취약점 탐지 및 해결을 위한 구조화된 소프트웨어 취약점 데이터베이스 정보 표현 규격

우승훈 고려대학교 소프트웨어보안연구소 교수
홍현지 고려대학교 컴퓨터학과 석사
이희조 고려대학교 컴퓨터학과 교수

1. 머리말

소프트웨어 개발 과정에서, 타사 소프트웨어의 소스코드를 재사용하는 개발 방식은, 개발 시간 단축 및 비용 감소 등의 효율성으로 인해 널리 활용되고 있다. 이렇듯 하나의 소프트웨어에서 개발된 소스코드가 다른 소프트웨어로 전파될 수 있는 상황에서, 재사용되는 소스코드에 취약점이 포함되어 있는 경우 이는 곧 취약점의 전파를 야기한다. 재사용되는 취약한 코드 조각은 전체 소프트웨어의 보안성을 위협할 수 있으며, 궁극적으로 소프트웨어 생태계에 다양한 보안 위협을 초래한다.

알려진 취약점의 전파로 인해 발생하는 보안 위협을 예방 및 완화하기 위해서는 소프트웨어 취약점 존재 여부를 판단하기 위한 정보와 취약점을 패치하기 위한 정보가 필요하다. 이를 활용하여 취약점 존재 여부를 조기에 탐지하고, 적절한 보안 패치를 적용함으로써 피해를 예방할 수 있다, 하지만 현재의 취약점 데이터베이스가 제공하는 취약점 정보들은 취약점 존재 여부를 판단하거나, 탐지된 취약점을 해결하는 데 활용하기 어렵다. 일례로 STIX™ Version 2.1. Part 4 같은 관련 표준은 취약점에 관한 기본적인 메타데이터만 제공할 뿐, 취약점 탐지 및 해결에 필요한 정보들을 거의 제공하지 않는다. 또한 CVE MITRE나 NVD 등 현재의 취약점 데이터베이스들은 취약점 탐지 및 해결과 관련된 정보를 대부분 선택적으로 제공할 뿐, 필수적으로 제공하지는 않는다.

본고에서는 취약점의 메타데이터와 더불어 실제 취약점 탐지 및 해결에 필요한 정보들을 구조화하여 제공하는 방법을 소개한다.

2. 소프트웨어 취약점

2.1 소프트웨어 취약점 현황

알려진 취약점의 전파로 인해 발생하는 보안 위협을 완화하기 위해서, 소프트웨어에서 발견된 취약점들은 관련 정보들과 함께 공개적으로 관

<표 1> NVD 취약점 정보 표현 규격

식별자	항목	설명
T1	취약점 식별자	취약점의 식별자를 나타내는 값
T2	상세 취약점 정보	취약점에 대한 전반적인 소개 및 관련 내용을 서술형으로 설명한 것
T3	취약점 위험도	취약점의 위험도를 나타내는 지표. CVSS를 활용하여 취약점 위험도를 나타냄
T4	취약점 유형	취약점의 유형(예: 버퍼 오버플로우)을 나타냄. CWE를 활용하여 취약점의 타입을 분류함
T5	취약 소프트웨어 이름	취약점에 영향을 받는 소프트웨어 이름(CPE)
T6	취약 소프트웨어 버전	취약점에 영향을 받는 소프트웨어 버전 정보
T7	참고자료	취약점과 관련 있는 웹사이트 등 참고자료
T8	취약점 정보 수정 내역	해당 취약점 정보 문서를 수정한 내역

리된다. 예를 들어, 미국의 MITRE Corporation은 보고된 소프트웨어 취약점들 각각에 고유한 번호(CVE)를 부여하여 관련 정보들과 함께 취약점들을 관리한다. [그림 1]에 나와 있듯, 2023년 7월 기준으로 MITRE에 등록된 전체 CVE의 수는 20만 개가 넘으며, 매년 CVE로 등록되는 보안 취약점의 수는 꾸준히 증가하고 있다.

