(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2012년04월20일
(11) 등록번호 10-1137157
(24) 등록일자 2012년04월09일

(51) 국제특허분류(Int. Cl.)
G06F 9/445 (2006.01) *G06F 9/30* (2006.01)
(21) 출원번호 10-2005-0031862
(22) 출원일자 2005년04월18일
심사청구일자 2010년04월08일
(65) 공개번호 10-2006-0045811
(43) 공개일자 2006년05월17일
(30) 우선권주장
10/880,848 2004년06월30일 미국(US)
60/570,124 2004년05월11일 미국(US)
(56) 선행기술조사문헌
JP2002055839 A*
KR1020010041771 A*
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
마이크로소프트 코포레이션
미국 워싱턴주 (우편번호 : 98052) 레드몬드 원 마이크로소프트 웨이
(72) 발명자
블럼필드, 안쏘니
미국 98052 워싱턴주 레드몬드 원 마이크로소프트 웨이마이크로소프트 코포레이션 내
골란, 길라드
미국 98052 워싱턴주 레드몬드 원 마이크로소프트 웨이마이크로소프트 코포레이션 내
(뒷면에 계속)
(74) 대리인
제일특허법인

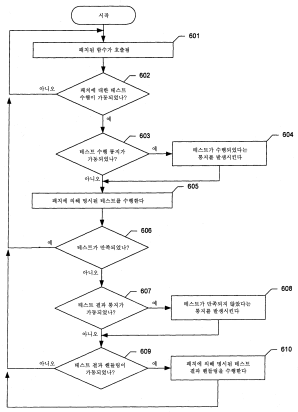
전체 청구항 수 : 총 12 항 심사관 : 지정훈

(54) 발명의 명칭 **효과적 패칭**

(57) 요 약

소프트웨어 패치를 적용하기 위한 도구가 기술된다. 자동 패칭 에이전트를 사용해서, 도구는 소프트웨어 패치를 수신한다. 소프트웨어 패치 수신에 응답해서, 사용자 중재 없이, 도구는 이하의 액션들을 실행한다: 첫번째로, 도구는 현재 로드되어 있고 수신된 소프트웨어 패치가 관련된 실행가능 모듈의 인스턴스를 식별한다. 두번째로, 도구는 식별된 실행가능 모듈 인스턴스의 동작을 수정하기 위해 식별되고 로드된 실행가능 모듈 인스턴스에 수신된 소프트웨어 패치를 적용한다.

대 표 도 - 도6



(72) 발명자

감스, 제이슨

미국 98052 워싱턴주 레드몬드 원 마이크로소프트 웨이마이크로소프트 코포레이션 내

알시바니, 사우드 에이.

미국 98052 워싱턴주 레드몬드 원 마이크로소프트 웨이마이크로소프트 코포레이션 내

필드, 스코트 에이.

미국 98052 워싱턴주 레드몬드 원 마이크로소프트 웨이마이크로소프트 코포레이션 내

특허청구의 범위

청구항 1

프로세서, 비휘발성 메모리 및 휘발성 메모리를 구비한 컴퓨팅 시스템에서, 소프트웨어 패치를 실행 가능 모듈의 다수의 인스턴스(instance)에 적용하기 위한 방법으로서,

자동화된 패칭 에이전트를 사용해서,

상기 소프트웨어 패치를 수신하는 단계;

상기 소프트웨어 패치를 수신하는 것에 응답하여, 사용자 중재 없이,

상기 컴퓨팅 시스템에 의해, 수신된 소프트웨어 패치와 관련된 실행 가능 모듈의 제1 인스턴스의 상기 비휘발성 메모리로부터 상기 휘발성 메모리로의 제1 로딩을 검출하는 단계;

상기 제1 로딩의 검출에 응답하여, 상기 컴퓨팅 시스템에 의해, 상기 수신된 소프트웨어 패치를 상기 휘발성 메모리 내의 상기 실행 가능 모듈의 제1 인스턴스에 적용하여, 상기 실행 가능 모듈의 제1 인스턴스의 일부를 수정함으로써, 상기 실행 가능 모듈의 제1 인스턴스의 동작(behavior)을 수정하는 단계;

상기 컴퓨팅 시스템에 의해, 상기 실행 가능 모듈의 제2 인스턴스의 상기 비휘발성 메모리로부터 상기 휘발성 메모리로의 제2 로딩을 검출하는 단계; 및

상기 제2 로딩의 검출에 응답하여, 상기 컴퓨팅 시스템에 의해, 상기 수신된 소프트웨어 패치를 상기 휘발성 메모리 내의 상기 실행 가능 모듈의 제2 인스턴스에 적용하여, 상기 실행 가능 모듈의 제2 인스턴스의 일부를 수정함으로써, 상기 실행 가능 모듈의 제2 인스턴스의 동작을 수정하는 단계

를 포함하는 방법.

청구항 2

제1항에 있어서,

동일한 실행가능 모듈과 관련된 2개의 소프트웨어 패치가 수신되고, 수신된 소프트웨어 패치들 둘 다는 상기 실행가능 모듈의 제1 인스턴스의 동작을 수정하기 위해 적용되는 방법.

청구항 3

제1항에 있어서,

상기 수신된 소프트웨어 패치가 적용된 후에, 상기 실행가능 모듈의 제1 인스턴스는 수정된 동작을 행하는 방법.

청구항 4

삭제

청구항 5

제1항에 있어서,

상기 소프트웨어 패치와 관련된 상기 실행가능 모듈의 제1 인스턴스는 상기 소프트웨어 패치와 관련된 복수 개의 실행가능 모듈 인스턴스 중에서 식별되고, 상기 수신된 소프트웨어 패치는 상기 수신된 소프트웨어 패치에 포함된 상기 실행가능 모듈의 제1 인스턴스에 대한 모듈 인스턴스 고유 정보(module instance-specific information)를 사용하여 적용되는 방법.

청구항 6

삭제

청구항 7

삭제

청구항 8

제1항에 있어서,

상기 수신된 소프트웨어 패치는 상기 실행가능 모듈의 제1 인스턴스의 명시된 부분에 테스트 및 테스트 결과 핸들링 로직을 추가하기 위해 상기 실행가능 모듈의 제1 인스턴스의 동작을 수정하는 것을 명시하는 방법.

청구항 9

제1항에 있어서,

상기 실행가능 모듈의 제1 인스턴스의 수정된 상기 동작이 패치 구성 인터페이스를 사용해서 가동(enabled) 또는 중지(disabled)될 수 있는 방법.

청구항 10

제1항에 있어서,

상기 소프트웨어 패치가 미리 정해진 위치에 복사되도록 허용함으로써 상기 소프트웨어 패치가 수신되고, 상기 검출 및 상기 적용은 상기 소프트웨어 패치가 상기 미리 정해진 위치로 복사되는 것에 응답하여 자동으로 수행되는 방법.

청구항 11

비휘발성 메모리 및 휘발성 메모리를 구비한 컴퓨팅 시스템으로 하여금 소프트웨어 패치를 현재 로딩된 실행가능 모듈 인스턴스에 적용하기 위한 방법을 수행하게 하는 콘텐츠를 갖는 컴퓨터-판독가능 기록 매체로서,

상기 방법은,

자동화된 패칭 에이전트를 사용하여,

상기 소프트웨어 패치를 수신하는 단계;

상기 소프트웨어 패치를 수신하는 것에 응답하여,

상기 컴퓨팅 시스템에 의해, 수신된 소프트웨어 패치와 관련된 실행 가능 모듈의 제1 인스턴스의 상기 비휘발성 메모리로부터 상기 휘발성 메모리로의 제1 로딩을 검출하는 단계;

상기 제1 로딩의 검출에 응답하여, 상기 컴퓨팅 시스템에 의해, 상기 수신된 소프트웨어 패치를 상기 휘발성 메모리 내의 상기 실행 가능 모듈의 제1 인스턴스에 적용하여, 상기 실행 가능 모듈의 제1 인스턴스의 일부를 수정함으로써, 상기 실행 가능 모듈의 제1 인스턴스의 동작(behavior)을 수정하는 단계;

상기 컴퓨팅 시스템에 의해, 상기 실행 가능 모듈의 제2 인스턴스의 상기 비휘발성 메모리로부터 상기 휘발성 메모리로의 제2 로딩을 검출하는 단계; 및

상기 제2 로딩의 검출에 응답하여, 상기 컴퓨팅 시스템에 의해, 상기 수신된 소프트웨어 패치를 상기 휘발성 메모리 내의 상기 실행 가능 모듈의 제2 인스턴스에 적용하여, 상기 실행 가능 모듈의 제2 인스턴스의 일부를 수정함으로써, 상기 실행 가능 모듈의 제2 인스턴스의 동작을 수정하는 단계

를 포함하는 컴퓨터-판독가능 기록 매체.

