(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2020년01월17일
(11) 등록번호 10-2067733
(24) 등록일자 2020년01월13일

(51) 국제특허분류(Int. Cl.)
G06F 21/57 (2013.01) *G06F 21/56* (2013.01)
(52) CPC특허분류
G06F 21/577 (2013.01)
G06F 11/3636 (2013.01)
(21) 출원번호 10-2019-0057045
(22) 출원일자 2019년05월15일
심사청구일자 2019년05월15일
(56) 선행기술조사문헌
KR101568872 B1*
(뒷면에 계속)

(73) 특허권자
세종대학교산학협력단
서울특별시 광진구 능동로 209 (군자동, 세종대학교)
(72) 발명자
윤주범
서울특별시 송파구 충민로4길 19, 704동 401호(장지동, 송파파인타운7단지)
유지현
경기도 부천시 조마루로371번길 52, 402호(춘의동, 삼성리치빌)
(74) 대리인
두호특허법인

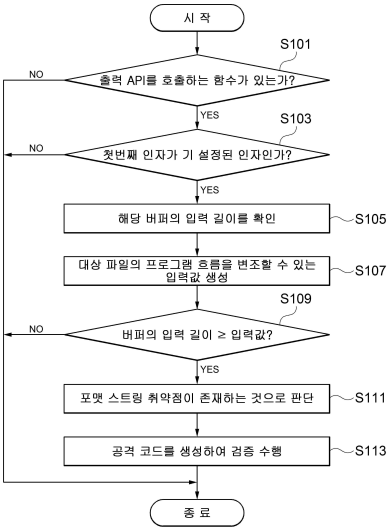
전체 청구항 수 : 총 18 항 심사관 : 구대성

(54) 발명의 명칭 **포맷 스트링 취약점 검출 방법 및 이를 수행하기 위한 장치**

(57) 요 약

포맷 스트링 취약점 검출 방법 및 이를 수행하기 위한 장치가 개시된다. 개시되는 일 실시예에 따른 포맷 스트링 검출 장치는, 하나 이상의 프로세서들, 및 하나 이상의 프로세서들에 의해 실행되는 하나 이상의 프로그램들을 저장하는 메모리를 구비한 컴퓨팅 장치로서, 대상 파일에 출력 API를 호출하는 함수가 있는 경우, 함수를 후킹하는 후킹 모듈 및 후킹한 함수에 기반하여 대상 파일에 포맷 스트링 취약점이 있는지 여부를 판단하는 취약점 판단 모듈을 포함한다.

대 표 도 - 도2



(52) CPC특허분류
G06F 21/563 (2013.01)

(56) 선행기술조사문헌
KR1020010056714 A*
KR1020100089256 A*
JP2007052625 A
KR1020090084530 A
*는 심사관에 의하여 인용된 문헌

이 발명을 지원한 국가연구개발사업
과제고유번호 1711075702
부처명 과학기술정보통신부
연구관리전문기관 정보통신기획평가원
연구사업명 대학ICT연구센터지원사업
연구과제명 지능형 비행로봇 융합기술 연구
기 여 율 1/1
주관기관 세종대학교 산학협력단
연구기간 2018.06.01 ~ 2021.12.31

명 세 서

청구범위

청구항 1

하나 이상의 프로세서들, 및

상기 하나 이상의 프로세서들에 의해 실행되는 하나 이상의 프로그램들을 저장하는 메모리를 구비한 컴퓨팅 장치로서,

대상 파일에 출력 API를 호출하는 함수-상기 출력 API를 호출하는 함수는, 상기 대상 파일이 실행되는 컴퓨팅 장치에서 해당 함수의 명령에 대응하는 결과를 디스플레이에 출력하기 위한 함수임-가 있는지 여부를 확인하고, 상기 출력 API를 호출하는 함수가 있는 경우, 상기 함수를 후킹하는 후킹 모듈; 및

상기 후킹한 함수의 인자에 사용자로부터 입력되는 값을 입력으로 하는 버퍼가 존재하고, 상기 버퍼가 상기 후킹한 함수의 첫 번째 인자인 경우, 상기 대상 파일에 포맷 스트링 취약점이 있는지 여부를 판단하는 취약점 판단 모듈을 포함하는, 포맷 스트링 검출 장치.

청구항 2

청구항 1에 있어서,

상기 취약점 판단 모듈은,

상기 첫 번째 인자가 상기 버퍼인 경우, 상기 버퍼에 설정된 메모리를 로딩하고, 상기 메모리에서 상기 버퍼의 입력 길이를 확인하는, 포맷 스트링 검출 장치.

청구항 3

삭제

청구항 4

청구항 2에 있어서,

상기 취약점 판단 모듈은,

상기 첫 번째 인자가 상기 버퍼인 경우, 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값을 생성하는, 포맷 스트링 검출 장치.

청구항 5

청구항 4에 있어서,

상기 취약점 판단 모듈은,

상기 버퍼의 입력 길이와 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값의 길이를 비교하여 상기 대상 파일에 포맷 스트링 취약점이 있는지 여부를 판단하는, 포맷 스트링 검출 장치.

청구항 6

청구항 5에 있어서,

상기 취약점 판단 모듈은,

상기 버퍼의 입력 길이가 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값의 길이 이상인 경우 상기 대상 파일에 포맷 스트링 취약점이 있는 것으로 판단하고, 상기 버퍼의 입력 길이가 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값의 길이 미만인 경우 상기 대상 파일에 포맷 스트링 취약점이 없는 것으로 판단하는, 포맷 스트링 검출 장치.

청구항 7

청구항 5에 있어서,

상기 포맷 스트링 검출 장치는,

공격 코드를 생성하고, 상기 공격 코드를 상기 포맷 스트링 취약점이 있는 것으로 판단된 대상 파일에 대해 적용하여 포맷 스트링 취약점 검증을 수행하는 검증 모듈을 더 포함하는, 포맷 스트링 검출 장치.

