제한이 아닌 예로서, 컴퓨터 판독가능 매체는 컴퓨터 판독가능 저장 매체 및 컴퓨터 판독가능 전송 매체를 포함할 수 있다. 컴퓨터 판독가능 저장 매체는 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈 또는 기타 데이터와 같은 정보를 저장하는 임의의 방법 또는 기술로 구현되는 휘발성 및 비휘발성 매체, 일시적 및 비-일시적 매체, 이동식 및 비이동식 매체를 포함한다. 컴퓨터 저장 매체는 RAM, ROM, EEPROM, 플래시 메모리 또는 기타 메모리 기술, CD-ROM, DVD(digital video disk) 또는 기타 광 디스크 저장 장치, 자기 카세트, 자기 테이프, 자기 디스크 저장 장

치 또는 기타 자기 저장 장치, 또는 컴퓨터에 의해 액세스될 수 있고 원하는 정보를 저장하는 데 사용될 수 있는 임의의 기타 매체를 포함하지만, 이에 한정되지 않는다.

【0098】컴퓨터 판독가능 전송 매체는 통상적으로 기타 전송 메커니즘(transport mechanism)과 같은 피변조 데이터 신호(modulated data signal)에 컴퓨터 판독가능 명령어, 데이터 구조, 프로그램 모듈 또는 기타 데이터 등을 구현하고 모든 정보 전달 매체를 포함한다. 피변조 데이터 신호라는 용어는 신호 내에 정보를 인코딩하도록 그 신호의 특성들 중 하나 이상을 설정 또는 변경시킨 신호를 의미한다. 제한이 아닌 예로서, 컴퓨터 판독가능 전송 매체는 유선 네트워크 또는 직접 배선 접속(direct-wired connection)과 같은 유선 매체, 그리고 음향, RF, 적외선, 기타 무선 매체와 같은 무선 매체를 포함한다. 상술된 매체들 중 임의의 것의 조합도 역시 컴퓨터 판독가능 전송 매체의 범위 안에 포함되는 것으로 한다.

【0099】컴퓨터(1102)를 포함하는 본 개시의 여러가지 측면들을 구현하는 예시 적인 환경(1100)이 나타내어져 있으며, 컴퓨터(1102)는 처리 장치(1104), 시스템 메모리(1106) 및 시스템 버스(1108)를 포함한다. 시스템 버스(1108)는 시스템 메모리(1106)(이에 한정되지 않음)를 비롯한 시스템 컴포넌트들을 처리 장치(1104)에 연결시킨다. 처리 장치(1104)는 다양한 상용 프로세서들 중 임의의 프로세서일 수 있다. 듀얼 프로세서 및 기타 멀티프로세서 아키텍처도 역시 처리 장치(1104)로서 이용될 수 있다.

【0100】시스템 버스(1108)는 메모리 버스, 주변장치 버스, 및 다양한 상용 버스 아키텍처 중 임의의 것을 사용하는 로컬 버스에 추가적으로 상호 연결될 수 있는 몇 가지 유형의 버스 구조 중 임의의 것일 수 있다. 시스템 메모리(1106)는 판독 전용 메모리(ROM)(1110) 및 랜덤 액세스 메모리(RAM)(1112)를 포함한다. 기본 입/출력 시스템(BIOS)은 ROM, EPROM, EEPROM 등의 비휘발성 메모리(1110)에 저장되며, 이 BIOS는 시동 중과 같은 때에 컴퓨터(1102) 내의 구성요소들 간에 정보를 전송하는 일을 돕는 기본적인 루틴을 포함한다. RAM(1112)은 또한 데이터를 캐싱하기 위한 정적 RAM 등의 고속 RAM을 포함할 수 있다.

【0101】컴퓨터(1102)는 또한 내장형 하드 디스크 드라이브(HDD)(1114)(예를 들어, EIDE, SATA)-이 내장형 하드 디스크 드라이브(1114)는 또한 적당한 섀시(도시 생략) 내에서 외장형 용도로 구성될 수 있음-, 자기 플로피 디스크 드라이브(FDD)(1116)(예를 들어, 이동식 디스켓(1118)으로부터 판독을 하거나 그에 기록을 하기 위한 것임), 및 광 디스크 드라이브(1120)(예를 들어, CD-ROM 디스크(1122)를 판독하거나 DVD 등의 기타 고용량 광 매체로부터 판독을 하거나 그에 기록을 하기 위한 것임)를 포함한다. 하드 디스크 드라이브(1114), 자기 디스크 드라이브(1116) 및 광 디스크 드라이브(1120)는 각각 하드 디스크 드라이브 인터페이스(1124), 자기 디스크 드라이브 인터페이스(1126) 및 광 드라이브 인터페이스(1128)에 의해 시스템 버스(1108)에 연결될 수 있다. 외장형 드라이브 구현을 위한 인터페이스(1124)는 USB(Universal Serial Bus) 및 IEEE 1394 인터페이스 기술 중 적어도 하나 또는 그 둘 다를 포함한다.

【0102】이들 드라이브 및 그와 연관된 컴퓨터 판독가능 매체는 데이터, 데이터 구조, 컴퓨터 실행가능 명령어, 기타 등등의 비 휘발성 저장을 제공한다. 컴퓨터(1102)의 경우, 드라이브 및 매체는 임의의 데이터를 적당한 디지털 형식으로 저장하는 것에 대응한다. 상기에서의 컴퓨터 판독가능 매체에 대한 설명이 HDD, 이동식 자기 디스크, 및 CD 또는 DVD 등의 이동식 광 매체를 언급하고 있지만, 당업자라면 집 드라이브(zip drive), 자기 카세트, 플래쉬 메모리 카드, 카트리지, 기타 등등의 컴퓨터에 의해 판독가능한 다른 유형의 매체도 역시 예시 적인 운영 환경에서 사용될 수 있으며 또 임의의 이러한 매체가 본 개시의 방법들을 수행하기 위한 컴퓨터 실행가능 명령어를 포함할 수 있다는 것을 잘 알 것이다.