청구항 12

소프트웨어 패치를 현재 로딩된 실행가능 모듈 인스턴스에 적용하기 위한 컴퓨팅 시스템으로서,

상기 소프트웨어 패치를 자동으로 수신하는 네트워크 인터페이스;

비휘발성 메모리;

휘발성 메모리; 및

프로세서를 포함하고,

상기 프로세서는

상기 네트워크 인터페이스에 의한 상기 소프트웨어 패치의 수신에 응답하여, 사용자 중재 없이, 수신된 소프트웨어 패치와 관련된 실행 가능 모듈의 인스턴스의 상기 비휘발성 메모리로부터 상기 휘발성 메모리로의 로딩을 검출하는 검출 서브시스템; 및

상기 검출 서브시스템이 상기 수신된 소프트웨어 패치와 관련된 상기 실행 가능 모듈의 인스턴스의 상기 로딩을 검출하는 것에 응답하여, 사용자 중재 없이, 상기 수신된 소프트웨어 패치를 상기 휘발성 메모리 내의 상기 실행 가능 모듈의 인스턴스에 적용하여, 상기 실행 가능 모듈의 인스턴스의 일부를 수정함으로써, 상기 실행가능 모듈의 인스턴스의 동작을 수정하는 패치 적용 서브시스템

을 포함하는 컴퓨팅 시스템.

청구항 13

소프트웨어 패치를 나타내는 소프트웨어 패치 데이타 구조를 포함하는 컴퓨터-판독가능 기록 매체로서,

상기 소프트웨어 패치 데이타 구조는,

상기 소프트웨어 패치를 적용할 하나 이상의 실행가능 소프트웨어 모듈들을 식별하는 제1 정보; 및

식별된 실행가능 소프트웨어 모듈들이 수정되는 방식을 명시하는 제2 정보를 포함하며,

상기 소프트웨어 패치 데이타 구조의 콘텐츠는, 상기 제1 정보에 의해 식별된 실행가능 소프트웨어 모듈의 컴퓨팅 시스템의 비휘발성 메모리로부터 상기 컴퓨팅 시스템의 휘발성 메모리로의 로딩을 검출하고, 상기 로딩의 검출에 응답하여, 상기 제2 정보에 의해 명시된 방식으로 상기 휘발성 메모리 내의 상기 실행 가능 소프트웨어 모듈의 일부를 수정하기 위해, 자동 패칭 에이전트에 의해 사용될 수 있는 컴퓨터-판독가능 기록 매체.

청구항 14

제13항에 있어서,

상기 소프트웨어 패치 데이타 구조는 서명을 더 포함하고,

상기 서명은, 상기 소프트웨어 패치 데이타 구조가 명시된 패치 권한(authority)에 의해 생성되었고, 상기 서명이 생성된 후에는 수정되지 않았음을 나타내는 컴퓨터-판독가능 기록 매체.

청구항 15

제14항에 있어서,

서명인 신원(identity)이 상기 소프트웨어 패치 데이타 구조의 상기 서명으로부터 구별될 수 있고, 구별된 상기 서명인 신원은 상기 식별된 실행가능 소프트웨어 모듈의 서명으로부터 구별될 수 있는 서명인 신원과 비교될 수 있는 컴퓨터-판독가능 기록 매체.

청구항 16

삭제

청구항 17

삭제

청구항 18

삭제

청구항 19

삭제

청구항 20

삭제

청구항 21

삭제

청구항 22

삭제

청구항 23

삭제

청구항 24

삭제

청구항 25

삭제

청구항 26

삭제

청구항 27

삭제

청구항 28

삭제

청구항 29

삭제

청구항 30

삭제

청구항 31

삭제

청구항 32

삭제

청구항 33

삭제

청구항 34

삭제

청구항 35

삭제

청구항 36

삭제

청구항 37

삭제

청구항 38

삭제

청구항 39

삭제

명 세 서

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

[0015] 본 발명은 설치된 컴퓨터 프로그램들의 동작을 업데이트하는 필드에 관한 것이다.

[0016] 패칭(patching)은 응용 프로그램, 유틸리티 프로그램, 운영 시스템과 운영 시스템 컴포넌트, 디바이스 드라이버 등을 포함하는 이미 설치된 프로그램들을 수정하는 프로세스이다. 패칭은 프로그램 오류 교정, 보안 위험의 감소나 제거, 또는 수정된 프로그램에 의해 사용되는 논리의 개선을 포함하는 다양한 목적들을 위한 프로그램들을 수정하기 위해 유용할 수 있다. 패칭은 통상적으로 패치되는 프로그램을 원래 공급하는 회사나 기타 조직에 의해 시작된다.

[0017] 설치된 프로그램들은 실행가능 코드 모듈로 거의 구성된다. 예를 들어, 와싱톤주 레드몬드시의 마이크로소프트사의 WINDOW XP 운영 시스템 상에서 실행하도록 고안된 다수의 프로그램들은 거의 "DLL"이라고 불리는 실행가능 코드 모듈들로 구성된다. 패칭에 대한 한 가지 인기있는 종래 접근법은, 패치되는 설치된 프로그램을 만드는 실행가능 코드 모듈들 중에, 패치로 수정하기를 원하는 프로그램 코드를 포함하는 실행가능 코드 모듈을 식별하여, 원하는 수정이 만들어지는 식별된 실행가능 코드 모듈의 새 버젼을 생성하고, 그 패치를 적용하기를 원하는 사용자들에게, 설치기 프로그램과 함께, 그 식별된 실행가능 코드 모듈의 새 버젼을 배포한다. 그 다음, 각각의 사용자는 그/그녀가 그 패치를 적용하기를 원하는지를 결정하고, 만약 그렇다면, 설치기 프로그램을 실행시켜서 식별된 실행가능 코드 모듈의 원래 버젼을 식별된 실행가능 코드 모듈의 새 버젼으로 대체한다.

발명이 이루고자 하는 기술적 과제

[0018] 패칭에 대한 종래 접근법들은 다수의 중요한 단점들을 갖는다. 이들 단점들은 종종 패치들을 수신하고 적용하는 것과 연관된 부담을 증가시킨다. 일부 경우들에서, 이 증가된 부담은 일부 사용자들에 의한 일부 패치들의 적용을 지연시키고, 일부 사용자들에 의한 일부 패치들의 적용을 막기도 한다. 그런 패치들의 적용의 지연 및 방지는 일부 경우들에서, 특히 보안 위험을 감소시키거나 제거하기 위해 고안된 패치들에 대해, 사용자들에게 심각한 부정적 결과들을 가져올 수 있다.

[0019] 패칭에 대한 종래 접근법들의 한 가지 단점은 복수 패치들이 한 개의 프로그램에 한 가지 수정을 하기 위해 생성되어 배포되어야 하는 일반 경우들과 관련있다. 일부 경우들에서, 패치되는 프로그램은 프로그램이 실행되도록 고안된 각각의 운영 시스템 또는 운영 시스템 버젼 및/또는 그 프로그램의 각각의 자연 언어 버젼을 위한 상이한 특질과 같은 특정 실행가능 코드 모듈의 여러 개의 상이한 "특질들(flavors)"을 갖는다. 식별된 실행가능 코드 모듈이 그런 실행가능 코드 모듈인 경우, 상술된 패치 생성 및 배포 프로세스는 식별된 실행가능 코드 모듈의 각각의 특질에 대해 반복되어야 한다. 그 다음, 사용자는 식별된 실행가능 코드 모듈의 적합한 특질을 위한 패치를 선택하고 적용해야 한다. 결과적인 많은 수의 패치들을 정렬하고 각각의 사용자의 컴퓨터 시스템에 적용할 적합한 패치들의 세트를 선택하는 것은 매우 부담스러울 수 있어서, 이 조건은 때때로 "패치 지옥(patch hell)"이라고 불리운다. 일부 경우들에서, 관리자는 각각의 타겟 시스템을 위한 적합한 종래 패치들을 선택하기 위해 사용되는 각각의 타겟 시스템에 설치되는 실행가능 모듈 버젼들의 세트를 식별하는 목록

데이터베이스를 관리해야 한다.

[0020] 패칭에 대한 종래 접근법들의 다른 단점은 배포되는 패치들의 큰 크기와 관련이 있다. 실행가능 코드 모듈들이 수 메가바이트의 크기를 갖는 것은 드문 경우가 아니며, 이것은 반면 한 개의 패치가 필적할만한 크기를 갖도록 하여, 일부 사용자들에게는 그것을 배포하고 저장하는 것을 힘들게하고, 또는 배포와 저장을 불가능하도록 하기도 한다. 이 문제는 복수 개의 특질들을 갖는 패치들에게 배가될 수 있다. 더욱이, 각각의 종래의 패치는 통상적으로 전체 대체(substitute) 실행가능 모듈을 포함하므로, 종래 패치들의 적용은 코드 쳔(code churn)의 문제를 일으킬 수 있다.

[0021] 패칭에 대한 종래 접근법들의 추가 단점은 생산 컴퓨터 시스템들에 그들을 적용하기 전에 일부 사용자들이 패치들을 테스트할 필요성과 관련이 있다. 일부 경우들에서, 컴퓨터 시스템에 패치의 설치는, 패치에 포함된 식별된 실행가능 코드 모듈의 새 버젼이 새 프로그래밍 오류를 발생시키는 경우, 또는 그것이 적용된 컴퓨터 시스템 상에서 실행하는 다른 프로그램과의 새롭고 예측이 안된 인터랙션의 원인이 되는 경우와 같은, 부정적 결과들을 가져올 수 있다. 따라서, 종종 데이터와 동작이 유지되는 것이 중요한 생산 시스템에 대해 패치를 적용하기 전에, 사용자는 우선 테스팅 시스템에 패치를 적용하여 패치가 생산 시스템에 적용하기에 안전한지를 평가한다. 그런 패치들의 개별 테스팅은 패칭에 연관된 부담을 증대시킨다. 추가로, 종래 패치가 패치가 적용된 후 충분한 이후에, 적용 호환성 문제 또는 새 작업의 약점과 같은 문제를 생성하는 경우, 패치로 다시 그런 문제들을 추적하는 것은 어려울 수 있다.