청구항 8

청구항 7에 있어서,

상기 검증 모듈은,

상기 대상 파일의 입력 유형, 상기 대상 파일에 적용된 보호 기법, 상기 대상 파일 내 취약한 함수 부분의 주소, 및 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값의 길이 중 하나 이상을 기반으로 상기 공격 코드를 생성하는, 포맷 스트링 검출 장치.

청구항 9

청구항 8에 있어서,

상기 검증 모듈은,

상기 공격 코드를 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값과 동일한 길이로 생성하는, 포맷 스트링 검출 장치.

청구항 10

하나 이상의 프로세서들, 및

상기 하나 이상의 프로세서들에 의해 실행되는 하나 이상의 프로그램들을 저장하는 메모리를 구비한 컴퓨팅 장치에서 수행되는 방법으로서,

대상 파일에 출력 API를 호출하는 함수-상기 출력 API를 호출하는 함수는, 상기 대상 파일이 실행되는 컴퓨팅 장치에서 해당 함수의 명령에 대응하는 결과를 디스플레이에 출력하기 위한 함수임-가 있는지 여부를 확인하는 동작;

상기 출력 API를 호출하는 함수가 있는 경우, 상기 함수를 후킹하는 동작; 및

상기 후킹한 함수의 인자에 사용자로부터 입력되는 값을 입력으로 하는 버퍼가 존재하고, 상기 버퍼가 상기 후킹한 함수의 첫 번째 인자인 경우, 상기 대상 파일에 포맷 스트링 취약점이 있는지 여부를 판단하는 동작을 포함하는, 포맷 스트링 검출 방법.

청구항 11

청구항 10에 있어서,

상기 포맷 스트링 취약점이 있는지 여부를 판단하는 동작은,

상기 첫 번째 인자가 상기 버퍼인 경우, 상기 버퍼에 설정된 메모리를 로딩하는 동작; 및

상기 메모리에서 상기 버퍼의 입력 길이를 확인하는 동작을 포함하는, 포맷 스트링 검출 방법.

청구항 12

삭제

청구항 13

청구항 11에 있어서,

상기 포맷 스트링 취약점이 있는지 여부를 판단하는 동작은,

상기 첫 번째 인자가 상기 버퍼인 경우, 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값을 생성하는 동작을 더 포함하는, 포맷 스트링 검출 방법.

청구항 14

청구항 13에 있어서,

상기 포맷 스트링 취약점이 있는지 여부를 판단하는 동작은,

상기 버퍼의 입력 길이와 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값의 길이를 비교하여 상기 대상 파일에 포맷 스트링 취약점이 있는지 여부를 판단하는 동작을 더 포함하는, 포맷 스트링 검출 방법.

청구항 15

청구항 14에 있어서,

상기 포맷 스트링 취약점이 있는지 여부를 판단하는 동작은,

상기 버퍼의 입력 길이가 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값의 길이 이상인 경우 상기 대상 파일에 포맷 스트링 취약점이 있는 것으로 판단하는 동작; 및

상기 버퍼의 입력 길이가 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값의 길이 미만인 경우 상기 대상 파일에 포맷 스트링 취약점이 없는 것으로 판단하는 동작을 포함하는, 포맷 스트링 검출 방법.

청구항 16

청구항 14에 있어서,

상기 포맷 스트링 검출 방법은,

공격 코드를 생성하고, 상기 공격 코드를 상기 포맷 스트링 취약점이 있는 것으로 판단된 대상 파일에 대해 적용하여 포맷 스트링 취약점 검증을 수행하는 동작을 더 포함하는, 포맷 스트링 검출 방법.

청구항 17

청구항 16에 있어서,

상기 검증을 수행하는 동작은,

상기 대상 파일의 입력 유형, 상기 대상 파일에 적용된 보호 기법, 상기 대상 파일 내 취약한 함수 부분의 주소, 및 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값의 길이 중 하나 이상을 기반으로 상기 공격 코드를 생성하는, 포맷 스트링 검출 방법.

청구항 18

청구항 17에 있어서,

상기 검증을 수행하는 동작은,

상기 공격 코드를 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값과 동일한 길이로 생성하는, 포맷 스트링 검출 방법.

청구항 19

하나 이상의 프로세서들;

메모리; 및

하나 이상의 프로그램들을 포함하고,

상기 하나 이상의 프로그램들은 상기 메모리에 저장되고, 상기 하나 이상의 프로세서들에 의해 실행되도록 구성되며,

상기 하나 이상의 프로그램들은,

대상 파일에 출력 API를 호출하는 함수-상기 출력 API를 호출하는 함수는, 상기 대상 파일이 실행되는 컴퓨팅 장치에서 해당 함수의 명령에 대응하는 결과를 디스플레이에 출력하기 위한 함수임-가 있는지 여부를 확인하기 위한 명령;

상기 출력 API를 호출하는 함수가 있는 경우, 상기 함수를 후킹하기 위한 명령; 및

상기 후킹한 함수의 인자에 사용자로부터 입력되는 값을 입력으로 하는 버퍼가 존재하고, 상기 버퍼가 상기 후킹한 함수의 첫 번째 인자인 경우, 상기 대상 파일에 포맷 스트링 취약점이 있는지 여부를 판단하기 위한 명령을 포함하는, 컴퓨팅 장치.

청구항 20

비일시적 컴퓨터 판독 가능한 저장 매체(non-transitory computer readable storage medium)에 저장된 컴퓨터 프로그램으로서,

상기 컴퓨터 프로그램은 하나 이상의 명령어들을 포함하고, 상기 명령어들은 하나 이상의 프로세서들을 갖는 컴퓨팅 장치에 의해 실행될 때, 상기 컴퓨팅 장치로 하여금,

대상 파일에 출력 API를 호출하는 함수-상기 출력 API를 호출하는 함수는, 상기 대상 파일이 실행되는 컴퓨팅 장치에서 해당 함수의 명령에 대응하는 결과를 디스플레이에 출력하기 위한 함수임-가 있는지 여부를 확인하도록 하고,

상기 출력 API를 호출하는 함수가 있는 경우, 상기 함수를 후킹하도록 하고, 그리고

상기 후킹한 함수의 인자에 사용자로부터 입력되는 값을 입력으로 하는 버퍼가 존재하고, 상기 버퍼가 상기 후킹한 함수의 첫 번째 인자인 경우, 상기 대상 파일에 포맷 스트링 취약점이 있는지 여부를 판단하도록 하는, 컴퓨터 프로그램.