【0103】운영 체제(1130), 하나 이상의 애플리케이션 프로그램(1132), 기타 프로그램 모듈(1134) 및 프로그램 데이터(1136)를 비롯한 다수의 프로그램 모듈이 드라이브 및 RAM(1112)에 저장될 수 있다. 운영 체제, 애플리케이션, 모듈 및/또는 데이터의 전부 또는 그 일부분이 또한 RAM(1112)에 캐싱될 수 있다. 본 개시가 여러가지 상업적으로 이용가능한 운영 체제 또는 운영 체제들의 조합에서 구현될 수 있다는 것을 잘 알 것이다.

【0104】사용자는 하나 이상의 유선/무선 입력 장치, 예를 들어, 키보드(1138) 및 마우스(1140) 등의 포인팅 장치를 통해 컴퓨터(1102)에 명령 및 정보를 입력할 수 있다. 기타 입력 장치(도시 생략)로는 마이크, IR 리모콘, 조이스틱, 게임 패드, 스타일러스 펜, 터치 스크린, 기타 등등이 있을 수 있다. 이들 및 기타 입력 장치가 종종 시스템 버스(1108)에 연결되어 있는 입력 장치 인터페이스(114

2)를 통해 처리 장치(1104)에 연결되지만, 병렬 포트, IEEE 1394 직렬 포트, 게임 포트, USB 포트, IR 인터페이스, 기타 등등의 기타 인터페이스에 의해 연결될 수 있다.

【0105】모니터(1144) 또는 다른 유형의 디스플레이 장치도 역시 비디오 어댑터(1146) 등의 인터페이스를 통해 시스템 버스(1108)에 연결된다. 모니터(1144)에 부가하여, 컴퓨터는 일반적으로 스피커, 프린터, 기타 등등의 기타 주변 출력 장치(도시 생략)를 포함한다.

【0106】컴퓨터(1102)는 유선 및/또는 무선 통신을 통한 원격 컴퓨터(들)(1148) 등의 하나 이상의 원격 컴퓨터로의 논리적 연결을 사용하여 네트워크화된 환경에서 동작할 수 있다. 원격 컴퓨터(들)(1148)는 워크스테이션, 컴퓨팅 디바이스 컴퓨터, 라우터, 퍼스널 컴퓨터, 휴대용 컴퓨터, 마이크로프로세서-기반 오락 기기, 피어 장치 또는 기타 통상의 네트워크 노드일 수 있으며, 일반적으로 컴퓨터(1102)에 대해 기술된 구성요소들 중 다수 또는 그 전부를 포함하지만, 간략함을 위해, 메모리 저장 장치(1150)만이 도시되어 있다. 도시되어 있는 논리적 연결은 근거리 통신망(LAN)(1152) 및/또는 더 큰 네트워크, 예를 들어, 원거리 통신망(WAN)(1154)에의 유선/무선 연결을 포함한다. 이러한 LAN 및 WAN 네트워킹 환경은 사무실 및 회사에서 일반적인 것이며, 인트라넷 등의 전사적 컴퓨터 네트워크(enterprise-wide computer network)를 용이하게 해주며, 이들 모두는 전세계 컴퓨터 네트워크, 예를 들어, 인터넷에 연결될 수 있다.

【0107】LAN 네트워킹 환경에서 사용될 때, 컴퓨터(1102)는 유선 및/또는 무선 통신 네트워크 인터페이스 또는 어댑터(1156)를 통해 로컬 네트워크(1152)에 연결된다. 어댑터(1156)는 LAN(1152)에의 유선 또는 무선 통신을 용이하게 해줄 수 있으며, 이 LAN(1152)은 또한 무선 어댑터(1156)와 통신하기 위해 그에 설치되어 있는 무선 액세스 포인트를 포함하고 있다. WAN 네트워킹 환경에서 사용될 때, 컴퓨터(1102)는 모뎀(1158)을 포함할 수 있거나, WAN(1154) 상의 통신 컴퓨팅 디바이스에 연결되거나, 또는 인터넷을 통하는 등, WAN(1154)을 통해 통신을 설정하는 기타 수단을 갖는다. 내장형 또는 외장형 및 유선 또는 무선 장치일 수 있는 모뎀(1158)은 직렬 포트 인터페이스(1142)를 통해 시스템 버스(1108)에 연결된다. 네트워크화 된 환경에서, 컴퓨터(1102)에 대해 설명된 프로그램 모듈들 또는 그의 일부분이 원격 메모리/저장 장치(1150)에 저장될 수 있다. 도시된 네트워크 연결이 예시 적인 것이며 컴퓨터들 사이에 통신 링크를 설정하는 기타 수단이 사용될 수 있다는 것을 잘 알 것이다.

【0108】컴퓨터(1102)는 무선 통신으로 배치되어 동작하는 임의의 무선 장치 또는 개체, 예를 들어, 프린터, 스캐너, 데스크톱 및/또는 휴대용 컴퓨터, PDA(portable data assistant), 통신 위성, 무선 검출가능 태그와 연관된 임의의 장비 또는 장소, 및 전화와 통신을 하는 동작을 한다. 이것은 적어도 Wi-Fi 및 블루투스 무선 기술을 포함한다. 따라서, 통신은 종래의 네트워크에서와 같이 미리 정의된 구조이거나 단순하게 적어도 2개의 장치 사이의 애드혹 통신(ad hoc communication)일 수 있다.

【0109】Wi-Fi(Wireless Fidelity)는 유선 없이도 인터넷 등으로의 연결을 가능하게 해준다. Wi-Fi는 이러한 장치, 예를 들어, 컴퓨터가 실내에서 및 실외에서, 즉 기지국의 통화권 내의 아무 곳에서나 데이터를 전송 및 수신할 수 있게 해주는 셀 전화와 같은 무선 기술이다. Wi-Fi 네트워크는 안전하고 신뢰성 있으며 고속인 무선 연결을 제공하기 위해 IEEE 802.11(a,b,g, 기타)이라고 하는 무선 기술을 사용한다. 컴퓨터를 서로에, 인터넷에 및 유선 네트워크(IEEE 802.3 또는 이더넷을 사용함)에 연결시키기 위해 Wi-Fi가 사용될 수 있다. Wi-Fi 네트워크는 비인가 2.4 및 5 GHz 무선 대역에서, 예를 들어, 11Mbps(802.11a) 또는 54 Mbps(802.11b) 데이터 레이트로 동작하거나, 양 대역(듀얼 대역)을 포함하는 제품에서 동작할 수 있다.

