【서지사항】

【서류명】 특허출원서

【출원구분】 특허출원

【출원인】

【명칭】 에스케이하이닉스 주식회사

【특허고객번호】 1-1998-004569-8

【출원인】

【명칭】 서울대학교산학협력단

【특허고객번호】 1-2007-050924-2

【대리인】

【성명】 김선종

【대리인번호】 9-2003-000244-3

【포괄위임등록번호】 2011-100152-1

【발명의 국문명칭】 데이터 보호 기능을 수행하는 데이터 저장 장치 및 프로그램 컨텍스트를 이용하여 데이터 보호 기능을 지원하는 호스트 장치

【발명의 영문명칭】 DATA STORAGE DEVICE PERFORMING DATA PROTECTION AND HOST DEVICE SUPPORTING DATA PROTECTION USING PROGRAM CONTEXT

【발명자】

【성명】 이상구

【성명의 영문표기】 Sanggu LEE

【주민등록번호】 940226-0XXXXXX

【우편번호】 08840

【주소】 서울특별시 관악구 서림9길 27(신림동) 202

【발명자】

【성명】 김지홍

【성명의 영문표기】 Jihong KIM

【주민등록번호】 640105-0XXXXXX

【우편번호】 06993

【주소】 서울특별시 동작구 동작대로39길 22(동작동, 이수힐스테이트) 101-1106

【출원언어】 국어

【공지예외적용대상증명서류의 내용】

【공개형태】 학술단체 서면발표

【공개일자】 2022.06.27

【취지】 위와 같이 특허청장에게 제출합니다.

대리인 김선종 (서명 또는 인)

【수수료】

【출원료】 0 면 46,000 원

【가산출원료】 23 면 0 원

【우선권주장료】 0 건 0 원

【심사청구료】 0 항 0 원

【합계】 46,000 원

2022-08-23

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

1 : 공지예외적용대상(신규성상실의예외,_출원시의특례)규정을_적용받기_위한_증명서류

PDF 파일 첨부

【발명의 설명】

【발명의 명칭】

데이터 보호 기능을 수행하는 데이터 저장 장치 및 프로그램 컨텍스트를 이용하여 데이터 보호 기능을 지원하는 호스트 장치{DATA STORAGE DEVICE PERFORMING DATA PROTECTION AND HOST DEVICE SUPPORTING DATA PROTECTION USING PROGRAM CONTEXT}

【기술분야】

【0001】 본 기술은 랜섬 웨어와 같이 악의적인 프로그램에 의한 입출력 동작을 차단하여 데이터를 보호하는 데이터 저장 장치, 프로그램 컨텍스트를 이용하여 데이터 보호 기능을 지원하는 호스트 장치에 관한 것이다.

【발명의 배경이 되는 기술】

【0002】 랜섬 웨어와 같은 악성 프로그램은 사용자 몰래 파일을 암호화하고 이를 사용할 수 없게 만들어 큰 피해를 야기한다.

【0003】 이를 방지하기 위하여 랜섬 웨어를 사전적으로 탐지하여 이를 제거하는 방식을 채택할 수 있는데 새로운 랜섬 웨어가 지속적으로 등장하고 있어 사전적으로 모든 랜섬 웨어를 완벽하게 차단하는 것은 매우 어렵다.

【0004】 또한 종래의 윈도우 시스템에서는 화이트리스트 방식을 도입하여 보호 기능이 설정된 폴더나 파일에 대해서는 허락된 응용 프로그램에 한하여 접근을 허용하기도 한다.

【0005】그러나 응용 프로그램 레벨에서 화이트리스트 방식을 사용하더라도 허용된 응용 프로그램 자체에 랜섬 웨어 코드가 주입되는 경우에는 악성 코드로 인한 피해를 방지할 수 없다.

【선행기술문헌】

【특허문헌】

【0006】**(특허문헌 0001)** US 7,421,570 B2

【비특허문헌】

【0007】**(비특허문헌 0001)** Chris Gniady, Ali R. Butt, and Y. Charlie Hu. 2004. Program-counter-based pattern classification in buffer caching. In Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation - Volume 6 (OSDI'04). USENIX Association, USA, 27.

【발명의 내용】

【해결하고자 하는 과제】

【0008】본 기술은 응용 프로그램 레벨보다 세분화된 입출력(I/O) 동작 레벨에서 화이트 리스트 방식을 적용하여 데이터를 보호하는 데이터 저장 장치, 데이터 보호 기능을 지원하는 호스트 장치를 제공한다.

【과제의 해결 수단】

【0009】본 발명의 일 실시예에 의한 데이터 저장 장치는 입출력 동작을 식별하는 프로그램 컨텍스트를 저장하는 화이트 리스트를 포함하는 메모리 장치; 및 호스트로부터 쓰기 요청과 함께 전송된 프로그램 컨텍스트를 화이트 리스트와 비교하여 쓰기 요청의 허용 여부를 판단하는 화이트 리스트 관리 회로를 포함하는 제어 회로를 포함한다.

【0010】본 발명의 일 실시예에 의한 호스트 장치는 운영체제에 의해 동작하는 프로세서를 포함하는 호스트 장치로서, 프로세서는 입출력 동작에 대응하는 프로그램 컨텍스트를 저장하는 프로그램 컨텍스트 레지스터; 및 상기 입출력 동작에 대응하는 사용자 함수 또는 시스템 쓰기 함수가 호출되는 경우 상기 사용자 함수의 프로그램 카운터 값 또는 상기 시스템 쓰기 함수의 프로그램 카운터 값을 이용하여 상기 프로그램 컨텍스트를 갱신하는 프로그램 컨텍스트 제어 회로를 포함한다.