발명의 설명

기 술 분 야

[0001] 본 발명의 실시예는 포맷 스트링 취약점 검출 기술과 관련된다.

배 경 기 술

[0003] 포맷 스트링(Format String) 취약점이란 사용자의 입력에 의해 프로그램 실행 흐름을 변경시킬 수 있는 취약점으로, 포맷 스트링을 잘못된 형태로 사용할 경우 발생하는 취약점이다. 포맷 스트링은 데이터를 어떤 형식에 따라 입력받거나 출력하기 위하여 사용하는데 그 형식을 오용하게 되면 메모리의 데이터가 유출되고 변조될 수 있다.

[0004] 기존에는 포맷 스트링 취약점을 수동으로 탐지하였는데, 수동으로 포맷 스트링 취약점을 탐지하려면 소스코드가 있거나 바이너리 실행 파일에서 디스어셈블 등을 통해 소스코드를 추출하여 분석가가 분석할 수 있는 형태로 만들어야 한다. 그 후, 코드를 하나하나 분석하면서 포맷스트링이 사용되는 함수마다 적절히 사용되었는지, 인자가 제대로 대응하여 사용자의 입력으로 프로그램의 흐름이 변경되지 않는지 확인해야 한다.

[0005] 이와 같이, 포맷 스트링 취약점을 수동으로 탐지하는 일은 많은 인력과 비용이 필요하고, 아주 복잡한 프로그램의 경우 수동으로 코드를 분석하는 일은 거의 불가능에 가깝다.

선행기술문헌

특허문헌

[0007] (특허문헌 0001) 한국공개특허공보 제10-2018-0010053호(2018.01.30)

발명의 내용

해결하려는 과제

[0008] 개시되는 실시예는 포맷 스트링 취약점을 자동으로 검출할 수 있는 포맷 스트링 취약점 검출 방법 및 이를 수행하기 위한 장치를 제공하기 위한 것이다.

과제의 해결 수단

[0010] 개시되는 일 실시예에 따른 포맷 스트링 검출 장치는, 하나 이상의 프로세서들, 및 상기 하나 이상의 프로세서들에 의해 실행되는 하나 이상의 프로그램들을 저장하는 메모리를 구비한 컴퓨팅 장치로서, 대상 파일에 출력 API를 호출하는 함수가 있는 경우, 상기 함수를 후킹하는 후킹 모듈; 및 상기 후킹한 함수에 기반하여 상기 대상 파일에 포맷 스트링 취약점이 있는지 여부를 판단하는 취약점 판단 모듈을 포함한다.

[0011] 상기 취약점 판단 모듈은, 상기 후킹한 함수의 첫 번째 인자가 기 설정된 인자인지 여부를 확인하고, 상기 첫 번째 인자가 기 설정된 인자인 경우, 상기 인자에 설정된 메모리를 로딩하고, 상기 메모리에서 상기 인자의 입력 길이를 확인할 수 있다.

[0012] 상기 기 설정된 인자는, 버퍼(buffer)이고, 상기 버퍼는 해당 버퍼에 설정된 메모리에서 사용자가 입력하는 값을 입력으로 하는 인자일 수 있다.

[0013] 상기 취약점 판단 모듈은, 상기 첫 번째 인자가 기 설정된 인자인 경우, 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값을 생성할 수 있다.

[0014] 상기 취약점 판단 모듈은, 상기 인자의 입력 길이와 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값의 길이를 비교하여 상기 대상 파일에 포맷 스트링 취약점이 있는지 여부를 판단할 수 있다.

[0015] 상기 취약점 판단 모듈은, 상기 인자의 입력 길이가 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값의 길이 이상인 경우 상기 대상 파일에 포맷 스트링 취약점이 있는 것으로 판단하고, 상기 인자의 입력 길이가 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값의 길이 미만인 경우 상기 대상 파일에 포맷 스트링 취약점이 없는 것으로 판단할 수 있다.

[0016] 상기 포맷 스트링 검출 장치는, 공격 코드를 생성하고, 상기 공격 코드를 상기 포맷 스트링 취약점이 있는 것으

로 판단된 대상 파일에 대해 적용하여 포맷 스트링 취약점 검증을 수행하는 검증 모듈을 더 포함할 수 있다.

[0017] 상기 검증 모듈은, 상기 대상 파일의 입력 유형, 상기 대상 파일에 적용된 보호 기법, 상기 대상 파일 내 취약한 함수 부분의 주소, 및 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값의 길이 중 하나 이상을 기반으로 상기 공격 코드를 생성할 수 있다.

[0018] 상기 검증 모듈은, 상기 공격 코드를 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값과 동일한 길이로 생성할 수 있다.

[0019] 개시되는 일 실시예에 따른 포맷 스트링 취약점 검출 방법은, 하나 이상의 프로세서들, 및 상기 하나 이상의 프로세서들에 의해 실행되는 하나 이상의 프로그램들을 저장하는 메모리를 구비한 컴퓨팅 장치에서 수행되는 방법으로서, 대상 파일에 출력 API를 호출하는 함수가 있는지 여부를 확인하는 동작; 상기 출력 API를 호출하는 함수가 있는 경우, 상기 함수를 후킹하는 동작; 및 상기 후킹한 함수에 기반하여 상기 대상 파일에 포맷 스트링 취약점이 있는지 여부를 판단하는 동작을 포함한다.