【0110】본 개시의 기술 분야에서 통상의 지식을 가진 자는 정보 및 신호들이 임의의 다양한 상이한 기술들 및 기법들을 이용하여 표현될 수 있다는 것을 이해할 것이다. 예를 들어, 위의 설명에서 참조될 수 있는 데이터, 지시들, 명령들, 정보, 신호들, 비트들, 심볼들 및 칩들은 전압들, 전류들, 전자기파들, 자기장들 또는 입자들, 광학장들 또는 입자들, 또는 이들의 임의의 결합에 의해 표현될 수 있다.

【0111】본 개시의 기술 분야에서 통상의 지식을 가진 자는 여기에 개시된 실시예들과 관련하여 설명된 다양한 예시 적인 논리 블록들, 모듈들, 프로세서들, 수단들, 회로들 및 알고리즘 단계들이 전자 하드웨어, (편의를 위해, 여기에서 "소프트웨어"로 지칭되는) 다양한 형태들의 프로그램 또는 설계 코드 또는 이들 모두

의 결합에 의해 구현될 수 있다는 것을 이해할 것이다. 하드웨어 및 소프트웨어의 이러한 상호 호환성을 명확하게 설명하기 위해, 다양한 예시 적인 컴포넌트들, 블록들, 모듈들, 회로들 및 단계들이 이들의 기능과 관련하여 위에서 일반적으로 설명되었다. 이러한 기능이 하드웨어 또는 소프트웨어로서 구현되는지 여부는 특정한 애플리케이션 및 전체 시스템에 대하여 부과되는 설계 제약들에 따라 좌우된다. 본 개시의 기술 분야에서 통상의 지식을 가진 자는 각각의 특정한 애플리케이션에 대하여 다양한 방식들로 설명된 기능을 구현할 수 있으나, 이러한 구현 결정들은 본 개시의 범위를 벗어나는 것으로 해석되어서는 안 될 것이다.

【0112】 여기서 제시된 다양한 실시예들은 방법, 장치, 또는 표준 프로그래밍 및/또는 엔지니어링 기술을 사용한 제조 물품(article)으로 구현될 수 있다. 용어 "제조 물품"은 임의의 컴퓨터-판독가능 장치로부터 액세스 가능한 컴퓨터 프로그램 또는 매체(media)를 포함한다. 예를 들어, 컴퓨터-판독가능 매체는 자기 저장 장치(예를 들면, 하드 디스크, 플로피 디스크, 자기 스트립, 등), 광학 디스크(예를 들면, CD, DVD, 등), 스마트 카드, 및 플래쉬 메모리 장치(예를 들면, EEPROM, 카드, 스틱, 키 드라이브, 등)를 포함하지만, 이들로 제한되는 것은 아니다. 또한, 여기서 제시되는 다양한 저장 매체는 정보를 저장하기 위한 하나 이상의 장치 및/또는 다른 기계-판독가능한 매체를 포함한다.

【0113】 제시된 프로세스들에 있는 단계들의 특정한 순서 또는 계층 구조는 예시 적인 접근들의 일례임을 이해하도록 한다. 설계 우선순위들에 기반하여, 본 개시의 범위 내에서 프로세스들에 있는 단계들의 특정한 순서 또는 계층 구조가 재

배열될 수 있다는 것을 이해하도록 한다. 첨부된 방법 청구항들은 샘플 순서로 다양한 단계들의 엘리먼트들을 제공하지만 제시된 특정한 순서 또는 계층 구조에 한정되는 것을 의미하지는 않는다.

【0114】 제시된 실시예들에 대한 설명은 임의의 본 개시의 기술 분야에서 통상의 지식을 가진 자가 본 개시를 이용하거나 또는 실시할 수 있도록 제공된다. 이러한 실시예들에 대한 다양한 변형들은 본 개시의 기술 분야에서 통상의 지식을 가진 자에게 명백할 것이다. 여기에 정의된 일반적인 원리들은 본 개시의 범위를 벗어남이 없이 다른 실시예들에 적용될 수 있다. 그리하여, 본 개시는 여기에 제시된 실시예들로 한정되는 것이 아니라, 여기에 제시된 원리들 및 신규한 특징들과 일관되는 최광의의 범위에서 해석되어야 할 것이다.

【청구범위】

【청구항 1】

직접 패치 링크(direct patch link)에 기초하여 데이터 소스로부터 보안 패치를 수집하는 단계;

간접 패치 링크(indirect patch link)에 기초하여 상기 데이터 소스로부터 보안 패치를 수집하는 단계; 및

비가시 패치 링크(invisible patch link)에 기초하여 상기 데이터 소스로부터 보안 패치를 수집하는 단계;

를 포함하는,

컴퓨팅 장치에 의해 수행되는 취약점 데이터베이스를 구축하는 방법.

【청구항 2】

제 1 항에 있어서,

상기 직접 패치 링크에 기초하여 데이터 소스로부터 보안 패치를 수집하는 단계는:

취약점 정보 페이지 상에서 사전 결정된 패턴을 가지는 보안 패치 링크를 식별하는 단계; 및

상기 보안 패치 링크를 통해 연결되는 보안 패치 페이지로부터 보안 패치를 수집하는 단계;

를 포함하는,

컴퓨팅 장치에 의해 수행되는 취약점 데이터베이스를 구축하는 방법.

【청구항 3】

제 2 항에 있어서,

상기 사전 결정된 패턴은 취약점 데이터 소스 도메인 네임 정보 및 보안 패치 식별 문자열 정보를 포함하는,

컴퓨팅 장치에 의해 수행되는 취약점 데이터베이스를 구축하는 방법.

【청구항 4】

제 1 항에 있어서,

상기 간접 패치 링크(indirect patch link)에 기초하여 상기 데이터 소스로부터 보안 패치를 수집하는 단계는:

취약점 정보 페이지 상에서 식별되는 웹사이트 주소를 크롤링(crawling)하는 단계;

사전 결정된 패턴을 가지는 보안 패치 링크 또는 사전 결정된 힌트 정보를 획득하는 단계; 및

상기 보안 패치 링크 또는 상기 사전 결정된 힌트 정보에 기초하여 보안 패치를 수집하는 단계;

를 포함하는,

컴퓨팅 장치에 의해 수행되는 취약점 데이터베이스를 구축하는 방법.

