함수 단위 동적 커널 업데이트를 위한 보안 정책 및 기법의 설계

박현찬○ 유 혁
고려대학교 컴퓨터학과
hcpark@os.korea.ac.kr, hxy@os.korea.ac.kr

Design of safety policy and mechanism for dynamic kernel update with function-granularity

Hyunchan Park○ Chuck Yoo
Department of Computer Science and Engineering, Korea University

최근 시스템의 복잡도가 증가함에 따라 보안 취약점 문제가 더욱 많이 발생하고 있다. 이를 해결하기 위해 보안 패치가 배포되고 있지만, 시스템 서비스의 중단이 필요하고 패치 자체의 안정성이 검증되지 못해 패치의 적용이 늦어지는 문제가 발생한다. 우리는 이러한 문제를 해결하기 위해 업데이트성이 없는 커널을 위한 함수 단위 동적 업데이트 시스템인 DUNK를 설계하였다. DUNK는 서비스 중단 없는 업데이트를 가능케 하고, 보안 기법인 MAFIA를 이용해 안전한 업데이트를 수행한다. MAFIA는 바이너리 패치 코드의 접근 행위를 분석함으로써 패치된 함수가 기존 함수의 접근 권한을 상속받도록 하고, 이를 검증하는 기술을 제공한다. 본 논문에서는 DUNK의 설계와 MAFIA의 알고리즘 및 수행에 대해 기술한다.

1. 서론

시스템 규모가 점점 더 커짐에 따라 시스템의 설계 및 구현에서 보다 많은 에러가 발생하고, 이는 보안 취약점(security vulnerability)의 발생으로 이어진다. 최근 이러한 보안 취약점들은 매우 커다란 문제로 인식되고 있어 미국 국립기술 표준원에서 제공하는 NVD(National Vulnerability Database)와 같은 국가적인 보안 취약점 관리 사이트도 등장하였다. 이런 데이터 베이스는 보안 취약점에 대한 보고 외에도 수정을 위한 패치(patch) 정보도 함께 제공한다. 그러나 [1]에 따르면, 보고되고 패치가 제공된 보안 취약점을 시스템 관리자들이 실제로 수정하기까지는 오랜 시간이 걸리거나, 아예 수정되지 않는 것으로 나타났다. 이렇게 패치가 지연되는 이유는 첫째로 제공하던 서비스가 중단되는 것이 큰 손해를 가져오기 때문이고, 둘째로 패치의 안정성에 대한 시스템 관리자들의 불안감 때문이다.

본 논문에서는 이러한 문제의 해결을 위해 먼저 보안 패치를 위한 커널의 함수 단위 동적 업데이트 시스템인 DUNK(Dynamic Update in Non-updatable Kernel)를 제안한다. 커널의 동적인 업데이트는 시스템이 수행 중인 상태에서도 재컴파일이나 재시작없이 커널을 수정할 수 있는 기법이다. 따라서 서비스의 중단 없이도 보안 패치를 적용할 수 있다. 이러한 업데이트를 바이너리 명령어 단위가 아닌 함수 단위로 수행하는 것은 패치 개발자가 기존 커널의 개발 환경을 그대로 활용할 수 있게 해준다. 따라서 많은 커널 개발에 사용되는 C와 같은 고급 언어를 이용하여 보안 패치를 빠르게 개발할 수 있게 된다. 그리고 우리는 DUNK의 보안 기법으로 간접 접근 경로의 매칭(MAFIA: MAtching the Footprints of Indirect Accesses) 기법을 설계하였다. MAFIA 기법은 기존 격리 정책들이 메모리 주소 영역을 접근 가능 구역과 불가능 구역으로 나누어 관리하는 것에 반해 업데이트 코드의 접근 행위(access behavior)를 분석하여 기존 코드의 접근 행위와 동일한지 검사하는 기법이다. MAFIA를 활용하면 업데이트하려는 코드가 공유 데이터의 접근에 있어서 기존의 코드와 동일한 동작을 한다는 것을 보장할 수 있다. 따라서 업데이트에 의해 커널의 안정성이 저하되는 것을 최소화할 수 있다. 그리고 간접 접근에 대한 권한 검사를 코드가 커널에 삽입될 때 한 번만 수행하면 되므로 실행 시간에 발생하는 부가적인 검사를 제거할 수 있다.