[0022] 패칭에 대한 종래 접근법들의 추가 단점은 패치에 포함된 설치기의 동작과 관련된다. 종종, 실행 프로그램의 일부인 실행가능 코드 모듈을 대체하기 위해, 설치기는 우선 그 프로그램의 실행을 종료해야 한다. 또한, 일부 경우들에서, 그런 대체는 컴퓨터 프로그램을 재시작하지 않고 완료될 수 없다. 이들 단계들 모두는 패치된 컴퓨터 시스템의 사용에서 크게 와해되는 원인이 될 수 있다.

[0023] 패칭에 대한 종래 접근법들의 다른 단점은 또한 "핫 픽스(hot fix)"라고도 불리우는 "사적인 픽스(private fix)"가 실행가능 모듈에 대해 고객들의 적절한 일부 세트에게 이전에 발생된 그 실행가능 모듈을 패치하려는 시도와 관련이 있다. 일부 경우들에서, 사용자가 핫 픽스를 적용했는지에 종속하여 각각의 사용자에 따라 실행가능 코드 모듈의 상이한 새 버젼들을 대체하는 종래 패치들의 배포에서 직면되는 어려움 때문에, 사용자가 핫 픽스를 적용했는지에 상관없이 실행가능 모듈의 한 개의 새 버젼을 대체하는 간단한 종래 패치를 대신 배포하는 것은 통상적이다. 그 새 버젼이 핫 픽스를 구현하면, 패치는 그것을 수신할 의도가 없는 고객들에게 핫 픽스를 부과한다. 반면, 그 새 버젼이 핫 픽스를 구현하지 않으면, 그것은 핫 픽스를 수신할 의도인 고객들을 제거한다.

[0024] 패칭에 대한 종래 접근법들의 다른 단점은, 특정 동적 링크 라이브러리와 같은, 특정 실행가능 모듈에 의존하는 소프트웨어 제품들을 위한 설치기는 종종 타겟 컴퓨터 시스템의 파일 시스템의 비표준 위치에 그것을 저장하여 그 실행가능 모듈을 "숨긴다"는 사실과 관련이 있다. 따라서, 특정 타겟 시스템이 패치되는 실행가능 모듈의 사본을 포함하는지, 및, 만약 그렇다면, 타겟 컴퓨터 시스템의 파일 시스템의 어디에 그것이 저장되는지를 결정하는 것은 때때로 어렵거나 불가능하다. 또한, 일부 소프트웨어 제품들은 그들의 설치기들에 의해 설치되는 실행가능 모듈 버젼들의 "카탈로그"를 유지한다. 소프트웨어 제품은 특정 실행가능 모듈의 버젼의 카탈로그의 지시의 정확성에 의존할 것이다. 그런 의존성은 종래 패치가 카탈로그를 업데이트시키지 않고 카탈로그에서 식별된 실행가능 모듈의 버젼을 실행가능 모듈의 새 버젼으로 대체하는 경우 붕괴된다.

[0025] 패칭에 대한 종래 접근법들의 다른 단점은 그들이 패치되는 실행가능 모듈이 타겟 컴퓨터 시스템에 설치되기 전의 한 시점에 적용하기가 불가능하다는 사실로부터 생겨난다. 결과적으로, 패치되는 실행가능 모듈이 실행 모듈에 대한 종래 패치가 수신된 후에 타겟 컴퓨터 시스템에 설치되면, 패치가 실행가능 모듈에 적용될 것이라는 것은 거의 불가능하다.

[0026] 패칭에 대한 종래 접근법들의 다른 단점은 그들은 통상적으로 자유로운 수정 권한을 갖는 관리 계좌를 사용하여 타겟 컴퓨터 시스템에 로그인된 사용자에 의해서만 적용될 수 있다는 점이다. 이 목적에서 관리 계좌로의 로그인은 타겟 컴퓨터 시스템의 양태들을 수정하려고 탐색하고 그를 위해 자유로운 사용권한들을 요구하는 타겟 컴퓨터 시스템에 존재하는 바이러스들에게 타겟 컴퓨터 시스템이 공격받기 쉽게 만들 수 있다.

[0027] 패칭에 대한 종래 접근법들의 다른 단점은 종래 패치들은 실행가능 모듈의 대체를 역전하는 것, 또는 시스템 등록부로의 한 번 이상의 수정을 역전하는 것과 같은 단계들을 중지하는 것이 어렵거나 불가능할 수 있다는 점이다.

[0028] 따라서, 상술된 패칭에 대한 종래 접근법들의 단점들의 일부나 전부를 극복하는 패칭에 대한 새 접근법은 매우 유용할 것이다.

발명의 구성 및 작용

[0029] 설치된 컴퓨터 프로그램 코드를 패칭하는 소프트웨어 도구("도구(facility)")가 제공된다. 일부 실시예들에서, 도구는 설치된 함수들에 파라미터 테스팅과 테스트 결과 핸들링을 추가한다. 다른 실시예들에서, 일부 경우들에서 설치된 함수들의 실행 흐름의 임의의 위치에서, 도구는 설치된 함수들에 다양한 다른 종류의 기능을 추가한다.

[0030] 일부 실시예들에서, 각각의 패치에 대해, 도구는 패치되는 각각의 컴퓨터 시스템에게 -즉, 각각의 "타겟 컴퓨터 시스템"- 테스트를 수행하는 지점의 명시, 수행할 테스트의 신원(identity), 및 한 개 이상의 상이한 테스트 결과들에 응답하여 동작하는 방법을 배포한다. 일부 실시예들에서, 도구는 패치들에서 그 사용이 명시될 수 있는 파라미터 유효화와 다른 테스트들의 표준 세트를 제공한다. 예를 들어, 패치는 특정 함수에 대해, 그 함수의 특정 파라미터가 특정 값을 갖지 않으면, 그 함수의 호출은 그것의 실질적 실행이 시작되기 전에 실패해야 함을 명시할 것이다. 다른 패치는, 특정 함수에 대해, 특정 파라미터가 명시된 최대 길이를 초과하는 길이를 가지면, 그 파라미터는 그 함수의 실행이 진행되도록 승인되기 전에 명시된 최대 길이로 절단되어야 함을 명시할 것이다. 다수의 보안 작업들은 파라미터 값들이 원래 버젼의 함수 코드에서 블럭킹되지 않으면서, 그 함수가 안전하지 않은 조건을 생성하거나 이용하도록 하는 파라미터 값들로 함수들이 호출되도록 하는 것에 의존한다. 많은 경우들에서, 그런 작업들은 그런 파라미터 값들로 함수가 실행되는 것을 막기 위해 그런 패치들을 사용하여 방지될 수 있다. 일부 실시예들에서, 패치들은 파일로부터 읽혀지거나 사용자에 의해 입력된 값들과 같은 함수 파라미터들과는 다른 값들의 테스팅을 명시한다.

[0031] 일부 실시예들에서, 자동 패칭 에이젼트는 자동으로 각각의 패치를 수신하고, 그것을 유효화하고, 가능한 적용을 위해 패치 표에 그것을 저장한다. 일부 실시예들에서, 각각의 패치는 패치가 수신되었을 때 타겟 컴퓨터 시스템에 이미 로드된 패치될 실행가능 모듈의 임의의 인스턴스들에 적용된다. 이 접근법은 본 명세서에서 "핫 패칭(hot patching)"으로서 언급되고, 패치들이 수신시에 즉시 효과적이 되도록 하고, 패치될 실행가능 모듈이 디스크의 어디에 저장되는지를 도구가 결정할 수 있도록 요구하지 않는다. 일부 실시예들에서, 각각의 수신된 패치는 패치되는 실행가능 모듈의 디스크 이미지에 적용되어서, 디스크 이미지가 미래의 시점에서 로드될 때, 그 로드된 디스크 이미지는 패치를 포함한다. 이 접근법은 본 명세서에서 "콜드 패칭(cold patching)"으로서 언급되고, 복수의 세션들에서 패치들이 지속적이도록 허용한다. 일부 실시예들에서, 도구는 핫 및 콜드 패칭 모두를 수행한다. 일부 실시예들에서, 각각의 패치는 패치되는 실행가능 모듈이 운영 시스템의 로더에 의해 로드되는 각각의 시간에 패치되는 실행 모듈에 적용된다. 이 접근법은 본 명세서에서 "로드-시간 패칭(load-time patching)"이라고 불리운다. 일부 실시예들에서, 각각의 패치는 패치되는 함수가 호출되는 각각의 시간에 패치되는 실행가능 모듈에 적용된다. 이 접근법은 본 명세서에서 "호출-인터셉션 패칭(call-interception patching)"이라고 불리운다. 로드-시간 패칭과 호출-인터셉션 패칭 모두는 (1) 패치되는 실행가능 모듈이 디스크의 어디에 저장되는지를 도구가 결정할 수 있도록 요구하지 않고, (2) 특정 패치의 준비완료된 역전성(ready reversibility)을 용이하게 하고, (3) 실행가능 모듈의 디스크 이미지의 수정을 요구하지 않는다.