【청구항 5】

제 4 항에 있어서,

상기 보안 패치 링크 또는 상기 사전 결정된 힌트 정보에 기초하여 보안 패치를 수집하는 단계는:

상기 보안 패치 링크를 통해 연결되는 보안 패치 페이지로부터 보안 패치를 수집하는 단계;

를 포함하는,

컴퓨팅 장치에 의해 수행되는 취약점 데이터베이스를 구축하는 방법.

【청구항 6】

제 5 항에 있어서,

상기 사전 결정된 패턴은 취약점 데이터 소스 도메인 네임 정보 및 보안 패치 식별 문자열 정보를 포함하는,

컴퓨팅 장치에 의해 수행되는 취약점 데이터베이스를 구축하는 방법.

【청구항 7】

제 4 항에 있어서,

상기 보안 패치 링크 또는 상기 사전 결정된 힌트 정보에 기초하여 보안 패치를 수집하는 단계는:

상기 사전 결정된 힌트 정보에 대응하는 패치 커밋을 수집하는 단계;

를 포함하는,

컴퓨팅 장치에 의해 수행되는 취약점 데이터베이스를 구축하는 방법.

【청구항 8】

제 7 항에 있어서,

상기 사전 결정된 힌트 정보는 커밋 ID 정보 또는 버그 ID 정보를 포함하는,

컴퓨팅 장치에 의해 수행되는 취약점 데이터베이스를 구축하는 방법.

【청구항 9】

제 1 항에 있어서,

상기 비가시 패치 링크에 기초하여 상기 데이터 소스로부터 보안 패치를 수집하는 단계는:

질답 사이트로부터 질답 포스트를 수집하는 단계;

상기 수집된 질답 포스트의 변경 이력을 추출하는 단계;

상기 추출된 변경 이력으로부터 사전 결정된 특징에 대응하는 변경 정보를 식별하는 단계;

상기 식별된 변경 정보에 기초하여 인시큐어 코드 스니펫(insecure code snippet)을 획득하는 단계;

를 포함하는,

컴퓨팅 장치에 의해 수행되는 취약점 데이터베이스를 구축하는 방법.

【청구항 10】

제 9 항에 있어서,

상기 사전 결정된 특징은 보안-민감 API(security-sensitive API), 보안-관련 키워드(security-related keyword), 및 컨트롤 플로우(control flow)에서의 변경을 포함하는,

컴퓨팅 장치에 의해 수행되는 취약점 데이터베이스를 구축하는 방법.

【청구항 11】

제 1 항에 있어서,

상기 비가시 패치 링크에 기초하여 상기 데이터 소스로부터 보안 패치를 수집하는 단계는:

저장소 또는 이슈 트랙커에서 CVE ID 정보를 포함하는 커밋 메시지를 검색하는 단계;

사전 결정된 특징에 기초하여 상기 검색된 커밋 메시지를 분석함으로써 상기 검색된 커밋 메시지로부터 보안 패치를 수집하는 단계;

를 포함하는,

컴퓨팅 장치에 의해 수행되는 취약점 데이터베이스를 구축하는 방법.

【청구항 12】

컴퓨터 판독가능 매체에 저장된 컴퓨터 프로그램으로서, 상기 컴퓨터 프로그램은 하나 이상의 프로세서로 하여금 취약점 데이터베이스를 구축하는 방법을 수행하게 하기 위한 명령들을 포함하며, 상기 방법은:

직접 패치 링크(direct patch link)에 기초하여 데이터 소스로부터 보안 패치를 수집하는 단계;

간접 패치 링크(indirect patch link)에 기초하여 상기 데이터 소스로부터 보안 패치를 수집하는 단계; 및

비가시 패치 링크(invisible patch link)에 기초하여 상기 데이터 소스로부터 보안 패치를 수집하는 단계;

컴퓨터 판독가능 매체에 저장된 컴퓨터 프로그램.

【청구항 13】

취약점 데이터베이스를 구축하는 방법을 수행하는 컴퓨팅 장치로서, 상기 컴

퓨팅 장치는:

컴퓨터 실행가능한 컴포넌트들을 포함하는 메모리;

메모리에 저장된 이하의 컴퓨터 실행가능한 컴포넌트들을 실행하는 프로세서; 를 포함하고,

상기 프로세서는,

직접 패치 링크(direct patch link)에 기초하여 데이터 소스로부터 보안 패치를 수집하고,

간접 패치 링크(indirect patch link)에 기초하여 상기 데이터 소스로부터 보안 패치를 수집하고,

비가시 패치 링크(invisible patch link)에 기초하여 상기 데이터 소스로부터 보안 패치를 수집하는,

컴퓨팅 장치.

【요약서】

【요약】

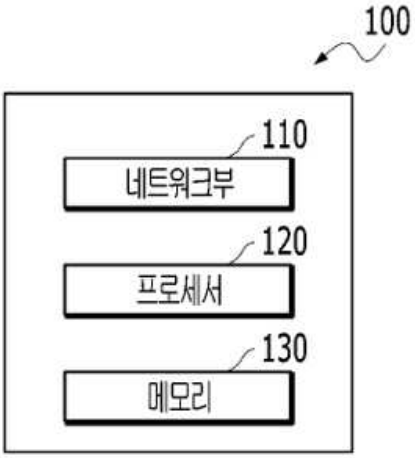
본 개시의 몇몇 실시예에 따라, 컴퓨팅 장치에 의해 수행되는 취약점 데이터베이스를 구축하는 방법이 개시된다. 상기 방법은: 직접 패치 링크(direct patch link)에 기초하여 데이터 소스로부터 보안 패치를 수집하는 단계; 간접 패치 링크(indirect patch link)에 기초하여 상기 데이터 소스로부터 보안 패치를 수집하는 단계; 및 비가시 패치 링크(invisible patch link)에 기초하여 상기 데이터 소스로부터 보안 패치를 수집하는 단계; 를 포함할 수 있다.