2. 본론

우리는 1) 서비스 중단 없는 업데이트, 2) 시스템 안정성 저하 최소화를 위한 보안 정책 수립, 3) 동일한 개발 환

경의 제공의 세 가지 설계 목표를 갖고 DUNK를 설계하였다. DUNK의 업데이트는 함수 단위의 동적 업데이트를 사용하며, 기존 코드의 첫 부분을 업데이트 코드로 분기하는 명령어로 대치하는 가장 단순한 구현을 수행한다. DUNK를 위한 보안 패치는 로드 가능한 오브젝트 코드 형태를 취해야 한다. 이를 위한 개발 과정은 전혀 제한하지 않기 때문에 기존 커널 개발을 위해 사용하던 개발 환경을 그대로 활용할 수 있다. DUNK의 보안 목표는 기존 함수와 패치된 함수의 커널 전역 데이터에 대한 접근 행위(access behavior)를 분석하여 시스템 관리자가 채택한 정책에 따라 업데이트의 수행 여부를 판별할 수 있도록 하는 것이다. 여기서 접근 행위란 데이터에 대한 일련의 접근들(읽기, 쓰기, 수행)이 이루어지는 과정을 뜻한다. 접근 행위는 단순히 어떤 영역에 대한 접근만이 아닌, 여러 접근들의 순서 또한 포함한다. 이런 접근 행위를 분석하면 해당 코드가 커널에 대해 어떤 접근을 하는지 알 수 있고, 만약 두 코드의 접근 행위가 완벽하게 일치한다면, 커널에 미치는 영향도 일치한다고 판단할 수 있다.

DUNK의 보안 목표를 달성하기 위한 기법으로 우리는 MAFIA(MAtching the Footprints of Indirect Accesses)를 고안하였다. MAFIA는 접근 행동을 분석하여 접근 가능 리스트를 구하는 기법으로 DUNK의 보안 정책을 구현하고 적용하는 발판이 된다. MAFIA의 동작은 1) 오브젝트 코드 분석 기법, 2) 분석을 통한 접근 권한 작성, 그리고 3) 업데이트 코드를 분석하여 접근 권한과 일치하는지 검사하는 방식을 이용하여 순서대로 수행된다. 아래 그림 1과 표 1이 본 논문의 주요한 성과이며 이 알고리즘을 기존 코드와 업데이트 코드 양쪽에 수행하여 도출된 결과를 비교 분석하고 시스템 관리자가 양 결과의 차이점에 대한 승인을 함으로써 업데이트가 이루어진다.

```
Rd: Destination Register and its number
Rs: Source Register and its number
ACL: Access Control List

lists regs[ALL_REGS];          //linked string list
all regs[] = "INITnr;";        //write down to all register

for(each instructions) {
    if(Rd is updated)
        alloc new block to regs[Rd];

    Write expressions to block following the table 1.

    if(READ or WRITE occurs)
        specify the unrolled expression to ACL;

    Call reculsively itself when in-bound branch is taken
    At the end of code: alloc new block, write "END;" and
    terminate this function
}
```

그림 1. MAFIA의 수행 알고리즘

명령어 종류	표현 방법
데이터 처리 (읽기/쓰기)	*<레지스터/메모리에 대한 직접 접근>* 읽기: RD(where) 쓰기: WR(where) *<레지스터/메모리에 대한 간접 접근>* 읽기: RD[where±@] 쓰기: WR[where±@] * where = Rd:블록번호:필드번호
분기 명령어	*<코드 내부로의 분기>* 무조건 분기: BAW (Always) 분기시: BT (Taken) 비분기시: BNT (Not-Taken) *<코드 외부로의 분기>* 실행/복귀 등: BOUT * 모든 레지스터에 기록함
연산 명령	① 읽기나 쓰기가 포함된 경우, 이에 대한 표현 처리 수행 ② "명령어 이름 및 피연산자" 기록
스택 명령	POP: PP Rs PUSH: PS Rs * 모든 대상 레지스터에 대해 기록
기타 명령	"명령어 이름 및 피연산자" 기록

표 1.명령어에 따른 표현 방법

3. 결론

본 논문에서는 보안 패치로 인해 시스템 서비스의 중단이 발생하거나 패치로 인해 시스템 안정성이 저하되는 것을 우려하여 패치의 적용이 늦어지는 문제를 해결하기 위해 DUNK와 MAFIA를 제안하였다. DUNK의 설계와 이에 필요한 기술들, 그리고 여러 가지 고려사항들을 제시하였으며, MAFIA 기법의 알고리즘과 구체적인 실행 방식을 기술하였다. MAFIA 기법의 사용으로 기존 함수 코드의 커널 메모리 접근 권한을 업데이트 함수의 코드가 상속받을 수 있다. 또한 정적으로 분석 및 검증이 가능하기 때문에 성능 상의 저하도 발생하지 않는다. DUNK와 MAFIA의 사용으로 우리는 동적 커널 업데이트 기법을 현재 업데이트성이 지원되지 않는 커널에서 안전하게 수행할 수 있다. 현재 우리는 DUNK와 MAFIA를 ARM 머신에서 구현하고 있다. 추가로 MAFIA를 이용한 정책 설정과 실제 보고된 여러 보안 패치에 적용해보는 연구가 필요할 것이다.

참고문헌

[1] W.A. Arbaugh, et al., "Windows of vulnerability: a case study analysis," Computer, vol. 33, no. 12, 2000, pp. 52-59.