【발명의 효과】

【0011】본 실시예에서는 입출력 동작 레벨에서 화이트리스트를 적용하므로 응용프로그램이 랜섬 웨어와 같은 악성코드를 포함하더라도 데이터 보호 동작이 가능하다.

【0012】본 실시예에서는 호스트에 포함된 프로세서에서 프로그램 컨텍스트 값을 계신하며 계산한 값을 커널 모드 레지스터에 저장함으로써 커널 권한이 없는 응용 프로그램을 통해 변조하기 어렵다.

【0013】또한 쓰기 함수가 호출되는 경우 커널이 저장된 프로그램 컨텍스트 값을 읽어 데이터 저장 장치에 전송하므로 커널 권한이 없는 응용 프로그램을 통해 변조하기 어렵다.

【0014】본 기술은 데이터 저장 장치 내부에 화이트 리스트를 저장하므로 랜섬 웨어와 같은 악성 코드가 화이트 리스트에 접근하는 것이 어렵다.

【도면의 간단한 설명】

【0015】도 1은 입출력 동작과 프로그램 컨텍스트의 관계를 나타내는 설명도.

도 2는 프로그램 컨텍스트 계산 동작을 설명하는 설명도.

도 3은 본 발명의 일 실시예에 의한 컴퓨팅 시스템을 나타내는 블록도.

도 4는 본 발명의 일 실시예에 의한 프로그램 컨텍스트 맵핑 회로의 데이터 구조를 나타내는 도표.

도 5는 본 발명의 효과를 나타내는 그래프.

【발명을 실시하기 위한 구체적인 내용】

【0016】이하에서는 첨부한 도면을 참조하여 본 발명의 실시예를 개시한다.

【0017】본 실시예에서 프로그램 컨텍스트(prc)는 시스템 입출력 함수의 호출에 이르기까지의 실행 경로를 인코딩한 정보로서, 프로그램 컨텍스트는 입출력 동작마다 서로 다른 값을 가지며 이에 따라 프로그램 컨텍스트를 비교함으로써 입출력 동작의 동일 여부를 판별할 수 있다.

【0018】 이하에서 입출력 동작은 데이터 쓰기가 수행되는 동작을 지칭하나 이에 한정되는 것은 아니다.

【0019】 본 실시예에서는 데이터 저장 장치의 데이터를 변조하는 것을 차단하기 위한 것이므로 시스템 입출력 함수는 시스템 쓰기 함수이다.

【0020】 본 실시예에서 프로그램 컨텍스트는 시스템 쓰기 함수가 호출되기까지 호출되는 함수들의 프로그램 카운터(PC) 값을 누적하여 계산한다.

【0021】 본 실시예에서 호스트에서 시스템 쓰기 함수가 호출되는 경우 프로그램 컨텍스트를 데이터 저장 장치에 전송한다. 본 실시예에서는 데이터 저장 장치는 SSD이다.

【0022】 본 실시예에서 데이터 저장 장치는 허용된 입출력 동작을 나타내는 프로그램 컨텍스트를 포함하는 화이트 리스트를 내부에 저장한다.

【0023】 데이터 저장 장치는 전송된 프로그램 컨텍스트가 화이트 리스트에 저장되었는지 판단하고, 프로그램 컨텍스트가 화이트리스트에 포함된 경우 입출력 동작을 허용하고 그렇지 않은 경우 입출력 동작을 허용하지 않는다.

【0024】 도 1은 입출력 동작과 프로그램 컨텍스트의 관계를 나타내는 설명도이다.

【0025】 도 1은 응용 프로그램인 데이터베이스 프로그램의 수행 중 사용될 수 있는 두 가지 입출력 동작인 갱신 동작과 로그 동작을 예시한다.

【0026】데이터베이스 프로그램의 동작 중 수행되는 갱신 동작과 관련된 함수의 호출은 다음과 같은 순서로 진행된다.

【0027】main() -> db_main() -> db_update() -> sys_write()

【0028】sys_write()는 커널 소프트웨어에서 제공하는 시스템 쓰기 함수이며 나머지는 데이터베이스 프로그램에서 제공하는 사용자 함수이다.

【0029】갱신 동작 시 호출되는 함수들의 프로그램 카운터 값은 순차적으로, 8, 20, 40, 60이며 이들의 합인 프로그램 컨텍스트는 128이다. 즉 갱신 동작에 대응하는 프로그램 컨텍스트는 128이다.

【0030】데이터베이스 프로그램의 동작 중 로그 동작과 관련된 함수의 호출은 다음과 같은 순서로 진행된다.

【0031】main() -> logger() -> sys_write()

【0032】로그 동작 시 각 함수에 대응하는 프로그램 카운터 값은 순차적으로, 84, 100이며 이들의 합인 프로그램 컨텍스트는 184이다. 즉 로그 동작에 대응하는 프로그램 컨텍스트는 184이다.

【0033】두 입출력 동작이 미리 허용되는 경우 대응하는 프로그램 컨텍스트 128, 184는 데이터 저장 장치 내부의 화이트 리스트에 저장된다.

【0034】시스템 쓰기 함수인 sys_write()가 실행되면 대응하는 프로그램 컨텍스트가 데이터 저장 장치에 전달되어 화이트 리스트와 비교된다.

【0035】도 1에서 랜섬 웨어 코드는 랜섬 웨어 프로그램 또는 다른 응용 프로그램에 주입된 코드이다.