[0032] 일부 실시예들에서, 도구는 사용자나 관리자가 적용된 패치들의 동작을 구성하도록 허용한다. 예로써, 적용된 특정 패치에 대해, 그런 구성은 다음을 포함할 수 있다: 실행이 패치를 위해 명시된 지점에 도달할 때 패치에 의해 명시된 테스트가 수행되는지, 패치에 의해 명시된 테스트 결과 핸들링(result handling)이 수행되거나 또는 무시되는지, 및/또는 테스트의 수행 및/또는 그것의 결과가 로그(log)되고 경고 메시지에 디스플레이되는지 등. 이들 실시예들에서, 도구는 초기에 로깅(logging)을 가동하고 결과 핸들링을 중지하여 생산 컴퓨터 시스템들에서 패치들이 테스트되도록 한다. 이들 실시예들에서, 패치가 적용 호환성 문제나 기타 IT 문제와 같은 문제를 생성하는 인스턴스들을 식별하는 것을 돕기 위해 그것의 결과 핸들링을 가동한 후에 "버보스 모드(verbose mode)"에 패치 동작이 로그인되도록 추가 허용한다. 이들 실시예들은 또한, 패치가 문제를 일으키고 있는 것이 발견되면, 패치가 적용된 후에 신속하게 중지되도록 한다. 도구의 일부 실시예들은 또한 타겟 컴퓨터 시스템에서 수신되고 저장되는 패치들의 세트로부터 패치를 단순히 삭제하여 패치가 신속하게 중지되도록 한다.

[0033] 일부 실시예들에서, 도구는 테스트를 수행하는 지점, 수행할 테스트의 신원, 및 한 개 이상의 다른 테스트 결과들에 응답하여 동작하는 방법을 명시하는 소형 인간-판독가능 텍스트나 XML 도큐먼트와 같은, 코드가 아닌, 데이터에 더 가까운 것을 패치가 포함하는 "데이터-구동(data-driven)" 패칭 접근법을 사용한다. 일부 실시예들

에서, 패칭 에이젼트는 데이터-구동 패치들을 수신하고, 패치들에 의해 명시된 테스트들과 테스트 핸들링을 추가한다. 일부 실시예들에서, 도구는 "코드-구동(code-driven)" 패칭 접근법을 사용하고, 여기서 각각의 패치는 그 자체가 도구의 표준 파라미터 테스팅 함수들을 호출하여 테스트를 수행하고 테스트 핸들링을 수행하는 패치되는 실행가능 모듈에 추가되는 소형 프로그램을 포함한다. 데이터-구동 패칭 또는 코드-구동 패칭을 사용하여, 때때로 한 개의 패치로 패치되는 실행가능 모듈의 모든 특질들을 해결하는 것은 가능하다.

[0034] 일부 실시예들에서, 도구는 각각의 패치를 사인하여 (1) 패치가 승인된 소스로부터의 것이고, (2) 패치 내용은 승인된 소스에 의해 패치가 생성된 시간 이후에 수정되지 않았음을 모두 나타낸다.

[0035] 일부 실시예들에서, 도구는 각각의 패치를 모든 타겟 컴퓨터 시스템에 배포하여, 타겟 컴퓨터 시스템의 특성들에 기초하여, 타겟 컴퓨터 시스템 상의 패칭 에이젼트는 어떤 패치들이 타겟 컴퓨터 시스템에 적용되는지, 및 어떻게 그들이 적용되는지를 자동으로 결정한다. 이것은 패치들을 선택하고 적용하는 것과 종래 연관된 부담들, 및 정확한 현재의 목록 데이터베이스를 유지하는 많은 부담으로부터 사용자와 관리자를 해제한다. 예를 들어, 이들 특성들은 패치되는 실행가능 모듈들 중에 어떤 버젼이 타겟 컴퓨터 시스템에 설치되는지를 포함할 수 있다. 이들 실시예들에서, 특정 실행가능 모듈의 핫-핏스(hot-fixed)와 언핫-픽스(unhot-fixed) 특질들에 대한 다른 핸들링을 명시하는 패치들을 배포하고, 특정 실행가능 모듈을 위해 핫-픽스를 희생하거나 또는 그 실행가능 모듈을 패칭할 때 핫-픽스를 편재하도록 하는 임의의 필요성을 제거하여, 핫-픽스들에 의해 통상적으로 기인되는 유형의 문제들을 도구가 극복할 수 있다.

[0036] 일부 실시예들에서, 패칭 에이젼트는, 패치가 수신될 때 그 특정 패치에 의해 패치되는 실행가능 모듈이 타겟 시스템에 설치되는지에 상관없이, 타겟 시스템에서 수신된 각각의 패치를 저장한다. 많은 경우들에서, 패치되는 실행가능 모듈의 로딩 또는 패치되는 함수의 호출에 응답하여, 도구가 패치들을 적용하기 때문에, 도구는 패치가 타겟 시스템에서 수신된 후에 타겟 시스템에 설치되는 실행가능 모듈에 패치를 적용할 수 있다. 또한, 패치들을 패치되는 실행가능 모듈의 설치제거 및 후속적 재설치 후에도 잔존할 수 있다.

[0037] 일부 실시예들에서, 패칭 에이젼트는 운영 시스템 서비스에서 구현된다. 이들 실시예들에서, 도구는 패치를 적용하기 위해 요구되는 임의의 승인들에 대해 패칭 에이젼트를 따른다. 이들 실시예들은, 사용자가 넓은 수정 권한을 갖는 관리 계좌를 사용하여 타겟 컴퓨터 시스템에 로그인하여 타겟 컴퓨터 시스템에 존재하는 임의의 바이러스들에게 타겟 컴퓨터 시스템의 민감한 양태들을 수정하도록 하는 더 많은 기회를 제공하도록 하는 임의의 요구를 제거하여, 종래 패치들이 적용될 때 통상적으로 부가되는 보안 위험을 감소시킨다.

[0038] 도구에 의해 사용되는 이들 패치들은 통상적으로 비교적 소형이고, 그러므로 전송과 저장을 위해 간소한 자원 요구사항들을 부과한다. 또한, 도구에 의해 사용되는 패치들은 소수의 잘 정의된 방식들로 패치된 소프트웨어의 동작을 수정하는 경향이 있으므로, 도구는 코드 쳔(code churn)의 문제를 감소시키는 것을 돕는다.

[0039] 도 1은 도구가 구현될 수 있는 적합한 컴퓨팅 시스템 환경(100)의 예를 도시한다. 컴퓨팅 시스템 환경(100)은 적합한 컴퓨팅 환경의 일 예일 뿐이고, 도구의 사용이나 기능의 범위에 대해 임의의 제한을 제안하려고 의도되지 않았다. 컴퓨팅 환경(100)은 운영 환경(100)의 예에서 도시되는 컴포넌트들 중의 임의의 것이나 조합에 관련된 임의의 종속성이나 요구사항을 갖는 것으로 해석되어서도 안된다.

[0040] 도구는 다수의 다른 일반 목적 및 특수 목적 컴퓨팅 시스템 환경들이나 구성들과 동작할 수 있다. 도구와 사용하기에 적합할 수 있는 잘 공지된 컴퓨팅 시스템, 환경, 및/또는 구성의 예들은 다음을 포함하지만 이에 제한되지는 않는다: 개인용 컴퓨터, 서버 컴퓨터, 핸드헬드나 랩톱 디바이스, 태블릿 디바이스, 멀티프로세서 시스템, 마이크로프로세서-기반 시스템, 셋톱 박스, 프로그램가능한 소비자 전자제품, 통신망 PC, 미니 컴퓨터, 메인프레임 컴퓨터, 상술된 시스템들이나 디바이스들 중의 임의의 것을 포함하는 분산 컴퓨팅 환경 등.

[0041] 도구는 컴퓨터에 의해 실행되는, 프로그램 모듈들과 같은, 컴퓨터-실행가능 명령어들이 일반 문맥으로 기재될 수 있다. 일반적으로, 프로그램 모듈들은 특수 작업들을 수행하거나 특수 추상 데이터 유형들을 구현하는 루틴, 프로그램, 객체, 컴포넌트, 데이터 구조 등을 포함한다. 도구는 또한 작업들이 통신망을 통해 링크된 원격 프로세싱 디바이스들에 의해 수행되는 분산 컴퓨팅 환경에서 실시될 수 있다. 분산 컴퓨팅 환경에서, 프로그램 모듈들은 메모리 저장 디바이스들을 포함하는 로컬 및/또는 원격 컴퓨터 저장 매체에 위치될 수 있다.