[0020] 상기 포맷 스트링 취약점이 있는지 여부를 판단하는 동작은, 상기 후킹한 함수의 첫 번째 인자가 기 설정된 인자인지 여부를 확인하는 동작; 상기 첫 번째 인자가 기 설정된 인자인 경우, 상기 인자에 설정된 메모리를 로딩하는 동작; 및 상기 메모리에서 상기 인자의 입력 길이를 확인하는 동작을 포함할 수 있다.

[0021] 상기 기 설정된 인자는, 버퍼(buffer)이고, 상기 버퍼는 해당 버퍼에 설정된 메모리에서 사용자가 입력하는 값을 입력으로 하는 인자일 수 있다.

[0022] 상기 포맷 스트링 취약점이 있는지 여부를 판단하는 동작은, 상기 첫 번째 인자가 기 설정된 인자인 경우, 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값을 생성하는 동작을 더 포함할 수 있다.

[0023] 상기 포맷 스트링 취약점이 있는지 여부를 판단하는 동작은, 상기 인자의 입력 길이와 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값의 길이를 비교하여 상기 대상 파일에 포맷 스트링 취약점이 있는지 여부를 판단하는 동작을 더 포함할 수 있다.

[0024] 상기 포맷 스트링 취약점이 있는지 여부를 판단하는 동작은, 상기 인자의 입력 길이가 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값의 길이 이상인 경우 상기 대상 파일에 포맷 스트링 취약점이 있는 것으로 판단하는 동작; 및 상기 인자의 입력 길이가 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값의 길이 미만인 경우 상기 대상 파일에 포맷 스트링 취약점이 없는 것으로 판단하는 동작을 포함할 수 있다.

[0025] 상기 포맷 스트링 검출 방법은, 공격 코드를 생성하고, 상기 공격 코드를 상기 포맷 스트링 취약점이 있는 것으로 판단된 대상 파일에 대해 적용하여 포맷 스트링 취약점 검증을 수행하는 동작을 더 포함할 수 있다.

[0026] 상기 검증을 수행하는 동작은, 상기 대상 파일의 입력 유형, 상기 대상 파일에 적용된 보호 기법, 상기 대상 파일 내 취약한 함수 부분의 주소, 및 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값의 길이 중 하나 이상을 기반으로 상기 공격 코드를 생성할 수 있다.

[0027] 상기 검증을 수행하는 동작은, 상기 공격 코드를 상기 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값과 동일한 길이로 생성할 수 있다.

[0028] 개시되는 일 실시예에 따른 컴퓨팅 장치는, 하나 이상의 프로세서들; 메모리; 및 하나 이상의 프로그램들을 포함하고, 상기 하나 이상의 프로그램들은 상기 메모리에 저장되고, 상기 하나 이상의 프로세서들에 의해 실행되도록 구성되며, 상기 하나 이상의 프로그램들은, 대상 파일에 출력 API를 호출하는 함수가 있는지 여부를 확인하기 위한 명령; 상기 출력 API를 호출하는 함수가 있는 경우, 상기 함수를 후킹하기 위한 명령; 및 상기 후킹한 함수에 기반하여 상기 대상 파일에 포맷 스트링 취약점이 있는지 여부를 판단하기 위한 명령을 포함한다.

발명의 효과

[0030] 개시되는 실시예에 의하면, 대상 파일에 포맷 스트링 취약점이 존재하는지를 동적 분석을 통해 자동으로 검출하고 검증할 수 있으므로, 기존의 포맷 스트링 취약점을 탐지하는데 소요되는 시간 및 노력을 현저히 줄일 수 있게 된다.

도면의 간단한 설명

[0032] 도 1은 본 발명의 일 실시예에 따른 포맷 스트링 취약점 검출 장치의 구성을 나타낸 도면

도 2는 본 발명의 일 실시예에 따른 포맷 스트링 취약점 검출 방법을 설명하기 위한 흐름도

도 3은 예시적인 실시예들에서 사용되기에 적합한 컴퓨팅 장치를 포함하는 컴퓨팅 환경을 예시하여 설명하기 위한 블록도

발명을 실시하기 위한 구체적인 내용

[0033] 이하, 도면을 참조하여 본 발명의 구체적인 실시형태를 설명하기로 한다. 이하의 상세한 설명은 본 명세서에서 기술된 방법, 장치 및/또는 시스템에 대한 포괄적인 이해를 돕기 위해 제공된다. 그러나 이는 예시에 불과하며 본 발명은 이에 제한되지 않는다.

[0034] 본 발명의 실시예들을 설명함에 있어서, 본 발명과 관련된 공지기술에 대한 구체적인 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명을 생략하기로 한다. 그리고, 후술되는 용어들은 본 발명에서의 기능을 고려하여 정의된 용어들로서 이는 사용자, 운용자의 의도 또는 관례 등에 따라 달라질 수 있다. 그러므로 그 정의는 본 명세서 전반에 걸친 내용을 토대로 내려져야 할 것이다. 상세한 설명에서 사용되는 용어는 단지 본 발명의 실시예들을 기술하기 위한 것이며, 결코 제한적이어서는 안 된다. 명확하게 달리 사용되지 않는 한, 단수 형태의 표현은 복수 형태의 의미를 포함한다. 본 설명에서, "포함" 또는 "구비"와 같은 표현은 어떤 특성들, 숫자들, 단계들, 동작들, 요소들, 이들의 일부 또는 조합을 가리키기 위한 것이며, 기술된 것 이외에 하나 또는 그 이상의 다른 특성, 숫자, 단계, 동작, 요소, 이들의 일부 또는 조합의 존재 또는 가능성을 배제하도록 해석되어서는 안 된다.