【0036】랜섬 웨어 코드에 의해 수행되는 변조 동작과 관련하여 encrypt_files(), crypto_write(), sys_write()가 순서대로 호출되며 이들에 대응하는 프로그램 컨텍스트는 164, 176, 212이며, 이에 따라 변조 동작에 대응하는 프로그램 컨텍스트는 452이다.

【0037】변조 동작 시 시스템 쓰기 함수가 호출되는 경우 프로그램 컨텍스트 452가 데이터 저장 장치에 전송되는데 이때 해당 프로그램 컨텍스트가 화이트 리스트에 포함되어 있지 않으면 쓰기 동작이 실패하며 이에 따라 변조 동작에 대하여 데이터 보호가 가능하다.

【0038】도 2는 프로그램 컨텍스트를 계산하는 방법을 설명한 도면이다.

【0039】(A)는 종래의 기술로서 프레임 포인터를 사용하여 프로그램 컨텍스트를 계산하는 방법이다.

【0040】콜 스택을 관리하기 위하여 프레임 포인터를 사용하는 경우라면 시스템 쓰기 함수가 호출되는 경우 콜 스택을 추적하여 프로그램 컨텍스트를 계산할 수 있다.

【0041】예를 들어 전술한 갱신 동작에서 함수 main(), db_main(), db_update()의 프로그램 카운터는 상대적인 리턴 주소를 이용하여 알 수 있다.

【0042】시스템 쓰기 함수에 대응하는 프로그램 카운터는 프로세서의 특수 레지스터를 통해 알 수 있는데 RISC-V ISA에서는 epc 레지스터 값을 읽어 프로그램 카운터를 알 수 있다.

【0043】다만 이러한 종래의 기술은 소프트웨어 컴파일 과정에서 프레임 포인터를 사용하지 않도록 설정되는 경우에는 프로그램 컨텍스트 값을 생성할 수 없다.

【0044】(B)는 본 실시예에 의한 프로그램 컨텍스트 계산 방법을 나타낸다.

【0045】본 실시예에서는 호스트의 프로세서 내부에서 프로그램 컨텍스트를 직접 관리한다.

【0046】이를 위해 본 발명의 일 실시예에 의한 프로세서는 프로그램 컨텍스트를 저장하는 프로그램 컨텍스트 레지스터(prc)와 프로그램 컨텍스트 관리 회로를 포함한다.

【0047】종래의 프로세서는 커널 모드에서 관리할 수 있는 특수 레지스터를 포함한다.

【0048】본 실시예에서는 특수 레지스터에 설정된 여유 공간을 프로그램 컨텍스트 레지스터로 사용하나 다른 실시예에서는 별도의 레지스터를 추가하여 프로그램 컨텍스트 레지스터로 사용할 수도 있다.

【0049】프로그램 컨텍스트는 입출력 동작 과정에서 수행되는 함수 호출 시 갱신된다.

【0050】도 2(B)는 도 1의 갱신 동작 시 프로그램 컨텍스트 계산 방법을 나타낸다.

【0051】프로세서는 호출된 함수에 대응하는 프로그램 카운트를 알고 있으므로 프로세서 내부의 프로그램 컨텍스트 관리 회로는 호출된 함수에 대응하는 프로그램 카운터 값을 이용하여 프로그램 컨텍스트를 갱신한다.

【0052】프로그램 컨텍스트 관리 회로는 예를 들어 main() 함수가 호출되면 이에 대응하는 프로그램 카운터 값 8을 프로그램 컨텍스트 레지스터(prc)에 누적한다.

【0053】마찬가지로 db_main(), db_update()에 대응하는 프로그램 카운터 값 20, 40이 순차적으로 누적되고, sys_write() 함수가 호출되면 대응하는 프로그램 카운터 값 60이 누적된다. 이에 따라 프로그램 컨텍스트는 128이 된다.

【0054】도 2(B)에서 call은 main(), db_main(), db_update()와 같은 사용자 함수를 호출하는 인스트럭션이고, ecall은 sys_write()와 같은 시스템 쓰기 함수를 호출하는 인스트럭션이다.

【0055】 함수 동작 마지막에 리턴 인스트럭션이 실행되는 경우 프로그램 카운터 값은 프로그램 컨텍스트 값에서 제외된다.

【0056】예를 들어 sys_write() 함수에서 리턴 인스트럭션(sret)이 실행되면 대응하는 프로그램 카운터 값 68이 프로그램 컨텍스트에서 제외되어 프로그램 컨텍스트는 68이 된다.

【0057】 전술한 바와 같이 본 실시예에서 프로그램 컨텍스트 레지스터는 커널 모드에서 접근 가능한 특수 레지스터의 일부에 저장된다.

【0058】 시스템 쓰기 함수가 실행되는 경우 시스템 호출 처리 인스트럭션이 실행되며 시스템 호출 처리 인스트럭션이 실행되는 경우 프로세서 내부의 특수 레지스터를 읽는 동작이 가능하며 이는 종래에도 잘 알려진 기술이다.

【0059】 본 실시예에서는 시스템 쓰기 함수인 sys_write()가 호출되면 시스템 호출 처리 인스트럭션을 통해 현재 프로그램 컨텍스트 레지스터의 값을 읽어 프로세서 외부로 출력할 수 있다.

【0060】 이와 같이 프로그램 컨텍스트 레지스터는 커널 모드의 함수에 의해 접근이 가능하고 사용자 모드의 함수에 의해서는 접근이 되지 않으므로 악성 코드에 의한 변조가 불가능하다.