【대표도】

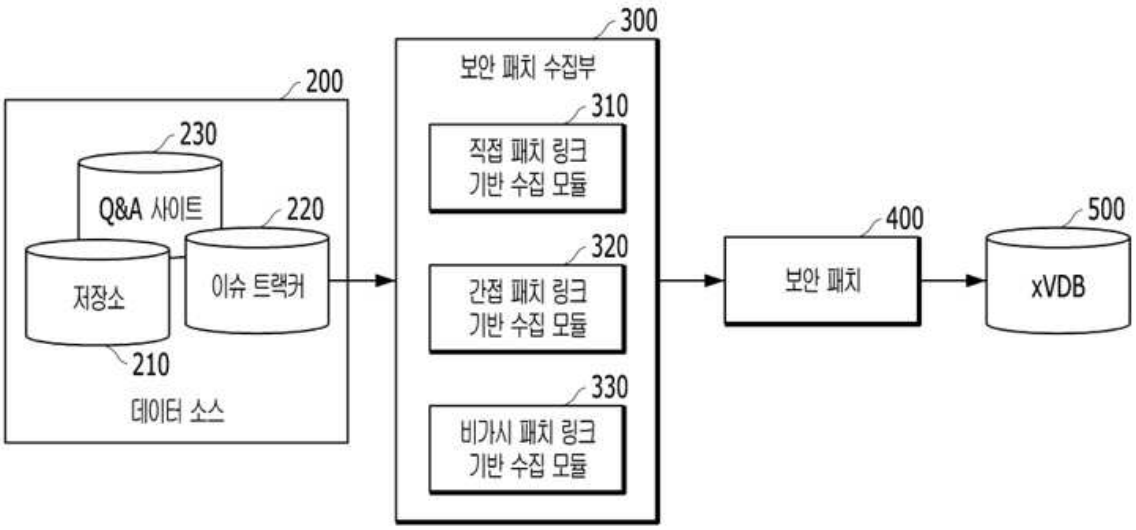
도 1

【도면】

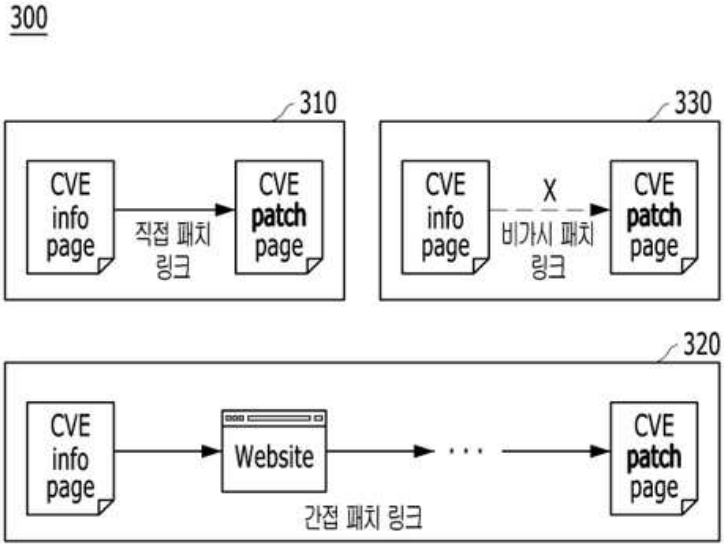
【도 1】


100
110
네트워크부
120
프로세서
130
메모리

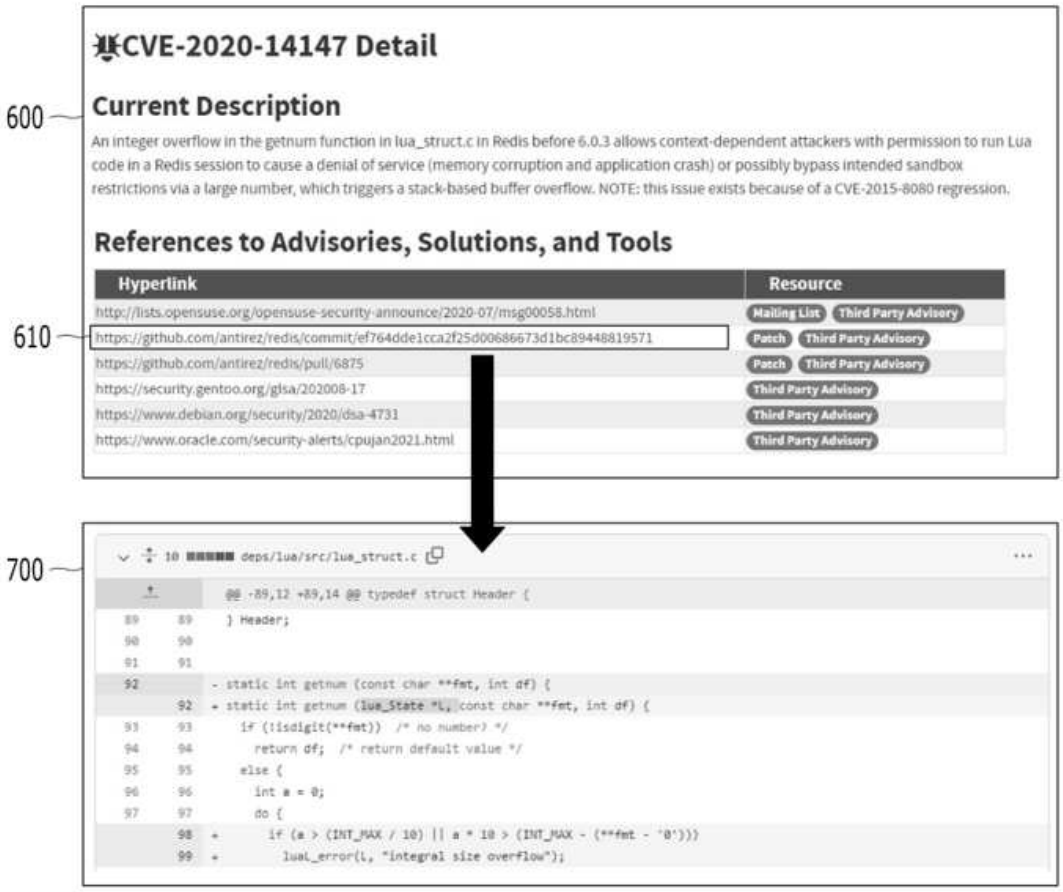
【도 2】


300
보안 패치 수집부
310
직접 패치 링크 기반 수집 모듈
320
간접 패치 링크 기반 수집 모듈
330
비가시 패치 링크 기반 수집 모듈
200
230
Q&A 사이트
220
저장소
이슈 트랙커
210
데이터 소스
400
보안 패치
500
xVDB