[0035] 이하의 설명에 있어서, 신호 또는 정보의 "전송", "통신", "송신", "수신" 기타 이와 유사한 의미의 용어는 일 구성요소에서 다른 구성요소로 신호 또는 정보가 직접 전달되는 것뿐만이 아니라 다른 구성요소를 거쳐 전달되는 것도 포함한다. 특히 신호 또는 정보를 일 구성요소로 "전송" 또는 "송신"한다는 것은 그 신호 또는 정보의 최종 목적지를 지시하는 것이고 직접적인 목적지를 의미하는 것이 아니다. 이는 신호 또는 정보의 "수신"에 있어서도 동일하다. 또한 본 명세서에 있어서, 2 이상의 데이터 또는 정보가 "관련"된다는 것은 하나의 데이터(또는 정보)를 획득하면, 그에 기초하여 다른 데이터(또는 정보)의 적어도 일부를 획득할 수 있음을 의미한다.

[0037] 도 1은 본 발명의 일 실시예에 따른 포맷 스트링 취약점 검출 장치의 구성을 나타낸 도면이다.

[0038] 도 1을 참조하면, 포맷 스트링 취약점 검출 장치(100)는 후킹 모듈(102), 취약점 판단 모듈(104), 및 검증 모듈(106)을 포함할 수 있다. 예시적인 실시예에서, 포맷 스트링 취약점 검출 장치(100)는 대상 파일(즉, 포맷 스트링 취약점이 있는지 여부를 확인하기 위한 파일)을 시뮬레이션을 통해 가상 실행시키는 동적 분석 방식으로 포맷 스트링 취약점을 검출 및 검증할 수 있다.

[0039] 일 실시예에서, 후킹 모듈(102), 취약점 판단 모듈(104), 및 검증 모듈(106)은 물리적으로 구분된 하나 이상의 장치를 이용하여 구현되거나, 하나 이상의 프로세서 또는 하나 이상의 프로세서 및 소프트웨어의 결합에 의해 구현될 수 있으며, 도시된 예와 달리 구체적 동작에 있어 명확히 구분되지 않을 수 있다.

[0040] 후킹 모듈(102)은 입력되는 대상 파일(즉, 포맷 스트링 취약점이 있는지 여부를 확인하기 위한 파일)을 분석하여 대상 파일에 출력 API(Application Programming Interface)를 호출하는 함수가 있는지 여부를 확인할 수 있다. 후킹 모듈(102)은 대상 파일에 출력 API를 호출하는 함수가 있는 경우, 해당 함수를 후킹하여 획득할 수 있다.

[0041] 후킹 모듈(102)로 입력되는 대상 파일은 바이너리(Binary) 파일 일 수 있다. 후킹 모듈(102)은 대상 파일을 역어셈블링 한 후 대상 파일에 출력 API를 호출하는 함수가 있는지 여부를 확인할 수 있다. 여기서, 출력 API를 호출하는 함수는 대상 파일이 실행되는 컴퓨팅 장치에서 해당 함수의 명령에 대응하는 결과를 디스플레이(화면)에 출력하기 위한 함수일 수 있다. 출력 API를 호출하는 함수로는 예를 들어, 프린트에프 함수군(printf, sprintf, fprintf, vsprintf) 또는 puts 등이 있다. 대상 파일에 출력 API를 호출하는 함수가 있는 경우, 후킹 모듈(102)은 해당 함수를 후킹할 수 있다.

[0042] 취약점 판단 모듈(104)은 후킹 모듈(102)에서 후킹한 함수에 기반하여 대상 파일에 포맷 스트링 취약점이 있는지 여부를 판단할 수 있다. 구체적으로, 취약점 판단 모듈(104)은 출력 API를 호출하는 함수의 첫 번째 인자가 기 설정된 인자인지 여부를 확인할 수 있다. 여기서, 인자란 해당 함수에 전달되는 값으로, 해당 함수가 소정 기능을 수행하도록 하는 명령어를 의미할 수 있다. 예시적인 실시예에서, 기 설정된 인자는 버퍼(buffer)일 수 있다.

[0043] 여기서, 버퍼(buffer)는 해당 버퍼에 설정된 메모리에서 사용자가 입력하는 값을 입력으로 하는 인자이다. 이하에서는, 기 설정된 인자가 버퍼인 것으로 하여 설명하기로 한다. 출력 API를 호출하는 함수의 첫 번째 인자가 버퍼인 경우, 버퍼는 고정된 값이 아닌 사용자가 입력하는 값을 입력으로 하기 때문에 프로그램의 흐름을 변조시킬 수 있는 인자가 될 수 있다.

[0044] 예를 들어, 출력 API를 호출하는 함수가 printf(buf)인 경우, 출력 API를 호출하는 함수의 첫 번째 인자가 기 설정된 인자인 버퍼(buffer)에 해당하게 된다. 반면, 출력 API를 호출하는 함수가 printf("hello world %s, buf)인 경우, 출력 API를 호출하는 함수의 첫 번째 인자는 %s로서, 기 설정된 인자에 해당하지 않게 된다.

[0045] 출력 API를 호출하는 함수의 첫 번째 인자가 기 설정된 인자(즉, 버퍼(buffer))인 경우, 취약점 판단 모듈(104)은 해당 버퍼에 대응하는 메모리를 로딩할 수 있다. 그리고, 취약점 판단 모듈(104)은 로딩한 메모리에서 해당 버퍼의 입력 길이를 확인할 수 있다.

[0046] 또한, 출력 API를 호출하는 함수의 첫 번째 인자가 기 설정된 인자(즉, 버퍼(buffer))인 경우, 취약점 판단 모듈(104)은 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값을 생성할 수 있다. 예를 들어, 취약점 판단 모듈(104)은 return 주소에 shellcode를 삽입한 것과 같은 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값을 생성할 수 있다.

[0047] 취약점 판단 모듈(104)은 해당 버퍼의 입력 길이가 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값 이상인지 여부를 확인할 수 있다. 즉, 취약점 판단 모듈(104)은 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값이 해당 메모리에서 해당 버퍼에 할당된 입력 길이에 들어갈 수 있는지 여부를 확인할 수 있다.