[0042] 도 1을 참조하면, 도구를 구현하는 시스템의 예는 컴퓨터(110)의 형태로 일반 목적 컴퓨팅 디바이스를 포함한다. 컴퓨터(110)의 컴포넌트들은 프로세싱 유닛(120), 시스템 메모리(130), 및 프로세싱 유닛(120)에 시스템 메모리를 포함하는 다양한 시스템 컴포넌트들을 결합하는 시스템 버스(121)를 포함할 수 있지만, 이에 제한되지는 않는다. 시스템 버스(121)는 메모리 버스나 메모리 제어기, 주변기기 버스, 및 다양한 버스 구조들

중의 임의의 것을 사용하는 로컬 버스를 포함하는 여러 유형들의 버스 구조들 중의 임의의 것일 수 있다. 예를 들어, 그런 구조들은 산업 표준 아키텍처(ISA) 버스, 마이크로 채널 아키텍처(MCA) 버스, 개선된 ISA(EISA) 버스, 비디오 전자 표준 학회(VESA) 로컬 버스, 및 또한 메자닌 버스라고도 공지된 주변 컴포넌트 상호연결(PCI) 버스를 포함하지만, 이에 제한되는 것은 아니다.

[0043] 컴퓨터(110)는 통상적으로 다양한 컴퓨터-판독가능 매체들을 포함한다. 컴퓨터-판독가능 매체는 컴퓨터(110)에 의해 액세스가능한 임의의 이용가능한 매체일 수 있고, 휘발성과 비휘발성, 및 분리형과 비분리형 매체를 모두 포함한다. 예를 들어, 컴퓨터-판독가능 매체는 컴퓨터 저장 매체와 통신 매체를 포함할 수 있지만, 이에 제한되지는 않는다. 컴퓨터 저장 매체는, 컴퓨터-판독가능 명령어, 데이터 구조, 프로그램 모듈, 또는 기타 데이터와 같은, 정보 저장을 위한 임의의 방법이나 기술로 구현되는 휘발성과 비휘발성, 분리형과 비분리형 매체를 포함한다. 컴퓨터 저장 매체는 RAM, ROM, EEPROM, 플래쉬 메모리나 다른 메모리 기술, CD-ROM, 디지탈 다용도 디스크(DVD)나 기타 광 디스크 저장장치, 자기 카세트, 자기 테이프, 자기 디스크 저장 장치나 기타 자기 저장 디바이스, 또는 원하는 정보를 저장하기 위해 사용될 수 있고 컴퓨터(110)에 의해 액세스가능한 임의의 기타 매체를 포함하지만, 이에 제한되지는 않는다. 통신 매체들은 통상적으로 컴퓨터-판독가능 명령어, 데이터 구조, 프로그램 모듈, 또는 반송파나 다른 전송 매체와 같은 변조 데이터 신호의 기타 데이터를 구현하고, 임의의 정보 전달 매체를 포함한다. "변조 데이터 신호"라는 용어는 신호에서 정보를 인코딩하는 방식으로 그것의 하나 이상의 특징들이 세트되거나 변경되는 신호를 의미한다. 예를 들어, 통신 매체는 유선 통신망이나 직접 유선 접속과 같은 유선 매체들, 및 음향, RF, 적외선, 및 기타 무선 매체들과 같은 무선 매체를 포함하지만, 이에 제한되지는 않는다. 상술된 것들 중의 임의의 것의 조합들은 또한 컴퓨터-판독가능 매체들의 범위 내에 포함되어야 한다.

[0044] 시스템 메모리(130)는 읽기용 메모리(ROM)(131) 및 랜덤 액세스 메모리(RAM)(132)와 같은 휘발성 및/또는 비휘발성 메모리의 형태로 컴퓨터 저장 매체를 포함한다. 스타트업과 같은 때, 컴퓨터(110) 내의 소자들 간의 정보 전송을 돕는 기본 루틴들을 포함하는 기본 입/출력 시스템(BIOS)(133)은 통상적으로 ROM(131)에 저장된다. RAM(132)은 통상적으로 즉시 액세스가능하고 및/또는 현재 프로세싱 유닛(120)에 의해 동작 중의 데이터 및/또는 프로그램 모듈들을 포함한다. 예를 들어, 도 1은 운영 시스템(134), 응용 프로그램들(135), 기타 프로그램 모듈들(136), 및 프로그램 데이터(137)를 도시하지만, 이에 제한되는 것은 아니다.

[0045] 컴퓨터(110)는 또한 다른 분리형/비분리형, 휘발성/비휘발성 컴퓨터 저장 매체들을 포함할 수 있다. 예를 들어, 도 1은 비분리형 비휘발성 자기 매체에 읽고 쓰는 하드 디스크 드라이브(141), 분리형 비휘발성 자기 디스크(152)에 읽고 쓰는 자기 디스크 드라이브(151), 및 CD-ROM이나 기타 광 매체들과 같은 분리형 비휘발성 광 디스크(156)에 읽고 쓰는 광 디스크 드라이브(155)를 도시한다. 운영 환경의 예에서 사용될 수 있는 다른 분리형/비분리형, 휘발성/비휘발성 컴퓨터 저장 매체는 자기 테이프 카세트, 플래쉬 메모리 카드, 디지탈 다용도 디스크, 디지탈 비디오 테이프, 반도체 RAM, 반도체 ROM 등을 포함하지만, 이에 제한되지는 않는다. 하드 디스크 드라이브(141)는 통상적으로 인터페이스(140)와 같은 비분리형 메모리 인터페이스를 통해 시스템 버스(121)에 접속되고, 자기 디스크 드라이브(151)와 광 디스크 드라이브(155)는 통상적으로, 인터페이스(150)와 같은, 분리형 메모리 인터페이스에 의해 시스템 버스(121)에 접속된다.

[0046] 상술되고 도 1에 도시된, 드라이버들과 그들과 연관된 컴퓨터 저장 매체들은 컴퓨터(110)에 대한 컴퓨터-판독가능 명령어, 데이터 구조, 프로그램 모듈, 및 기타 데이터의 저장을 제공한다. 예를 들어, 도 1에서, 하드 디스크 드라이브(141)는 운영 시스템(144), 응용 프로그램들(145), 기타 프로그램 모듈들(146), 및 프로그램 데이터(147)를 저장하는 것으로 도시된다. 이들 컴포넌트들은 운영 시스템(134), 응용 프로그램들(135), 기타 프로그램 모듈들(136), 및 프로그램 데이터(137)와 동일하거나 상이할 수 있음을 주목한다. 운영 시스템(144), 응용 프로그램들(145), 기타 프로그램 모듈들(146), 및 프로그램 데이터(147)는 본 명세서에서 다른 부호들이 주어져서, 적어도, 그들이 다른 사본들임을 나타낸다. 사용자는 태블릿이나 전자 디지타이져(164), 마이크로폰(163), 키보드(162), 및 마우스, 트랙볼, 또는 터치패드로서 일반적으로 일컬어지는 포인팅 디바이스(161)와 같은 입력 디바이스들을 통해 컴퓨터(110)에 커맨드와 정보를 입력할 수 있다. 도 1에 도시 안된 다른 입력 디바이스들은 조이스틱, 게임 패드, 위성 접시, 스캐너 등을 포함할 수 있다. 이들과 다른 입력 디바이스들은 종종 시스템 버스와 접속된 사용자 입력 인터페이스(160)를 통해 프로세싱 유닛으로 접속되지만, 병렬 포트, 게임 포트, 또는 범용 직렬 버스(USB)와 같은, 다른 인터페이스와 버스 구조들에 의해 접속될 수 있다. 모니터(191)나 기타 유형의 디스플레이 디바이스는 또한, 비디오 인터페이스(190)와 같은, 인터페이스를 통해 시스템 버스(121)에 접속된다. 모니터(191)는 또한 터치-스크린 패널 등과 병합될 수 있다. 모니터 및/또는 터치-스크린 패널은, 태블릿-유형의 개인용 컴퓨터와 같이, 컴퓨팅 디바이스(110)가 병합된 하우징에 물리적으로 결합될 수 있음을

주목한다. 추가로, 컴퓨팅 디바이스(110)와 같은 컴퓨터들은 또한, 출력 주변기기 인터페이스(194) 등을 통해 접속될 수 있는, 스피커들(195) 및 프린터(196)와 같은 다른 주변 출력 디바이스들을 포함할 수 있다.

[0047] 컴퓨터(110)는, 원격 컴퓨터(180)와 같은, 한 개 이상의 원격 컴퓨터들에 논리 접속을 사용하여 통신망 환경에서 동작할 수 있다. 원격 컴퓨터(180)는 개인용 컴퓨터, 서버, 라우터, 통신망 PC, 피어 디바이스, 또는 기타 일반 통신망 노드일 수 있고, 단지 메모리 저장 디바이스(181)만이 도 1에 도시되지만, 통상적으로 컴퓨터(110)에 관련되어 상술된 소자들 중의 다수나 전체를 포함한다. 도 1에 도시된 논리 접속들은 구내 통신망(LAN)(171)과 광역 통신망(WAN)(173)을 포함하지만, 또한 다른 통신망들을 포함할 수 있다. 그런 통신망 환경들은 사무실, 기업-기반 컴퓨터 통신망, 인트라넷, 및 인터넷에서 일반적이다. 예를 들어, 현재 도구에서, 컴퓨터(110)는 데이터가 이동되는 소스 머신을 포함할 수 있고, 원격 컴퓨터(180)는 목적지 머신을 포함할 수 있다. 그러나, 소스와 목적지 머신들은 통신망이나 임의의 기타 수단으로 접속될 필요가 없고, 그 대신, 데이터는 소스 플랫폼에 의해 작성될 수 있는 임의의 매체를 통해 이동될 수 있고 목적지 플랫폼이나 플랫폼들에 의해 판독될 수 있음을 주목한다.