【0061】 도 3은 본 발명의 일 실시예에 의한 컴퓨팅 시스템(1)을 나타내는 블록도이다.

【0062】 본 실시예의 컴퓨팅 시스템(1)은 데이터 저장 장치(100), 호스트(200) 및 인터페이스 회로(300)를 포함한다.

【0063】 본 실시예에서 데이터 저장 장치(100)는 SSD에 대응하며 기본적인 입출력 동작에 필요한 구성은 본 발명의 개시에 반드시 필요한 것이 아니므로 이들에 대한 설명은 생략한다.

【0064】 데이터 저장 장치(100)는 제어 회로(110), 메모리 장치(120) 및 인터페이스 제어 회로(130)를 포함한다.

【0065】 제어 회로(110)는 주소 맵핑, 웨어 레벨링, 가비지 콜렉션 등 종래의 FTL(Flash Translation Layer)에 대응한다.

【0066】 제어 회로(110)는 하드웨어, 소프트웨어, 또는 이들의 조합으로 구현될 수 있으며 특정한 형태로 제한되지 않는다.

【0067】 본 실시예에서 제어 회로(110)는 화이트 리스트를 관리하는 화이트 리스트 관리 회로(111)를 더 포함한다.

【0068】 메모리 장치(120)는 저장 공간의 일부에 화이트 리스트(121)를 저장한다.

【0069】 화이트 리스트(121)는 데이터 저장 장치(100)에 대해서 허용되는 입출력 동작에 대응하는 프로그램 컨텍스트를 저장한다.

【0070】 화이트 리스트(121)에 포함되는 프로그램 컨텍스트는 미리 결정되어 저장되는 것으로 가정한다.

【0071】 예를 들어 리버스 엔지니어링을 통해 시스템에서 발생하는 함수에 대한 호출 및 리턴 정보를 추출하고 이들로부터 데이터 저장 장치(100)에 대한 쓰기 동작과 관련된 정보를 추출하여 화이트 리스트에 추가할 프로그램 컨텍스트를 결정할 수 있다.

【0072】 다른 방법으로 데이터 저장 장치(100)에서 화이트 리스트(121)에 포함되지 않은 프로그램 컨텍스트가 발견되는 경우 이를 호스트(200)에 전달할 수 있다.

【0073】 컴퓨터 시스템(1)의 관리자로 하여금 호스트(200)에 전달된 프로그램 컨텍스트를 분석하여 정상적인 입출력 동작에 관련된 것이면 이를 화이트 리스트(121)에 추가함으로써 화이트 리스트(121)에 누락된 프로그램 컨텍스트를 줄이고 이를 통해 시스템의 정상적인 동작을 보장할 수 있다.

【0074】 이외에도 화이트 리스트(121)에 저장할 프로그램 컨텍스트를 결정하는 방법은 다양하게 존재할 수 있으며 본 발명은 특정한 방법을 사용하여 화이트 리스트(121)를 생성할 것을 제한하지 않는다.

【0075】 화이트 리스트 관리 회로(111)는 호스트(200)로부터 생성된 쓰기 요청과 이와 함께 전달되는 프로그램 컨텍스트 리스트(prcs)를 화이트 리스트(121)와 비교한다.

【0076】 도 3에서 nvme_write() 함수에서 id는 쓰기 요청의 식별자이고, LBA는 쓰기 주소를 나타내고, data는 데이터를 표시한다. 또한 send_prc() 함수에서 id는 쓰기 요청의 식별자이고, prcs는 전달할 프로그램 컨텍스트 리스트를 나타낸다.

【0077】 화이트 리스트 관리 회로(111)는 전달된 프로그램 컨텍스트 리스트(prcs)에 포함된 프로그램 컨텍스트를 화이트 리스트(121)와 비교한다.

【0078】본 실시예에서 프로그램 컨텍스트 리스트에 포함된 프로그램 컨텍스트 중 적어도 하나가 화이트 리스트(121)에 포함되지 않은 경우 화이트 리스트 관리 회로(111)는 실패를 표시한다.

【0079】이에 따라 제어 회로(110)는 쓰기 요청에 대하여 관련 프로그램 컨텍스트 정보를 포함하는 에러 메시지를 생성하여 호스트(200)로 전달할 수 있다.

【0080】호스트(200)는 프로세서(210), 캐시 메모리(220), 프로그램 컨텍스트 맵핑 회로(230), 호스트 인터페이스 제어 회로(240)를 포함한다.

【0081】운영 체제(10)는 호스트(200)의 전반적인 동작을 제어한다. 운영 체제(10)는 커널, 파일 시스템 등의 소프트웨어를 포함하고, 운영 체제(10)가 탑재된 컴퓨터 시스템(1)의 동작 자체는 종래의 기술로서 이하에서는 종래의 기술과 차별되는 구성을 중심으로 발명을 개시한다.

【0082】프로세서(210)는 CPU에 대응하는 구성으로서 종래의 CPU에서 수행하는 기본적인 동작을 공통적으로 수행하므로 이에 대한 구체적인 개시는 생략한다.

【0083】본 실시예에 의한 프로세서(210)는 프로그램 컨텍스트 레지스터(211)와 프로그램 컨텍스트 제어 회로(212)를 포함한다.

【0084】프로그램 컨텍스트 레지스터(211)는 현재 실행되고 있는 입출력 동작에 대응하는 프로그램 컨텍스트를 저장하며, 다수의 입출력 동작이 병렬적으로 수행되는 경우 다수의 입력력 동작에 대응하는 다수의 프로그램 컨텍스트를 저장한다.