[0048] 취약점 판단 모듈(104)은 해당 버퍼의 입력 길이가 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값 이상인 경우, 대상 파일에 포맷 스트링 취약점이 존재하는 것으로 판단할 수 있다. 해당 버퍼의 입력 길이가 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값보다 작은 경우, 취약점 판단 모듈(104)은 대상 파일에 포맷 스트링 취약점이 존재하지 않는 것으로 판단할 수 있다.

[0049] 검증 모듈(106)은 포맷 스트링 취약점이 있는 것으로 판단된 대상 파일에 대해 포맷 스트링 취약점 검증을 수행할 수 있다. 검증 모듈(106)은 포맷 스트링 취약점을 이용한 공격 코드를 생성하고, 생성한 공격 코드를 대상 파일에 실행시켜 포맷 스트링 취약점 검증을 수행할 수 있다.

[0050] 예시적인 실시예에서, 검증 모듈(106)은 대상 파일의 입력 유형(예를 들어, stdin 또는 argument 등), 대상 파일에 적용된 보호 기법(예를 들어, canary 또는 NX 등), 대상 파일 내 취약한 함수 부분의 주소(Winfunc address)(예를 들어, return 함수가 들어있는 main 함수 또는 system 함수가 들어있는 사용자 정의 solve 함수의 주소 등), 및 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값의 길이 중 하나 이상을 기반으로 공격 코드를 생성할 수 있다.

[0051] 검증 모듈(106)은 대상 파일의 입력 유형에 따라 공격 코드를 대상 파일에 주입하는 방식을 결정할 수 있다. 검증 모듈(106)은 대상 파일에 적용된 보호 기법에 따라 공격 기법을 결정할 수 있다. 검증 모듈(106)은 공격 코드를 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값의 길이와 대응되는 길이로 생성할 수 있다. 검증 모듈(106)에서 모의적으로 공격 코드를 생성하여 대상 파일에 적용함으로써, 대상 파일이 포맷 스트링 취약점에 의해 실제 공격받는 경우에 대비하여 이를 보완할 수 있다.

[0052] 본 명세서에서 모듈이라 함은, 본 발명의 기술적 사상을 수행하기 위한 하드웨어 및 상기 하드웨어를 구동하기 위한 소프트웨어의 기능적, 구조적 결합을 의미할 수 있다. 예컨대, 상기 "모듈"은 소정의 코드와 상기 소정의 코드가 수행되기 위한 하드웨어 리소스의 논리적인 단위를 의미할 수 있으며, 반드시 물리적으로 연결된 코드를 의미하거나, 한 종류의 하드웨어를 의미하는 것은 아니다.

[0053] 개시되는 실시예에 의하면, 대상 파일에 포맷 스트링 취약점이 존재하는지를 동적 분석을 통해 자동으로 검출하고 검증할 수 있으므로, 기존의 포맷 스트링 취약점을 탐지하는데 소요되는 시간 및 노력을 현저히 줄일 수 있게 된다.

[0055] 도 2는 본 발명의 일 실시예에 따른 포맷 스트링 취약점 검출 방법을 설명하기 위한 흐름도이다. 도 2에 도시된 방법은 예를 들어, 전술한 포맷 취약점 검출 장치(100)에 의해 수행될 수 있다. 도시된 흐름도에서는 상기 방법을 복수 개의 단계로 나누어 기재하였으나, 적어도 일부의 단계들은 순서를 바꾸어 수행되거나, 다른 단계와 결합되어 함께 수행되거나, 생략되거나, 세부 단계들로 나뉘어 수행되거나, 또는 도시되지 않은 하나 이상의 단계가 부가되어 수행될 수 있다.

[0056] 도 2를 참조하면, 포맷 스트링 취약점 검출 장치(100)는 대상 파일을 분석하여 대상 파일에 출력 API를 호출하는 함수가 있는지 여부를 확인한다(S 101).

[0057] 단계 S 101의 확인 결과, 대상 파일에 출력 API를 호출하는 함수가 있는 경우, 포맷 스트링 취약점 검출 장치(100)는 출력 API를 호출하는 함수의 첫 번째 인자가 기 설정된 인자인지 여부를 확인한다(S 103). 실시예에서, 기 설정된 인자는 버퍼(buffer)일 수 있다.

[0058] 단계 S 103의 확인 결과, 출력 API를 호출하는 함수의 첫 번째 인자가 기 설정된 인자인 경우, 포맷 스트링 취약점 검출 장치(100)는 해당 버퍼에 대응하는 메모리를 로딩하고, 로딩한 메모리에서 해당 버퍼의 입력 길이를 확인한다(S 105).

[0059] 다음으로, 포맷 스트링 취약점 검출 장치(100)는 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값을 생성한다(S 107).

[0060] 다음으로, 포맷 스트링 취약점 검출 장치(100)는 해당 버퍼의 입력 길이가 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값 이상인지 여부를 확인한다(S 109).

[0061] 단계 S 109의 확인 결과, 해당 버퍼의 입력 길이가 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값 이상인 경우, 포맷 스트링 취약점 검출 장치(100)는 대상 파일에 포맷 스트링 취약점이 존재하는 것으로 판단한다(S 111).

[0062] 다음으로, 포맷 스트링 취약점 검출 장치(100)는 대상 파일의 입력 유형, 대상 파일에 적용된 보호 기법, 대상 파일 내 취약한 함수 부분의 주소, 및 대상 파일의 프로그램 흐름을 변조할 수 있는 입력 값의 길이 중 하나 이상을 기반으로 공격 코드를 생성하고, 생성한 공격 코드를 대상 파일에 적용하여 포맷 스트링 취약점 검증을 수행한다(S 113).

[0064] 도 3은 예시적인 실시예들에서 사용되기에 적합한 컴퓨팅 장치를 포함하는 컴퓨팅 환경(10)을 예시하여 설명하기 위한 블록도이다. 도시된 실시예에서, 각 컴포넌트들은 이하에 기술된 것 이외에 상이한 기능 및 능력을 가질 수 있고, 이하에 기술된 것 이외에도 추가적인 컴포넌트를 포함할 수 있다.