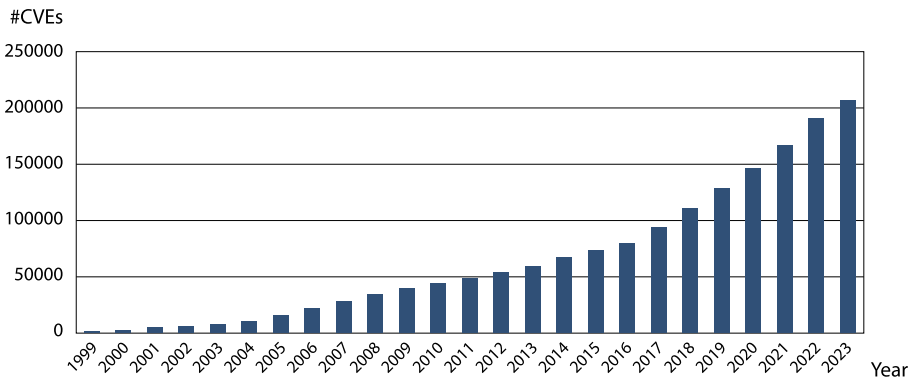
2.2 소프트웨어 취약점 데이터베이스

소프트웨어 취약점 정보들을 데이터베이스화하여 관리한 후, 이를 유저들에게 공개하는 CVE MITRE나 NVD 등의 플랫폼을 소프트웨어 취약점 데이터베이스라 칭한다. <표 1>은 취약점 데이터베이스 중 하나인 NVD의 취약점 정보 표현 규격을 나타낸다.

소프트웨어 취약점 데이터베이스의 목적은 알려진 취약점들에 관한 정보들을 관리 및 공개함으로써 (1)취약점의 원인을 명시하고 (2)해결책을 제시하며 (3)알려진 취약점으로부터 재발할 수 있는 위협을 방지할 수 있도록 취약점에 관한 정보를 제공하는 것이다.

3. 소프트웨어 취약점 데이터베이스의 정보 표현 규격 표준

3.1 표준 개발 목적



[그림 1] 연도별 누적 CVE 취약점 개수

<표 2> 소프트웨어 취약점 데이터베이스 정보 표현 규격

번호	항목	설명	필수 여부
T1	취약점 식별자	취약점의 식별자를 나타내는 값(CVE ID와 같은 취약점의 고유한 ID 값)	필수
T2	상세 취약점 정보	취약점 관련 내용을 서술형으로 설명한 것	필수
T3	취약점 위험도	취약점의 위험도를 나타내는 지표(CVSS)	필수
T4	취약점 유형	취약점의 유형을 나타냄(CWE)	필수
T5	취약 소프트웨어 이름	취약점에 영향받는 소프트웨어 이름	필수
T6	취약 소프트웨어 버전	취약점에 영향받는 소프트웨어 버전정보	필수
T7	참고 자료	취약점과 관련 있는 웹사이트 등 참고 자료	필수
T8	취약점 정보 수정 내역	해당 취약점 정보 문서를 수정한 내역	필수
N1	원점 소프트웨어 이름	취약점이 최초 등장한 원점 소프트웨어 이름	필수
N2	원점 소프트웨어 버전	취약점이 최초 등장한 원점 소프트웨어 버전정보	필수
N3	취약 함수/파일 소스코드	취약한 함수 및 파일의 소스코드(GitHub 링크 등 활용)	선택
N4	취약 함수/파일 해시값	취약한 함수 및 파일의 해시값(해시 생성 툴 공개 예정)	필수
N5	취약점 재현 방법	취약점 재현에 관한 정보(PoC 파일 혹은 재현 과정 설명)	선택
N6	취약점 위치정보	원점 소프트웨어 내의 취약점 위치정보(취약 코드 경로)	선택
N7	취약점 패치	취약점 패치 정보(예: ".patch" 패치 파일)	선택
N8	패치된 함수/파일 해시값	취약점이 패치된 함수 및 파일의 해시값 (해시 생성 툴 공개 예정)	필수

본고에서 소개하는 표준의 목적은 소스코드 레벨에서 발생하는 취약점의 탐지 및 해결을 위한 소프트웨어 취약점 데이터베이스의 구조화된 정보 표현 규격을 정의하는 것이다. 특히 소프트웨어 개발 단계에서 발생할 수 있는 취약점 전파 문제로 인한 보안 위협에 대응할 때 효과적으로 활용될 수 있는 취약점 관련 정보들을 일관성 있게 제공하는 데 그 목적이 있다.