【0085】 프로그램 컨텍스트 제어 회로(212)는 도 2(B)와 같이 함수 호출 인스트럭션(call, ecall)이 실행되는 경우 프로그램 컨텍스트 레지스터(211)에 호출된 함수에 대응하는 프로그램 카운터 값을 더하여 프로그램 컨텍스트를 갱신한다.

【0086】 전술한 바와 같이 시스템 쓰기 함수를 호출하는 인스트럭션(ecall)이 실행되는 경우 커널 모드에서 시스템 호출 처리 함수가 자동으로 호출된다.

【0087】 커널 모드에서 시스템 호출 처리 함수가 실행되는 경우 프로그램 컨텍스트 레지스터(211)에서 프로그램 컨텍스트를 읽어서 외부로 출력할 수 있다.

【0088】 캐시 메모리(220)는 종래의 캐시 메모리에 대응한다.

【0089】 쓰기 데이터를 캐싱하는 동작은 종래의 기술과 유사하다.

【0090】 즉, 본 실시예에서 쓰기 요청된 데이터는 우선 캐시 메모리(220)에 저장되며 플러시 동작에 의해 캐시 메모리(220)에서 선택된 데이터가 데이터 저장 장치(300)에 쓰여진다.

【0091】 다만, 전술한 바와 같이 본 실시예에서는 호스트(200)에서 데이터 저장 장치(100)에 쓰기 요청을 전송할 때 쓰기 요청과 함께 관련된 프로그램 컨텍스트 리스트를 함께 전송하는 점에서 종래의기술과 차이가 있다.

【0092】 이와 같이 호스트(200)에서 쓰기 명령이 처리되는 시간과 실제로 데이터 저장 장치(100)에 데이터가 쓰이는 시간에는 차이가 발생하므로 프로그램 컨텍스트 리스트를 임시로 저장할 필요가 있다.

【0093】 프로그램 컨텍스트 맵핑 회로(230)는 캐시 메모리(220)에 저장된 데이터와 이에 대응하는 프로그램 컨텍스트를 저장한다.

【0094】 도 4는 프로그램 컨텍스트 맵핑 회로(230)의 데이터 구조를 나타낸다.

【0095】 본 실시예에서 프로그램 컨텍스트 맵핑 회로(230)는 데이터의 논리 주소 즉 페이지 번호와 이에 대응하는 프로그램 컨텍스트 리스트를 저장한다.

【0096】 예를 들어 페이지 번호 100은 프로그램 컨텍스트 27과 연관되고, 페이지 번호 500는 두 개의 프로그램 컨텍스트 79, 99와 연관된다.

【0097】 이와 같이 하나의 페이지 번호에 대해서 다수의 입출력 동작이 수행되는 경우 하나의 페이지 번호에 대해서 다수의 프로그램 컨텍스트가 리스트 형태로 저장될 수 있다.

【0098】 캐시 메모리(220)의 데이터에 대해서 플러시 동작이 진행되는 경우 캐시 메모리(220)와 함께 프로그램 컨텍스트 맵핑 회로(230)를 참조하여 쓰기 요청과 함께 대응하는 프로그램 컨텍스트 리스트를 데이터 저장 장치(100)로 전송한다.

【0099】 전술한 바와 같이 프로그램 컨텍스트 리스트에 포함된 모든 프로그램 컨텍스트가 화이트 리스트(121)에 포함되는 경우 데이터 저장 장치(100)에서 쓰기 동작이 정상적으로 수행된다.

【0100】 프로그램 컨텍스트 리스트에 포함된 적어도 하나의 프로그램 컨텍스트가 화이트 리스트(121)에 포함되지 않은 경우 데이터 저장 장치(100)에서 쓰기

동작을 수행하지 않고 호스트(200)에 실패를 통지한다.

【0101】 호스트(200)는 페일이 통지되는 경우 관리자가 검토할 수 있도록 페일과 관련된 프로그램 컨텍스트에 대해서 로깅 동작을 수행할 수 있다.

【0102】 종래와 같이 호스트(200)에서 동작하는 운영체제(10)는 파일 시스템을 포함하며 데이터 저장 장치(100)에 저장되는 파일의 메타 데이터를 관리한다.

【0103】 전술한 바와 같이 본 실시예에서는 쓰기 요청의 실패 여부를 플러시 단계에서 알 수 있고 호스트(100)에서 쓰기 데이터를 캐싱하는 단계에서는 미리 알 수 없다.

【0104】 이에 따라 본 실시예에서 호스트(200)는 파일에 대한 메타 데이터의 일관성을 보장하기 위하여 추가적인 동작을 수행한다.

【0105】 호스트(200)는 파일에 대한 쓰기 요청이 제공되는 경우 대응하는 파일에 대응하는 기존 메타 데이터를 백업한 후 파일에 대한 메타 데이터를 갱신한다.

【0106】 메타 데이터의 백업을 위해 호스트(200)에서 사용하는 주소 공간 중 일부를 할당할 수 있다.

【0107】 파일 데이터가 플러시 되는 과정에서 데이터 저장 장치(100)에서 해당 파일에 대한 쓰기 요청이 실패하는 경우 백업된 메타 데이터를 이용하여 복구할 수 있다.

【0108】본 실시예에서는 입출력 동작 레벨의 화이트 리스트를 이용하여 데이터 보호 기능을 수행한다.