[0048] LAN 통신망 환경에서 사용될 때, 컴퓨터(110)는 통신망 인터페이스나 어댑터(170)를 통해 LAN(171)에 접속된다. WAN 통신망 환경에서 사용될 때, 컴퓨터(110)는 통상적으로, 인터넷과 같은, WAN(173)을 통해 통신을 개설하는 모뎀(172)이나 기타 수단을 포함한다. 내장이나 외장일 수 있는 모뎀(172)은 사용자 입력 인터페이스(160) 또는 기타 적합한 메카니즘을 통해 시스템 버스(121)로 접속될 수 있다. 통신망 환경에서, 컴퓨터(110), 또는 그 일부, 에 관련하여 도시된 프로그램 모듈들은 원격 메모리 저장 디바이스에 저장될 수 있다. 예를 들어, 도 1은 메모리 디바이스(181)에 상주하는 것으로서 원격 응용 프로그램들(185)을 도시하지만, 이에 제한되지는 않는다. 도시된 통신망 접속은 예일 뿐이고, 컴퓨터들 간에 통신 링크를 개설하는 기타 수단이 사용될 수 있음을 이해할 것이다.

[0049] 다양한 기능들과 데이터가 특정 방식으로 배치된 특정 컴퓨터 시스템들에 상주하는 것으로 도 1에서 도시되는 한편, 당업자들은 그런 기능들과 데이터가 상이한 배치들의 컴퓨터 시스템들 간에 다양한 다른 방식들로 분산될 수 있음을 이해할 것이다. 상술된 바와 같이 구성된 컴퓨터 시스템들은 통상적으로 도구의 동작을 지원하기 위해 사용되는 한편, 당업자는 도구가 다양한 유형들의 디바이스들과 구성들을 사용하고 다양한 컴포넌트들을 갖추어 구현될 수 있음을 이해할 것이다.

[0050] 도 2는 도구에 따라 컴퓨터 시스템들 간에 전형적인 데이터의 교환을 도시하는 데이터 흐름도이다. 도 2에 도시된 컴퓨터 시스템들(컴퓨터 시스템들(210, 220, 221, 222, 230, 231, 및 232))은 통상적으로 도 1과 연결하여 도시되고 논의된 컴포넌트들 중의 일부나 전부를 갖는다. 패치 분산 서버에서, 도구는 한 개 이상의 패치들을 생성한다. 이들 패치들(201)은 패치 분산 서버로부터, 관리 서버(220, 230)와 같은, 한 개 이상의 관리 서버들로 전송된다. 반면, 각각의 관리 서버는 패치들을, 타겟 컴퓨터 시스템들(221, 222, 231, 및 232)과 같은, 한 개 이상의 타겟 컴퓨터 시스템들로 전송한다. 일부 실시예들에서(도시 안됨), 패치 분산 서버는 한 개 이상의 타겟 컴퓨터 시스템들로 직접, 또는 한 개의 관리 서버를 통해서 보다 더 간접적인 루트를 통해 패치들을 전송한다. 타겟 컴퓨터 시스템에서 수신된 패치들은 아래 더 상세히 기재되는 바와 같이 타겟 컴퓨터 시스템에서 프로세스된다. 관리 서버는 또한, 패치 구성 커맨드들을 적용하여 특정 패치들의 동작을 재구성하는 한 개 이상의 타겟 컴퓨터 시스템들에게 패치 구성 커맨드들(202)을 전송할 수 있다. 아래 더 상세히 기재되는 바와 같이, 패치는 완전히 중지될 수 있다; 패치가 중지되지 않으면, 그것의 동작 및 그것의 테스트 결과 핸들링의 통지가 각각 독립적으로 가동되거나 중지될 수 있다. 그것의 동작의 통지가 가동될 때, 통지들은 타겟 컴퓨터 시스템에 디스플레이되거나 로컬에 저장될 수 있거나, 또는 통지들(203)로서 적합한 관리 서버로 전송될 수 있다.

[0051] 도 3은 새 패치를 수신하고 프로세스하기 위해 도구에 의해 통상적으로 수행되는 단계들을 도시하는 흐름도이다. 단계(301)에서, 도구는 패치를 수신한다. 단계(301)에서 수신되는 패치는 데이터-구동 패치 또는 코드-구동 패치일 수 있다. 샘플 데이터-구동 패치는 아래 표 1에 도시된다.

표 1

```
1       <Softpatch Patch="Q382429">
2        <AffectedApplication AffectedExe="sqlservr.exe">
3         <AffectedVersion Version="9.*">
4          <AffectedModules Name="SQLSORT.DLL">
5           <Version "8.0.*, 9.*">
6                 <Function Name="SsrpEnumCore" Address="0x0802E76B"
7                     Param="2" Paramtype="LPSTR">
8                  <Filter MaxByteLength="60" />
9                  <Resolution ActionType="BOOL" Action="FALSE" />
10                </Function>
11          </Version>
12          <Version "10.*, 11.*">
13                <Function Name="SsrpEnumCore" Address="0x0802D283"
14                    Param="2" Paramtype="LPSTR">
15                 <Filter MaxByteLength="128" />
16                 <Resolution ActionType="BOOL" Action="FALSE" />
17                </Function>
18          </Version>
19         </AffectedModules>
20        </AffectedVersion>
21       </AffectedApplication>
22
23       <Signature Hash="MD5" Signature="C509-64AA-9161-8C52-
24        9F6D-BF5A-AEF2-ECE1-0038-34D1"/>
25      </Softpatch>
```

[0052]

[0053] 라인 1은 패치에 대한 고유 식별자를 포함한다. 라인 2는 패치에 의해 영향받는 응용 프로그램을 식별한다. 라인 3은 패치에 의해 영향을 받는 응용 프로그램 버젼을 식별한다. 라인 4는 패치에 의해 영향을 받는 실행가능 모듈을 식별한다. 라인 5는 패칭 방향들이 제공되는 영향을 받는 실행가능 모듈의 2개의 버젼들 -버젼 8.0.* 및 버젼 9.*- 을 식별한다. 라인 6 내지 라인 10은 이들 실행가능 모듈의 2개 버젼들에 대한 패칭 방향들을 포함한다. 라인 6 내지 라인 7은 패치되는 함수, 실행가능 모듈에서의 그것의 주소, 패치에 의해 테스트되는 그것의 파라미터, 및 테스트되는 파라미터 유형을 식별한다. 라인 8은 라인 6 내지 라인 7에서 식별되는 파라미터는 그것의 길이가 60 바이트를 초과하는지를 판정하기 위해 테스트되어야 함을 지시한다. 라인 9는 테스트가 성공하면 함수 호출이 실패해야 함을 지시한다. 라인 12는 패칭 방향들이 제공되는 영향을 받는 실행가능 모듈의 2개 이상의 버젼들 -버젼들 10.* 및 11.*- 을 식별한다. 라인 13 내지 라인 17은 이들 실행가능 모듈의 2개의 버젼들을 위한 패칭 방향들을 포함한다. 라인 13 내지 라인 14는 패치되는 함수, 실행가능 모듈에서의 그것의 주소, 패치에 의해 테스트되는 그것의 파라미터, 및 테스트되는 파라미터 유형을 식별한다. 버젼 10.*과 11.*에 대한 라인 13 내지 라인 14에서 식별되는 실행가능 모듈에서 패치되는 함수의 주소는 버젼 8.0.*과 9.*를 위한 라인 6 내지 라인 7에서 식별되는 패치되는 함수의 주소와는 상이함을 볼 수 있을 것이다. 라인 15는 라인 13 내지 라인 14에서 식별된 파라미터가 그것의 길이가 128 바이트를 초과하는지를 판정하기 위해 테스트되어야 함을 지시한다. 라인 16은, 테스트가 성공하면, 함수의 호출이 실패해야 함을 지시한다. 패치는 패치된 함수의 호출을 실패하게 하는 것, 예외처리를 발생시키는 것, 패치된 실행가능 모듈이 실행되는 프로세스를 종료시키는 것, 또는 공격적인 값을 교정하는 것(예를 들어, 너무 긴 스트링을 절단함)을 포함하는, 다양한 결과 핸들링 액션 유형들을 명시할 수 있다. 라인 23 내지 라인 25는 패치의 소스를 식별하고 패치가 그것의 소스를 떠난 후로 변경되지 않았음을 확인시키기도 하는 패치에 대한 서명을 포함한다.

[0054] 아래 표 2는 위의 표 1에 도시된 패치의 코드-구동 버젼을 포함한다.