3.2 표준 내용

<표 2>는 표준에서 정의하는, 취약점 탐지 및 해결을 위한 구조화된 소프트웨어 취약점 데이터베이스 정보 표현 규격을 나타낸다. 현재의 소프트웨어 취약점 데이터베이스가 제공하는 정보들은 취약점으로 인해 발생하는 보안 위협을 탐지 및 해결하는 데 도움이 되는 정보들을 일관성 있게 제공해주지 않는다. 따라서 소프트웨어 취약점으로 인해 발생할 수 있는 보안 위협을 완화하는데 효율적인 정보들, 즉 취약점 탐지 및 해결에 실질적으로 활용될 수 있는 정보들이 일관성 있게 제공될 수 있도록 구조화된 정보 표현 규격을 제안하였다.

T1부터 T8까지는 기존 취약점 데이터베이스에서 제공하는 정보들이며, N1부터 N8까지는 표준에서 새로 제안하는 정보 표현 규격이다.

3.3 표준 활용 방안

본고에서 제시하는 취약점 데이터베이스의 정보 표현 규격은 소프트웨어 내의 취약점을 탐지 및 해결에 효율적으로 활용될 수 있다. <표 3>은 본고에서 제안하는 취약점 정보 표현 규격을 취약점 탐지 및 해결의 관점에서 분류한 결과이다. 단일 원(○)으로 표기된 정보는 선택 정보이며, 이중 원(◎)으로 표기된 정보는 필수 정보를 나타

<표 3> 소프트웨어 취약점 데이터베이스 정보 표현 규격

번호	예시	탐지	해결	메타데이터
T1	CVE-2020-14147			◎
T2	An integer overflow in the “getnum” function in ...			◎
T3	CVSS 7.7 (High severity)			◎
T4	CWE-190: Integer Overflow or Wraparound			◎
T5	Redis	◎	◎	
T6	6.0.0, 6.0.1, 6.0.2, ...	◎	◎	
T7	https://github.com/antirez/redis/pull/6875, ...			◎
T8	Add references at 7/29/2021, ...			◎
N1	Lua	◎		
N2	5.1	◎		
N3	static int getnum (const char **fmt, int df) { ... }	○		
N4	34c4dcf8615caa1f3be710927440bad2	◎		
N5	https://github.com/antirez/redis/pull/6875	○		
N6	Redis/deps/lua/src/lua_struct.c	○	○	
N7	https://github.com/redis/redis/commit/ef764dde1cca2f25d00686673d1bc89448819571		○	
N8	5e0589b36f32034479d2f83994b37d6c		◎	

낸다. 메타데이터로 분류된 항목들은 취약점과 관련된 정보를 제공함으로써 취약점 탐지 및 해결에 간접적으로 활용할 수 있다.

3.3.1. 취약점 탐지 과정에서의 활용 방안

<표 3>의 내용에 따라, 원점 소프트웨어 이름 및 버전, 취약 함수 혹은 파일의 소스코드/해시값, 취약점 재현 방법, 취약 소프트웨어 및 취약점 위치정보는 취약점 탐지에 도움이 되는 주 정보를 제공해주며, 부수적으로 메타데이터에 속하는 상세 취약점 정보나 참고 자료에 나타나 있는 정보들을 활용할 수 있다.

개발자들이 소프트웨어 명세서를 통해 재사용 중인 소프트웨어 컴포넌트를 파악하고 있다고 하면, 취약 소프트웨어 정보를 통해 취약점의 존재 여부를 판단할 수 있다. 또한 취약 함수의 소스코드와 위치가 제공된다면 직접 해당 파일을 검토하여 취약점 존재 여부를 판단할 수 있다. 추가적으로 취약 함수의 소스코드가 제공되지 않는 경우에는 취약 함수의 해시값을 활용할 수도 있다. 개발자들은 소프트웨어의 코드베이스 내의 모든 함수에 대한 해시값을 계산한 후, 취약 함수의 해시값과 같은 해시값을 갖는 함수가 있는지를 확인하여, 해당 취약점이 존재한다면 이를 탐지해 낼 수 있다.