[0065] 도시된 컴퓨팅 환경(10)은 컴퓨팅 장치(12)를 포함한다. 일 실시예에서, 컴퓨팅 장치(12)는 포맷 스트링 취약점 검출 장치(100)일 수 있다.

[0066] 컴퓨팅 장치(12)는 적어도 하나의 프로세서(14), 컴퓨터 판독 가능 저장 매체(16) 및 통신 버스(18)를 포함한다. 프로세서(14)는 컴퓨팅 장치(12)로 하여금 앞서 언급된 예시적인 실시예에 따라 동작하도록 할 수 있다. 예컨대, 프로세서(14)는 컴퓨터 판독 가능 저장 매체(16)에 저장된 하나 이상의 프로그램들을 실행할 수 있다. 상기 하나 이상의 프로그램들은 하나 이상의 컴퓨터 실행 가능 명령어를 포함할 수 있으며, 상기 컴퓨터 실행 가능 명령어는 프로세서(14)에 의해 실행되는 경우 컴퓨팅 장치(12)로 하여금 예시적인 실시예에 따른 동작들을 수행하도록 구성될 수 있다.

[0067] 컴퓨터 판독 가능 저장 매체(16)는 컴퓨터 실행 가능 명령어 내지 프로그램 코드, 프로그램 데이터 및/또는 다른 적합한 형태의 정보를 저장하도록 구성된다. 컴퓨터 판독 가능 저장 매체(16)에 저장된 프로그램(20)은 프로세서(14)에 의해 실행 가능한 명령어의 집합을 포함한다. 일 실시예에서, 컴퓨터 판독 가능 저장 매체(16)는 메모리(랜덤 액세스 메모리와 같은 휘발성 메모리, 비휘발성 메모리, 또는 이들의 적절한 조합), 하나 이상의 자기 디스크 저장 디바이스들, 광학 디스크 저장 디바이스들, 플래시 메모리 디바이스들, 그 밖에 컴퓨팅 장치(12)에 의해 액세스되고 원하는 정보를 저장할 수 있는 다른 형태의 저장 매체, 또는 이들의 적합한 조합일 수 있다.

[0068] 통신 버스(18)는 프로세서(14), 컴퓨터 판독 가능 저장 매체(16)를 포함하여 컴퓨팅 장치(12)의 다른 다양한 컴포넌트들을 상호 연결한다.

[0069] 컴퓨팅 장치(12)는 또한 하나 이상의 입출력 장치(24)를 위한 인터페이스를 제공하는 하나 이상의 입출력 인터페이스(22) 및 하나 이상의 네트워크 통신 인터페이스(26)를 포함할 수 있다. 입출력 인터페이스(22) 및 네트워크 통신 인터페이스(26)는 통신 버스(18)에 연결된다. 입출력 장치(24)는 입출력 인터페이스(22)를 통해 컴퓨팅 장치(12)의 다른 컴포넌트들에 연결될 수 있다. 예시적인 입출력 장치(24)는 포인팅 장치(마우스 또는 트랙패드 등), 키보드, 터치 입력 장치(터치패드 또는 터치스크린 등), 음성 또는 소리 입력 장치, 다양한 종류의 센서 장치 및/또는 촬영 장치와 같은 입력 장치, 및/또는 디스플레이 장치, 프린터, 스피커 및/또는 네트워크 카드와 같은 출력 장치를 포함할 수 있다. 예시적인 입출력 장치(24)는 컴퓨팅 장치(12)를 구성하는 일 컴포넌트로서

컴퓨팅 장치(12)의 내부에 포함될 수도 있고, 컴퓨팅 장치(12)와는 구별되는 별개의 장치로 컴퓨팅 장치(12)와 연결될 수도 있다.

[0071] 이상에서 본 발명의 대표적인 실시예들을 상세하게 설명하였으나, 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자는 상술한 실시예에 대하여 본 발명의 범주에서 벗어나지 않는 한도 내에서 다양한 변형이 가능함을 이해할 것이다. 그러므로 본 발명의 권리범위는 설명된 실시예에 국한되어 정해져서는 안 되며, 후술하는 특허청구범위뿐만 아니라 이 특허청구범위와 균등한 것들에 의해 정해져야 한다.