표 2

```
1    00411A7E  push     3Ch
2    00411A80  mov      eax,dword ptr [str]
3    00411A83  push     eax
4    00411A84  call     ValidateStringLength (411082h)
5    00411A89  add      esp,8
6    00411A8C  movzx    ecx,al
7    00411A8F  test     ecx,ecx
8    00411A91  je       411A9Ah
9    00411A93  jmp      foo+2 (411AD2h)
10   00411A9A  xor      eax,eax
11   00411A9C  ret
```

[0055]

[0056] 라인 1 내지 라인 3은 테스팅 함수에 대한 파라미터들을 스택으로 밀어넣는다. 라인 4는 테스팅 함수를 호출한다. 라인 5 내지 라인 8은 테스팅 함수에 대한 리턴 코드에 분기한다. 테스팅 함수가 성공하면, 라인 9는 패치된 함수 몸체의 실행을 시작하기 위해 뒤로 점프한다. 테스팅 함수가 실패하면, 라인 10 내지 라인 11은 실패 결과 코드를 스택으로 밀어넣고, 그 패치된 함수로부터 패치된 함수의 칼라로 리턴한다. 판독성을 위해, 표 2는 확인가능한 서명, 패치 구성 플래그들의 현재 값들을 테스트하는 명령어, 및 패치된 함수를 위한 코드의 시작으로부터 재위치되는 명령어를 포함하는, 일부 코드-구동 패치들에 존재하는 특정 세부사항들을 생략한다.

[0057] 일부 실시예들에서, 양쪽 유형들의 패치들은, 패치되는 실행가능 모듈의 한 개 이상의 버젼들의 각각에 대해, 실행가능 모듈의 특정 인스턴스가 그 버젼의 적절한 사본임을 유효화하기 위해 사용될 수 있는 파일 서명을 포함하는 추가 정보를 포함할 수 있다. 그런 파일 서명은, 예를 들어, 전체 실행가능 모듈 버젼에 대한 크기나 검사합(checksum), 또는, 실행가능 모듈이 패치되는 오프셋(offset)에서와 같이, 실행가능 모듈의 특정 지점에서 발생되도록 기대되는 코드일 수 있다.

[0058] 단계(302)에서, 패치가 유효한 서명으로 사인되면, 도구는 단계(303)에서 계속되고, 그렇지 않으면, 도구는 단계(301)에서 계속하여 다음 패치를 수신한다. 단계(303)에서, 도구는 로컬 패치 표에 패치를 추가한다. 단계(304)에서, 도구는, 예를 들어, 그것을 디폴트 구성으로 전송하여, 패치의 초기 구성을 초기화시킨다.

[0059] 도 4는 도구에 의해 통상적으로 사용되는 것들의 샘플 패치 표를 도시하는 데이터 구조 도면이다. 패치 표(400)는 행(401)과 행(402)과 같은 행들을 포함하고, 각각의 행은 다음 열들로 분리된다: 패치로부터 추출된 패치 식별자를 포함하는 패치 식별자 열(411); 그것의 이름과 같은, 패치되는 실행가능 모듈을 식별하는 정보를 포함하는 실행가능 모듈 열(412); 패치가 적용되는 열(412)에서 식별되는 실행가능 모듈의 모든 버젼들을 식별하는 실행가능 모듈 버젼 열(413); 패치에 의해 명시된 테스트가 패치 함수가 호출되는 각각의 시간에 수행되어야 하는지에 대한 현재 구성 값을 포함하는 테스트 수행 가동 열(414); 통지가 패치 테스트가 수행되는 각각의 시간에 생성되어야 하는지에 대한 현재 구성 값을 포함하는 테스트 수행 통지 가동 열(415); 통지가 패치 테스트가 성공하는 각각의 시간에 생성되어야 하는지에 대한 현재 구성 값을 포함하는 테스트 결과 통지 가동 열(416); 패치 결과 핸들링이 패치 테스트가 실패할 때 구현되어야 하는지에 대한 현재 구성 값을 포함하는 테스트 결과 핸들링 가동 열(417); 및 테스트가 실패하는 각각의 시간에 수행되는 테스트와 테스트 결과 핸들링을 명시하는 패치 자체로의 포인터를 포함하는 패치 열(418). 일부 실시예들에서, 도시된 바와 같이 패치로의 포인터를 포함하는 대신에, 패치 열(418)은 각각의 패치를 직접 포함한다. 특정 패치 표는, 모든 코드-구동 패치들, 모든 데이터-구동 패치들, 또는 코드-구동 및 데이터-구동 패치들의 조합과 같은, 다양한 유형들의 패치들을 포함하거나 포인트할 수 있다.

[0060] 단계(305)에서, 일단 도구가 패치 표에 수신된 패치를 추가하여 그것의 구성을 초기화하면, 패치는 도구에 의해 실행가능 모듈로 자동 적용되도록 이용할 수 있다. 단계(305)에서, 도구는 참조되는 출원서에 기재되는 것들, 실시간 함수 호출 인터셉션, 및/또는 운영 시스템 로더에 의해 로드되는 (1) 이미 로드된 실행가능 모듈들, (2) 실행가능 모듈의 한 개 이상의 디스크 이미지들, 또는 (3) 실행가능 모듈들의 인스턴스들의 코드 재작성을 포함하는, 패치의 적용에 대한 다양한 접근법들을 사용할 수 있다. 단계(305) 이후에, 도구는 단계(301)로 진행하여 다음 패치를 수신한다.

[0061] 도 5는 특정 패치에 대한 구성 명령어들을 업데이트하기 위해 도구에 의해 통상적으로 수행되는 단계들을 도시하는 흐름도이다. 단계(501)에서, 도구는, 예를 들어, 관리자 등으로부터 특정 패치에 대한 구성 명령어들을 수신한다. 일부 실시예들에서, 그런 구성 명령어들은 그룹 정책을 사용하여 관리자들에 의해 생성될 수 있다. 단계(502)에서, 도구는 수신 명령들에 따라 패치 표의 패치 구성을 업데이트한다. 단계(502) 이후에, 도구는 다음 구성 명령어들을 수신하기 위해 단계(501)에서 계속된다.

[0062] 도 6은 패치에 의해 명시된 파라미터 유효화를 도구에 의해 수행하기 위해 통상적으로 수행되는 단계들을 도시하는 흐름도이다. 단계(601)에서, 패치된 함수가 호출된다. 단계(602)에서, 호출된 함수에 영향을 미치는 패치에 대해 테스팅이 가동되면, 도구는 단계(603)에서 계속되고, 그렇지 않으면 도구는 단계(601)에서 패치된 함수로의 다음 호출을 프로세스하기 위해 계속된다. 단계(603)에서, 테스트 수행 통지가 패치에 대해 가동되면, 도구는 단계(604)에서 계속되고, 그렇지 않으면 도구는 단계(605)에서 계속된다. 단계(604)에서, 도구는 테스트가 수행되었다는 통지를 생성한다. 단계(604, 608, 및 610)는 타겟 컴퓨터 시스템에 테스트가 만족되었다는 지시를 디스플레이하거나 저장하는 것 및/또는 그런 지시를 원격 컴퓨터 시스템에서 디스플레이하거나 거기에 로깅하기 위해 전송하는 것과 관련이 있을 것이다.

[0063] 단계(605)에서, 도구는 패치에 의해 명시된 유효화 테스트를 수행한다. 일부 실시예들에서, 단계(605)는 테스팅을 위해 도구에 의해 사용되는 표준 루틴들의 그룹 중의 하나를 호출하는 것과 관련이 있다. 단계(606)에서, 단계(605)에서 수행되는 테스트가 만족되면, 도구는 단계(601)에서 계속되고, 그렇지 않으면 도구는 단계(607)에서 계속된다. 단계(607)에서, 테스트 결과 통지가 패치에 대해 가동되면, 도구는 단계(608)에서 계속되고, 그렇지 않으면 도구는 단계(609)에서 계속된다. 단계(608)에서, 도구는 테스트가 만족되지 않았다는 통지를 생성한다. 단계(609)에서, 테스트 결과 핸들링이 패치에 대해 가동되면, 도구는 단계(610)에서 계속되고, 그렇지 않으면 도구는 단계(601)에서 계속된다. 단계(610)에서, 도구는 패치에 의해 명시된 테스트 결과 핸들링을 수

행한다. 단계(610) 이후에, 도구는 단계(601)에서 계속된다.

[0064] 당업자라면 상술된 도구가 다양한 방식들로 쉽게 적응되고 확장될 수 있음을 이해할 것이다. 예를 들어, 도구는 다양한 목적으로 다양한 유형의 실행가능 모듈들에서 다양한 위치들에서 다양한 방식으로 다양하고 상이한 유형들의 패치들을 적용하기 위해 사용될 수 있다. 또한, 본 명세서에서 패치들이 그들이 실패할 때 문제를 지시하는 값 유효화 테스트들을 포함하는 것으로 기재되는 한편, 도구는 또한 그들이 성공할 때 문제를 지시하는 값 무효 테스트들을 사용하여 구현될 수 있다. 일부 실시예들에서, 각각의 테스트는 그것의 성공이나 실패가 문제를 지시하는지에 대한 지시가 동반된다. 상술된 기재에서 선호되는 실시예들로의 참조가 되는 한편, 본 발명의 범위는 뒤따르는 청구범위 및 거기에 기재되는 소자들에 의해서만 정의된다.

발명의 효과

[0065] 타겟 컴퓨터 시스템에 소프트웨어를 효과적으로 패칭하는 방법 및 시스템이 개시된다.

도면의 간단한 설명

[0001] 도 1은 도구가 구현될 수 있는 적합한 컴퓨팅 시스템 환경의 예를 도시한다.

[0002] 도 2는 도구에 따라 컴퓨터 시스템들 간에 전형적 데이터 교환을 도시하는 데이터 흐름도이다.

[0003] 도 3은 새 패치를 수신하여 프로세스하기 위해 도구에 의해 통상적으로 수행되는 단계들을 도시하는 흐름도이다.