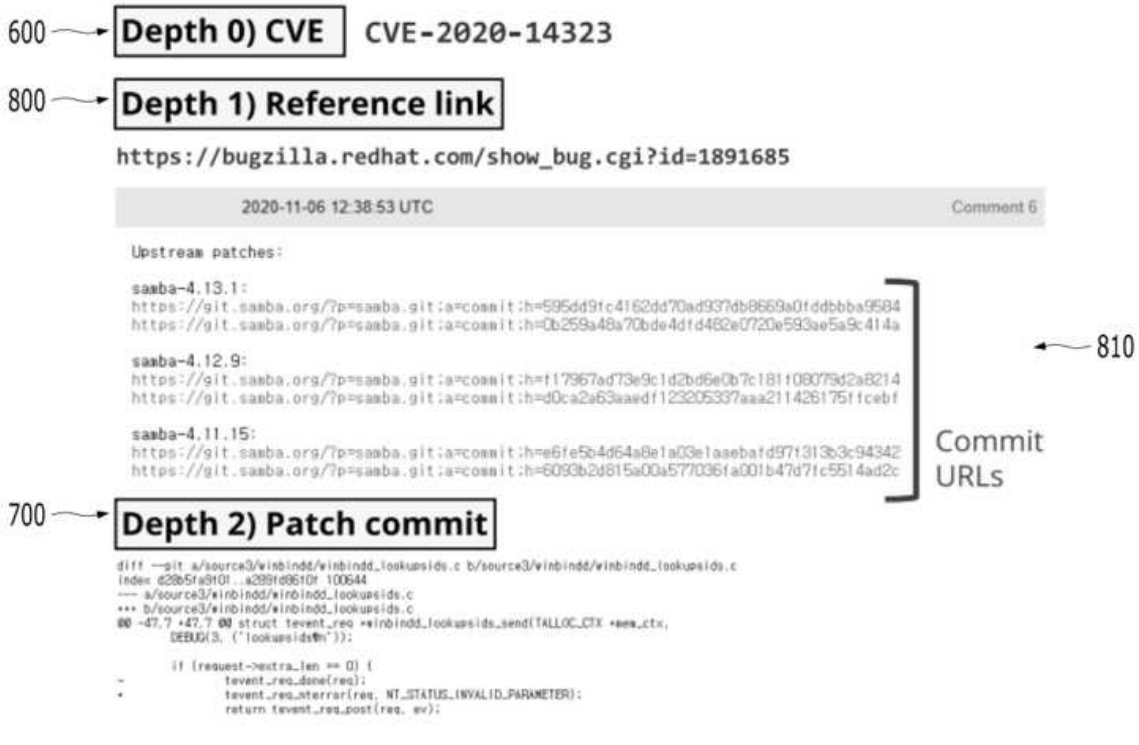
【도 3】


300
310
CVE info page
직접 패치 링크
CVE patch page
330
CVE info page
X
비가시 패치 링크
CVE patch page
320
CVE info page
Website
CVE patch page
간접 패치 링크

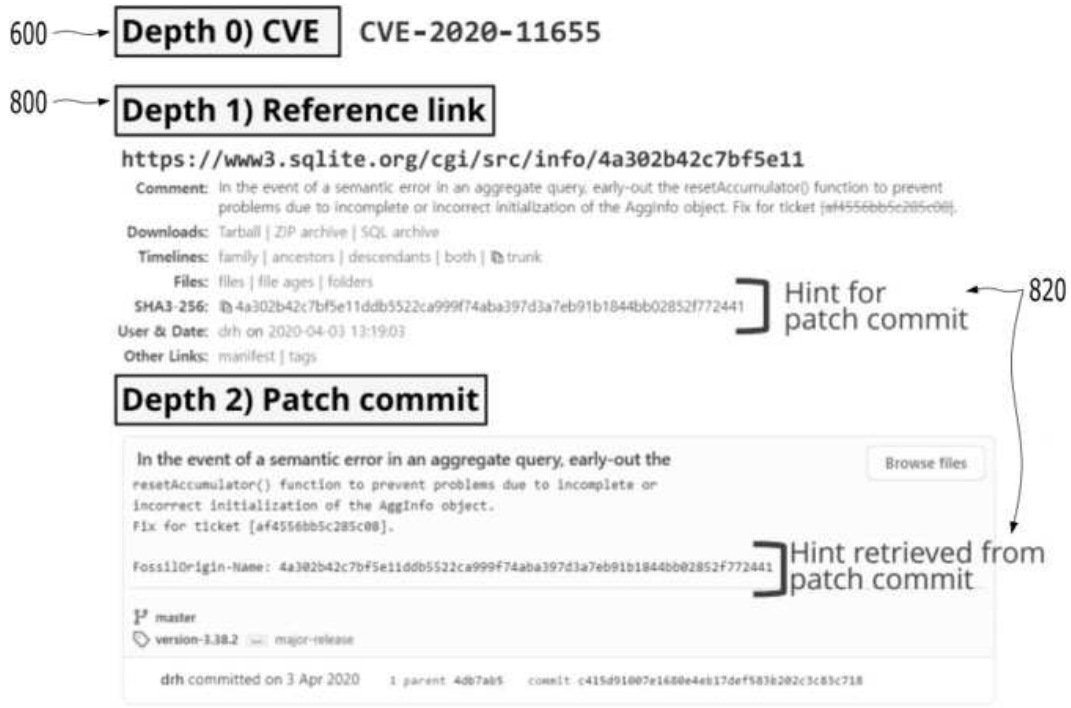
【도 4】


CVE-2020-14147 Detail
600
Current Description
An integer overflow in the getnum function in lua_struct.c in Redis before 6.0.3 allows context-dependent attackers with permission to run Lua code in a Redis session to cause a denial of service (memory corruption and application crash) or possibly bypass intended sandbox restrictions via a large number, which triggers a stack-based buffer overflow. NOTE: this issue exists because of a CVE-2015-8080 regression.
References to Advisories, Solutions, and Tools
Hyperlink
Resource
610
http://lists.opensuse.org/opensuse-security-announce/2020-07/msg00058.html
Mailing List
Third Party Advisory
https://github.com/antirez/redis/commit/ef764dde1cca2f25d00686673d1bc89448819571
Patch
Third Party Advisory
https://github.com/antirez/redis/pull/6875
Patch
Third Party Advisory
https://security.gentoo.org/glsa/202008-17
Third Party Advisory
https://www.debian.org/security/2020/dsa-4731
Third Party Advisory
https://www.oracle.com/security-alerts/cpujan2021.html
Third Party Advisory
700
10 deps/lua/src/lua_struct.c
@@ -89,12 +89,14 @@ typedef struct Header {
} Header;
- static int getnum (const char **fmt, int df) {
+ static int getnum (lua_State *L, const char **fmt, int df) {
if (!isdigit(**fmt)) /* no number? */
return df; /* return default value */
else {
int a = 0;
do {
+ if (a > (INT_MAX / 10) || a * 10 > (INT_MAX - (**fmt - '0')))
+ luaL_error(L, "integral size overflow");