【0109】이에 따라 본 실시예에 의한 데이터 보호 기능을 전혀 수행하지 않는 경우와 비교하여 데이터 입출력 성능은 다소 저하될 수 있다.

【0110】도 5는 본 발명의 효과를 설명하는 그래프이다.

【0111】도 5에서 가로 축은 벤치마크의 종류이며 세로 축은 종래 기술을 기준으로 정규화된 초당 입출력 횟수(IOPS)를 나타낸 것이다. 각 벤치마크에서 읽기와 쓰기의 비율은 2:8이다.

【0112】본 실시예 1은 화이트리스트에 1K개의 엔트리가 저장되는 경우이고, 본 실시예 2는 10K개의 엔트리가 저장되는 경우이고, 본 실시예 3은 100K개의 엔트리가 저장되는 경우이다.

【0113】도시된 바와 같이 본 실시예의 경우 데이터 보호 기능을 추가함으로 인하여 종래의 기술에 비하여 IOPS의 성능은 저하되었는데, 평균적으로 실시예 1의 경우 1.9%의 IOPS 저하가 발생하고 실시예 3의 경우 3.7%의 IOPS 저하가 발생하여 성능 저하는 미미한 수준이다.

【0114】이와 같이 본 기술에 의한 컴퓨팅 시스템은 입출력 성능 저하를 최소화하면서 데이터 보호 기능을 수행할 수 있다.

【0115】본 발명의 권리범위는 이상의 개시로 한정되는 것은 아니다. 본 발명의 권리범위는 청구범위에 문언적으로 기재된 범위와 그 균등범위를 기준으로 해

석되어야 한다.

【부호의 설명】

【0116】1: 컴퓨팅 시스템

100: 데이터 저장 장치

110: 제어 회로

111: 화이트 리스트 관리 회로

120: 메모리 장치

121: 화이트 리스트

130: 인터페이스 제어 회로

200: 호스트

210: 프로세서

211: 프로그램 컨텍스트 레지스터

212: 프로그램 컨텍스트 제어 회로

220: 캐시 메모리

230: 프로그램 컨텍스트 맵핑 회로

10: 운영 체제

300: 인터페이스 회로

【청구범위】

【청구항 1】

입출력 동작을 식별하는 프로그램 컨텍스트를 저장하는 화이트 리스트를 포함하는 메모리 장치; 및

호스트로부터 쓰기 요청과 함께 전송된 프로그램 컨텍스트를 상기 화이트 리스트와 비교하여 상기 쓰기 요청의 허용 여부를 판단하는 화이트 리스트 관리 회로를 포함하는 제어 회로

를 포함하는 데이터 저장 장치.

【청구항 2】

청구항 1에 있어서, 상기 쓰기 요청과 함께 다수의 프로그램 컨텍스트가 전송된 경우, 상기 화이트 리스트 관리 회로는 상기 다수의 프로그램 컨텍스트 중 적어도 하나가 상기 화이트 리스트에 포함되지 않는 경우 상기 쓰기 요청에 대한 실패를 표시하는 데이터 저장 장치.

【청구항 3】

청구항 1에 있어서, 상기 제어 회로는 상기 쓰기 요청이 실패로 판단되는 경우 상기 쓰기 요청에 대응하는 프로그램 컨텍스트를 포함하는 쓰기 요청 실패 정보를 출력하는 데이터 저장 장치.

【청구항 4】

운영체제에 의해 동작하는 프로세서를 포함하는 호스트 장치로서, 상기 프로

세서는

입출력 동작에 대응하는 프로그램 컨텍스트를 저장하는 프로그램 컨텍스트 레지스터; 및

상기 입출력 동작에 대응하는 사용자 함수 또는 시스템 쓰기 함수가 호출되는 경우 상기 사용자 함수의 프로그램 카운터 값 또는 상기 시스템 쓰기 함수의 프로그램 카운터 값을 이용하여 상기 프로그램 컨텍스트를 갱신하는 프로그램 컨텍스트 제어 회로

를 포함하는 호스트 장치.

【청구항 5】

청구항 4에 있어서, 상기 프로세서는 상기 시스템 쓰기 함수가 호출되는 경우 상기 프로그램 컨텍스트 레지스터에 누적된 프로그램 카운터의 값을 상기 입출력 동작에 대응하는 프로그램 컨텍스트로 출력하는 호스트 장치.

【청구항 6】

청구항 4에 있어서, 상기 시스템 쓰기 함수에 의한 쓰기 데이터를 임시 저장하는 캐시 메모리; 및

상기 쓰기 데이터의 주소와 상기 입출력 동작에 대응하는 프로그램 컨텍스트를 연관하여 저장하는 프로그램 컨텍스트 맵핑 회로

를 더 포함하는 호스트 장치.

【청구항 7】

청구항 6에 있어서, 상기 캐시 메모리에서 선택된 데이터를 데이터 저장 장치로 플러시 하는 경우 상기 선택된 데이터에 대한 쓰기 요청과 상기 선택된 데이터에 대한 프로그램 컨텍스트를 상기 데이터 저장 장치로 전송하는 호스트 장치.

【청구항 8】

청구항 7에 있어서, 상기 쓰기 데이터를 캐시 메모리에 저장하는 경우 상기 쓰기 데이터에 대응하는 메타 데이터를 백업하고, 상기 데이터 저장 장치로부터 페일을 통지받는 경우 백업된 메타 데이터를 복구하는 호스트 장치.