3.3.2. 취약점 해결과정에서의 활용 방안

<표 3>에 기술되어 있는 정보 중 취약점 해결 항목에 속하는 취약 소프트웨어 이름 및 버전 정보, 취약점 위치정보, 취약점 패치 정보 및 패치된 함수 혹은 파일의 해시값 정보가 취약점을 해결하는데 필요한 핵심 정보들을 제공하며, 기타 메타데이터에 속하는 상세 취약점 정보나 참고 자료를 부수적으로 활용할 수 있다.

개발자들은 취약한 소프트웨어 및 버전 정보를 참고하여 재사용하는 소프트웨어를 안전한 버전으로 업데이트할 수 있다. 또한 취약점 패치 정보가 제공된다면 직접 코드 레벨 패치를 적용하는 방식으로 취약점을 해결할 수 있고, 취약점 패치 파일(".patch")을 그대로 적용하여 취약점을 해결할 수 있다. 패치를 적용한 후, 해당 함수의 해시값을 본고에서 소개하는 표준의 취약점 정보 표현 규격에서 명시하는 패치 된 함수의 해시값과 비교하여 취약점이 올바르게 패치되었는지 최종 검증할 수 있다.

4. 맺음말

본고에서는 소스코드 레벨에서 발생하는 취약점의 탐지 및 해결을 위한 구조화된 취약점 정보 표현 규격을 소개했다. 기존 취약점 데이터베이스 기술 및 표준은 취약점 존재 여부를 판단하거나, 탐지된 취약점을 해결할 수 있는 정보를 필수적으로 제공하지 않아, 개발자들이 활용하는 데 어려움이 있었다. 본고에서 소개하는 표준은 취약점에 대한 메타데이터 정보뿐 아니라 실제 취약점 탐지 및 해결에 필요한 정보까지 필수적으로 제공함에 따라, 보안 위협 예방 및 조기 대응에 유용하게 활용될 수 있다.

※ 본 연구는 과학기술정보통신부 및 정보통신기획평가원의 ICT명품인재양성 사업(IITP-2023-2020-0-01819), 지역전략산업 융합보안 핵심인재 양성 사업(No.2022-0-01198) 및 SW공급망 보안을 위한 SBOM 자동생성 및 무결성 검증기술 개발 사업(No.2022-0-00277) 결과로 수행되었음

주요 용어 풀이

- **취약점 정보** : 본고에서 소프트웨어 취약점 정보는 취약점 메타데이터, 취약점 탐지에 도움이 되는 정보, 취약점 해결에 도움이 되는 정보를 포괄하는 것으로 정의함. 취약점 메타데이터는 취약점의 식별자 및 취약점 설명과 같은 취약점 자체의 정보를 제공해주는 데이터를 말하며, 취약점 탐지 및 해결에 도움이 되는 정보는 실제 취약점으로부터 인한 보안 위협을 완화하기 위해 취약점을 탐지 및 해결하는 데 도움이 되는 정보들을 지칭함
- **공개 소프트웨어** : 저작권자가 소스 코드를 개방하여, 원 저작자의 라이선스 규칙에 따르는 한 누구나 자유롭게 소스코드를 수정 및 재배포할 수 있는 소프트웨어
- **사유 소프트웨어** : 공개 소프트웨어와 달리 저작권 소유자의 예외적인 법적 권한 하에서 허가된 소프트웨어
- **소프트웨어 취약점** : 소프트웨어 시스템에서 제기되는 보안상의 결점. 프로그램이 의도대로 동작하지 않는 버그 들 중에 악용할 수 있는 결점을 말하며 정보 유출, 권한 상승 등을 일으킬 수 있음
- **취약 함수** : 소프트웨어의 내의 함수들 중 실제 취약점이 내재 되어 있는 함수
- **취약점 패치** : 공격에 악용될 수 있는 취약한 소스 코드를 수정하여 안전한 코드로 업데이트하는 행위
- **해시 알고리즘** : 임의의 길이의 데이터를 고정된 길이의 데이터로 변환하는 함수를 말함. 해시 알고리즘을 적용하여 나온 고정된 길이의 값을 해시값(hash value)이라고 지칭함
- **취약 코드가 최초로 생성된 원점 소프트웨어** : 특정 취약점이 포함된 소스코드가 가장 먼저 생성된 소프트웨어

참고문헌

[1] TTAK.KO-12.0384, 취약점 탐지 및 해결을 위한 구조화된 소프트웨어 취약점 데이터베이스 정보 표현 규격