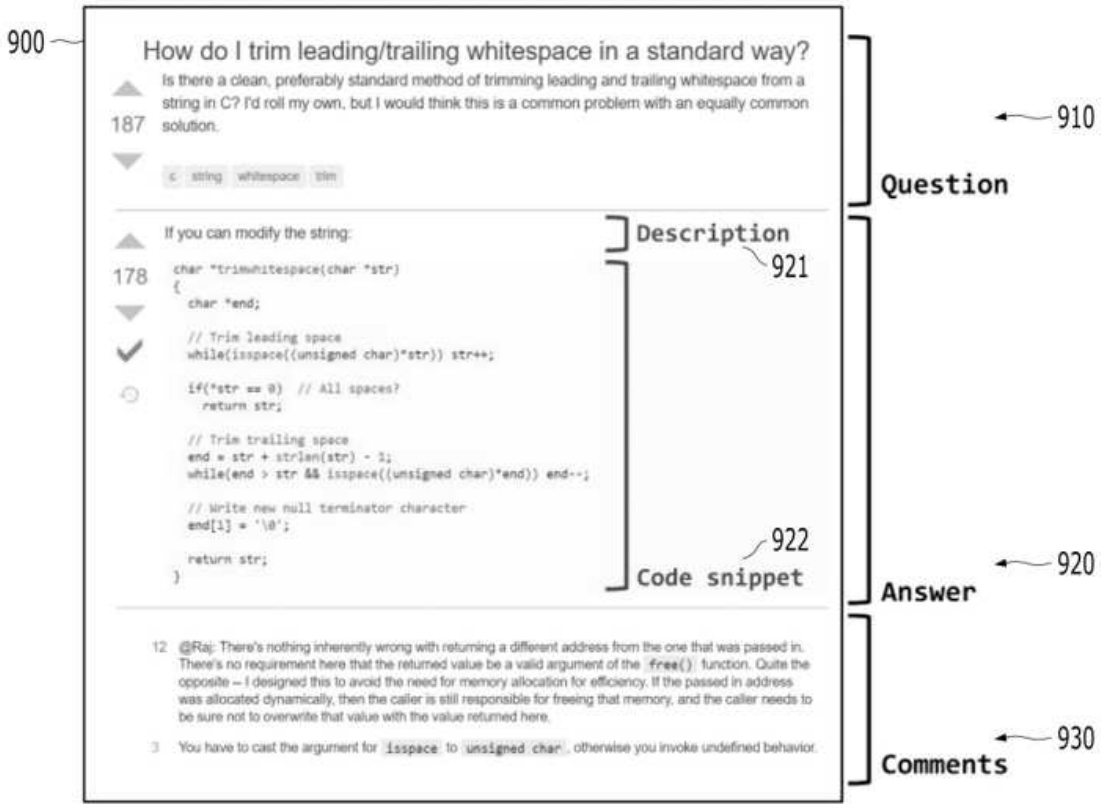
【도 5】


600
Depth 0) CVE
CVE-2020-14323
800
Depth 1) Reference link
https://bugzilla.redhat.com/show_bug.cgi?id=1891685
2020-11-06 12:38:53 UTC
Comment 6
Upstream patches:
samba-4.13.1:
https://git.samba.org/?p=samba.git;a=commit;h=595dd9fc4162dd70ad937db8669a0fddbbba9584
https://git.samba.org/?p=samba.git;a=commit;h=0b259a48a70bde4dfd482e0720e593ae5a9c414a
samba-4.12.9:
https://git.samba.org/?p=samba.git;a=commit;h=f17967ad73e9c1d2bd6e0b7c181f08079d2a8214
https://git.samba.org/?p=samba.git;a=commit;h=d0ca2a63aaedf123205337aaa211426175ffcebf
samba-4.11.15:
https://git.samba.org/?p=samba.git;a=commit;h=e6fe5b4d64a8e1a03e1aaebafd97f313b3c94342
https://git.samba.org/?p=samba.git;a=commit;h=6093b2d815a00a577036fa001b47d7fc5514ad2c
810
Commit URLs
700
Depth 2) Patch commit
diff --git a/source3/winbindd/winbindd_lookupsids.c b/source3/winbindd/winbindd_lookupsids.c
index d28b5fa9101..a2891d9610f 100644
--- a/source3/winbindd/winbindd_lookupsids.c
+++ b/source3/winbindd/winbindd_lookupsids.c
@@ -47,7 +47,7 @@ struct tevent_req *winbindd_lookupsids_send(TALLOC_CTX *mem_ctx,
DEBUG(3, ("lookupsids\n"));
if (request->extra_len == 0) {
-		tevent_req_done(req);
+		tevent_req_nterror(req, NT_STATUS_INVALID_PARAMETER);
return tevent_req_post(req, ev);

【도 6】


600
Depth 0) CVE
CVE-2020-11655
800
Depth 1) Reference link
https://www3.sqlite.org/cgi/src/info/4a302b42c7bf5e11
Comment: In the event of a semantic error in an aggregate query, early-out the resetAccumulator() function to prevent problems due to incomplete or incorrect initialization of the AggInfo object. Fix for ticket [af4556bb5c285c08].
Downloads: Tarball | ZIP archive | SQL archive
Timelines: family | ancestors | descendants | both | trunk
Files: files | file ages | folders
SHA3-256: 4a302b42c7bf5e11ddb5522ca999f74aba397d3a7eb91b1844bb02852f772441
User & Date: drh on 2020-04-03 13:19:03
Other Links: manifest | tags
Hint for patch commit
820
Depth 2) Patch commit
In the event of a semantic error in an aggregate query, early-out the
Browse files
resetAccumulator() function to prevent problems due to incomplete or
incorrect initialization of the AggInfo object.
Fix for ticket [af4556bb5c285c08].
FossilOrigin-Name: 4a302b42c7bf5e11ddb5522ca999f74aba397d3a7eb91b1844bb02852f772441
Hint retrieved from patch commit
master
version-3.38.2 major-release
drh committed on 3 Apr 2020 1 parent 4db7ab5 commit c415d91007e1680e4eb17def583b202c3c85c718