【요약서】

【요약】

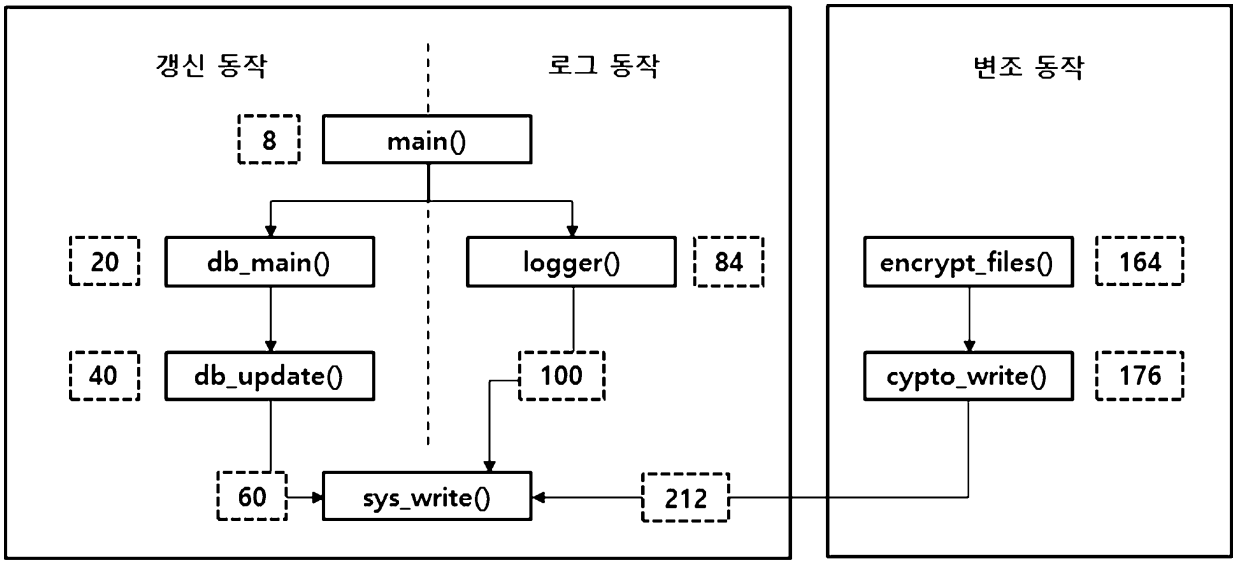
본 기술에 의한 데이터 저장 장치는 입출력 동작을 식별하는 프로그램 컨텍스트를 저장하는 화이트 리스트를 포함하는 메모리 장치; 및 호스트로부터 쓰기 요청과 함께 전송된 프로그램 컨텍스트를 화이트 리스트와 비교하여 쓰기 요청의 허용 여부를 판단하는 화이트 리스트 관리 회로를 포함하는 제어 회로를 포함한다.