부호의 설명

[0073] 100 : 포맷 스트링 취약점 검출 장치

102 : 후킹 모듈

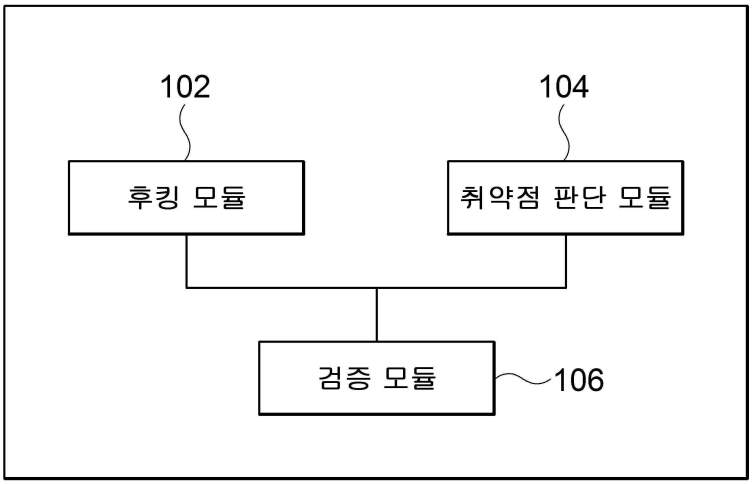
104 : 취약점 판단 모듈

106 : 검증 모듈

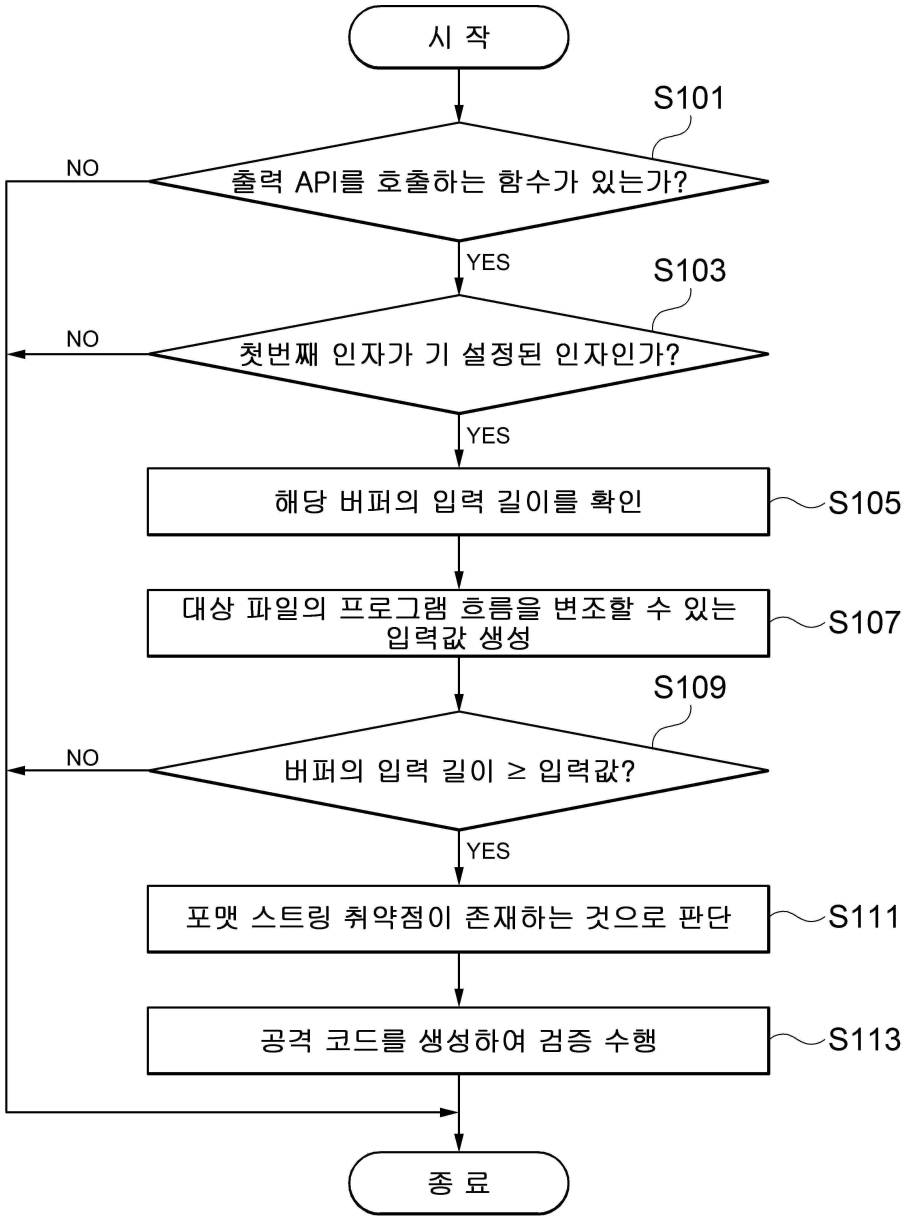
도면

도면1

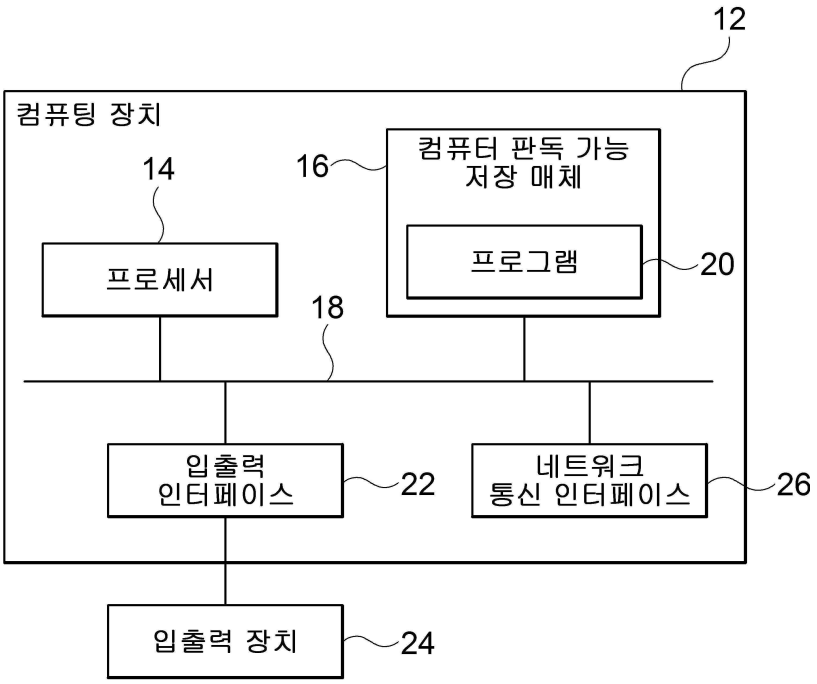
100



도면2


시 작
S101
출력 API를 호출하는 함수가 있는가?
NO
YES
S103
첫번째 인자가 기 설정된 인자인가?
NO
YES
해당 버퍼의 입력 길이를 확인
S105
대상 파일의 프로그램 흐름을 변조할 수 있는 입력값 생성
S107
S109
버퍼의 입력 길이 ≥ 입력값?
NO
YES
포맷 스트링 취약점이 존재하는 것으로 판단
S111
공격 코드를 생성하여 검증 수행
S113
종 료

도면3


10
12
컴퓨팅 장치
14
프로세서
16
컴퓨터 판독 가능
저장 매체
프로그램
20
18
입출력
인터페이스
22
네트워크
통신 인터페이스
26
입출력 장치
24