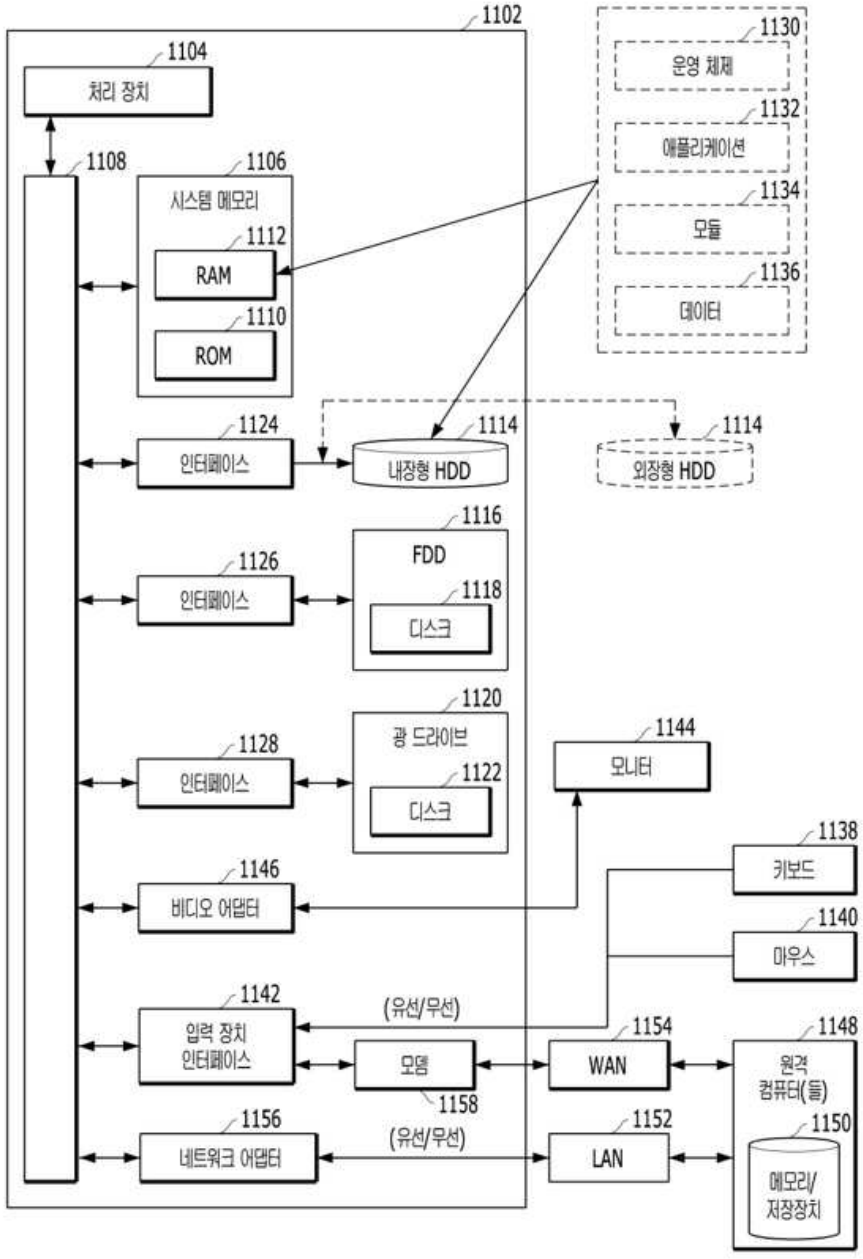
【도 7】


900
How do I trim leading/trailing whitespace in a standard way?
Is there a clean, preferably standard method of trimming leading and trailing whitespace from a string in C? I'd roll my own, but I would think this is a common problem with an equally common solution.
187
c string whitespace trim
910
Question
If you can modify the string:
178
Description
921
char *trimwhitespace(char *str)
{
char *end;
// Trim leading space
while(isspace((unsigned char)*str)) str++;
if(*str == 0) // All spaces?
return str;
// Trim trailing space
end = str + strlen(str) - 1;
while(end > str && isspace((unsigned char)*end)) end--;
// Write new null terminator character
end[1] = '\0';
return str;
}
922
Code snippet
920
Answer
12 @Raj: There's nothing inherently wrong with returning a different address from the one that was passed in. There's no requirement here that the returned value be a valid argument of the free() function. Quite the opposite -- I designed this to avoid the need for memory allocation for efficiency. If the passed in address was allocated dynamically, then the caller is still responsible for freeing that memory, and the caller needs to be sure not to overwrite that value with the value returned here.
3 You have to cast the argument for isspace to unsigned char, otherwise you invoke undefined behavior.
930
Comments

【도 8】


직접 패치 링크(direct patch link)에 기초하여 데이터 소스로부터 보안 패치를 수집하는 단계
S100
간접 패치 링크(indirect patch link)에 기초하여 데이터 소스로부터 보안 패치를 수집
S200
비가시 패치 링크(invisible patch link)에 기초하여 데이터 소스로부터 보안 패치를 수집
S300

【도 9】


1102
1104
처리 장치
1108
1106
시스템 메모리
1112
RAM
1110
ROM
1130
운영 체제
1132
애플리케이션
1134
모듈
1136
데이터
1124
인터페이스
1114
내장형 HDD
1114
외장형 HDD
1116
FDD
1126
인터페이스
1118
디스크
1120
광 드라이브
1128
인터페이스
1122
디스크
1144
모니터
1138
키보드
1146
비디오 어댑터
1140
마우스
1142
입력 장치
인터페이스
(유선/무선)
1154
WAN
1148
원격
컴퓨터(들)
모뎀
1158
1156
네트워크 어댑터
(유선/무선)
1152
LAN
1150
메모리/
저장장치