【대표도】

도 3

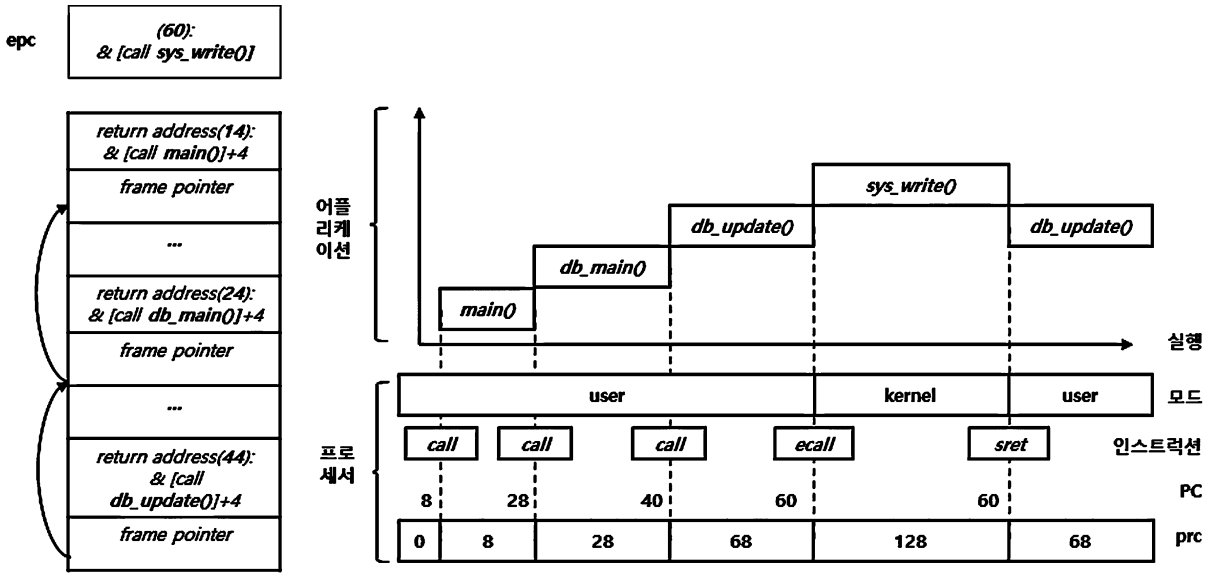
【도면】

【도 1】



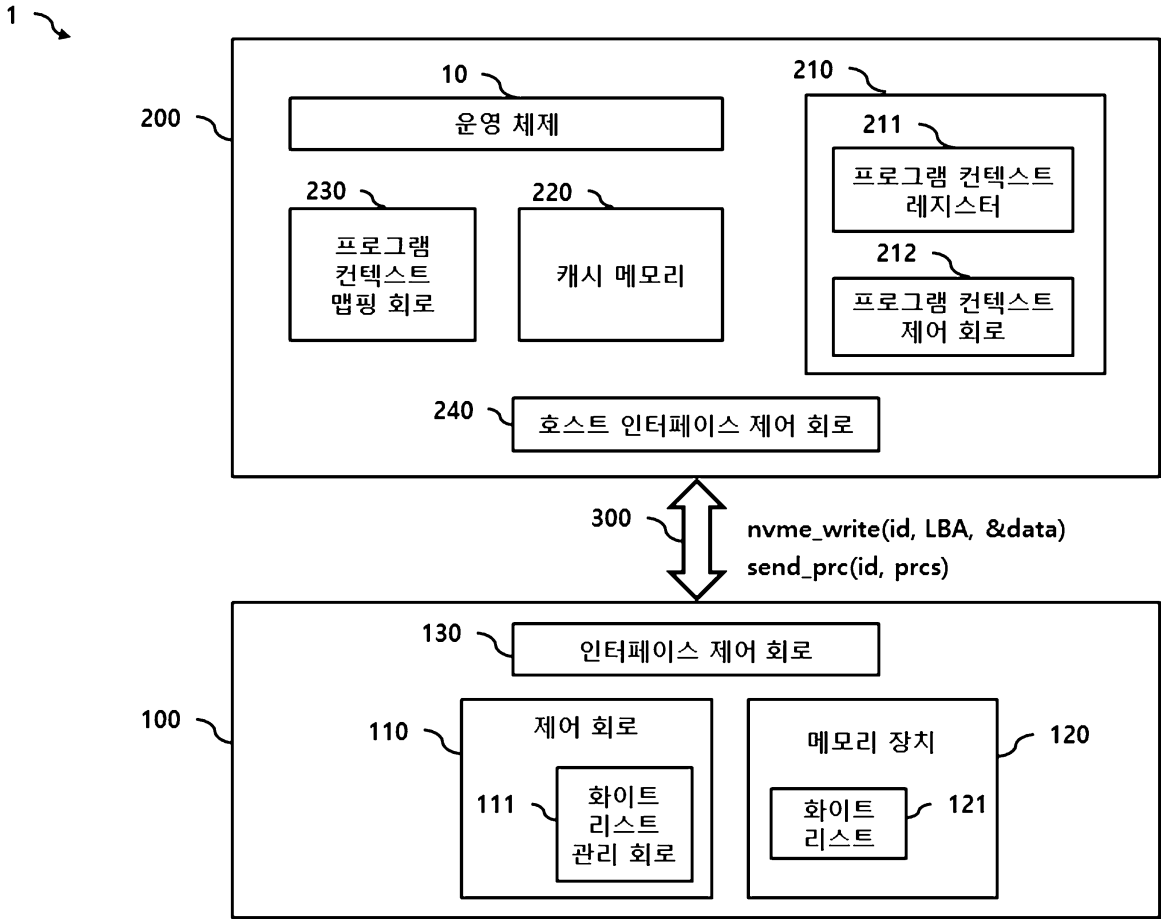
<데이터베이스 프로그램>　　<랜섬웨어 코드>

【도 2】



(A) 종래의 기술　　(B) 본 실시예

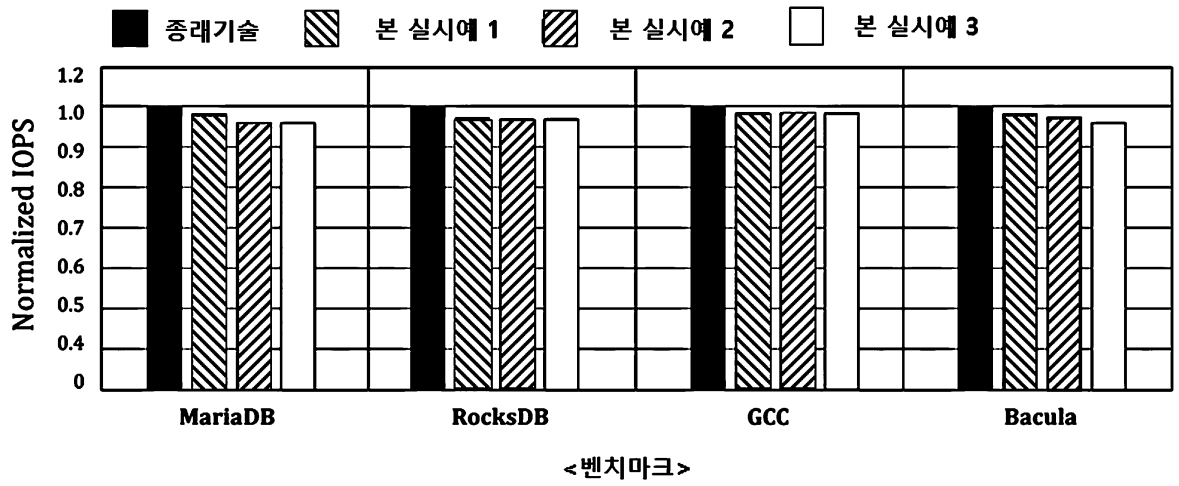
【도 3】



【도 4】

논리 주소 (페이지 번호)	프로그램 컨텍스트 리스트 (prcs)
100	27
500	79, 99

【도 5】


종래기술
본 실시예 1
본 실시예 2
본 실시예 3
Normalized IOPS
1.2
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
MariaDB
RocksDB
GCC
Bacula
<벤치마크>