[0004] 도 4는 도구에 의해 사용되는 통상적인 것들의 샘플 패치 표를 도시하는 데이터 구조 도면이다.

[0005] 도 5는 특정 패치에 대한 구성 명령어들을 업데이트하기 위해 도구에 의해 통상적으로 수행되는 단계들을 도시하는 흐름도이다.

[0006] 도 6은 패치에 의해 명시된 파라미터 유효화를 수행하기 위해 도구에 의해 통상적으로 수행되는 단계들을 도시하는 흐름도이다.

[0007] <주요 도면 부호 설명>

[0008] 210 패치 배포 서버

[0009] 201 패치들

[0010] 220, 230 관리 서버

[0011] 203 통지

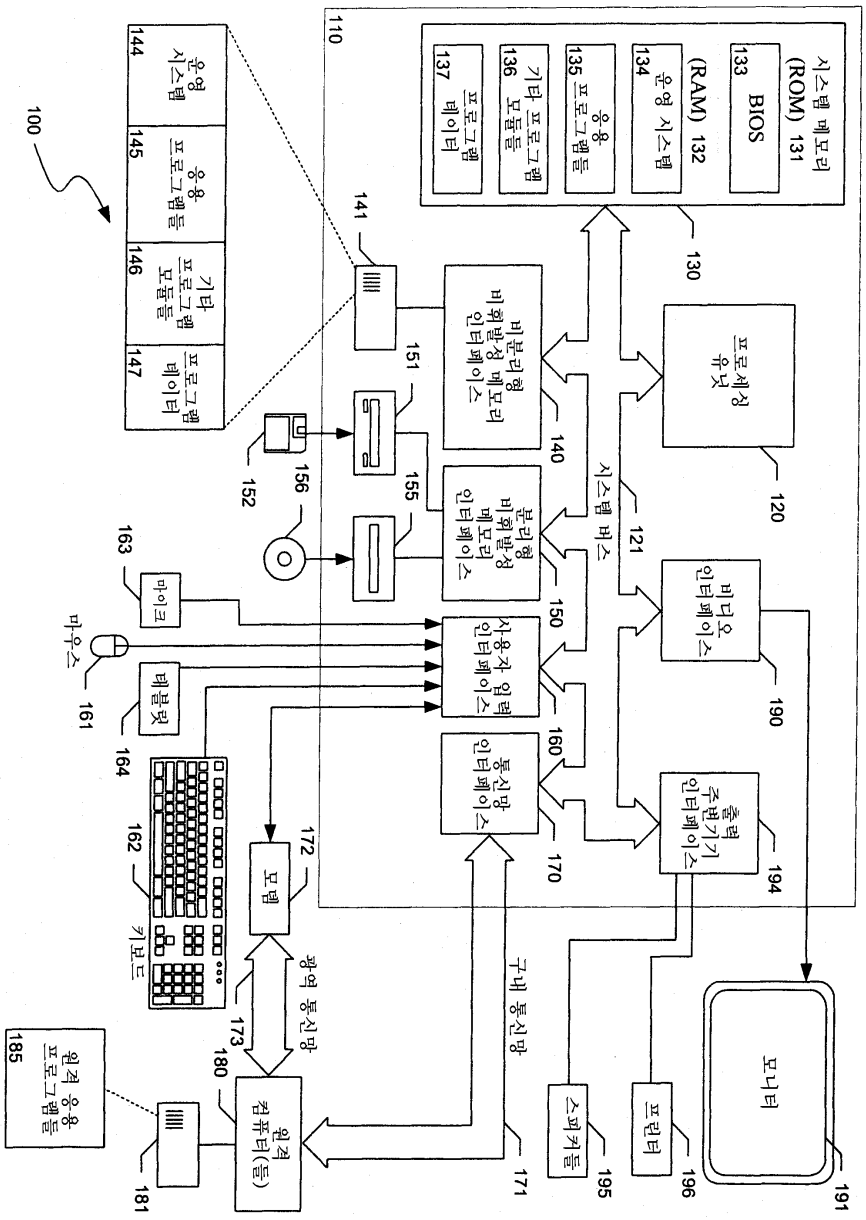
[0012] 202 패치 구성들

[0013] 201 패치들

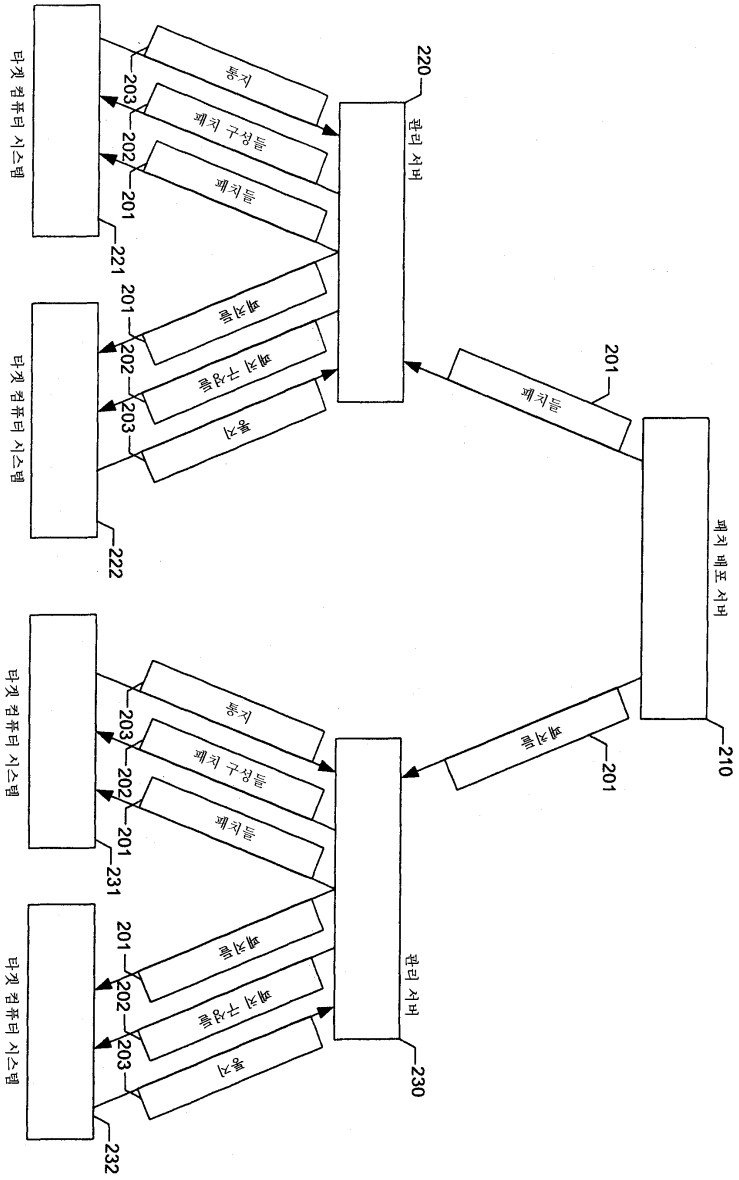
[0014] 221, 222, 231, 232 타겟 컴퓨터 시스템

도면

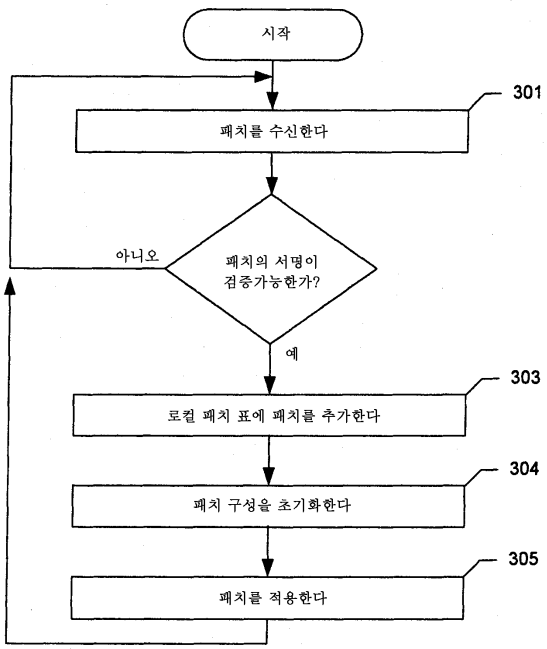
도면1


100
110
시스템 메모리 (ROM) 131
BIOS 133
(RAM) 132 운영 시스템 134
응용 프로그램들 135
기타 프로그램 모듈들 136
프로그램 데이터 137
130
프로세싱 유닛 120
시스템 버스 121
비분리형 비휘발성 메모리 인터페이스 140
분리형 비휘발성 메모리 인터페이스 150
비디오 인터페이스 190
출력 주변기기 인터페이스 194
사용자 입력 인터페이스 160
통신망 인터페이스 170
141
151
155
156
152
163
마이크
마우스
161
태블릿
164
162
키보드
172
모뎀
173
광역 통신망
180
원격 컴퓨터(들)
181
원격 응용 프로그램들 185
구내 통신망
171
스피커들 195
프린터 196
모니터 191
운영 시스템 144
응용 프로그램들 145
기타 프로그램 모듈들 146
프로그램 데이터 147

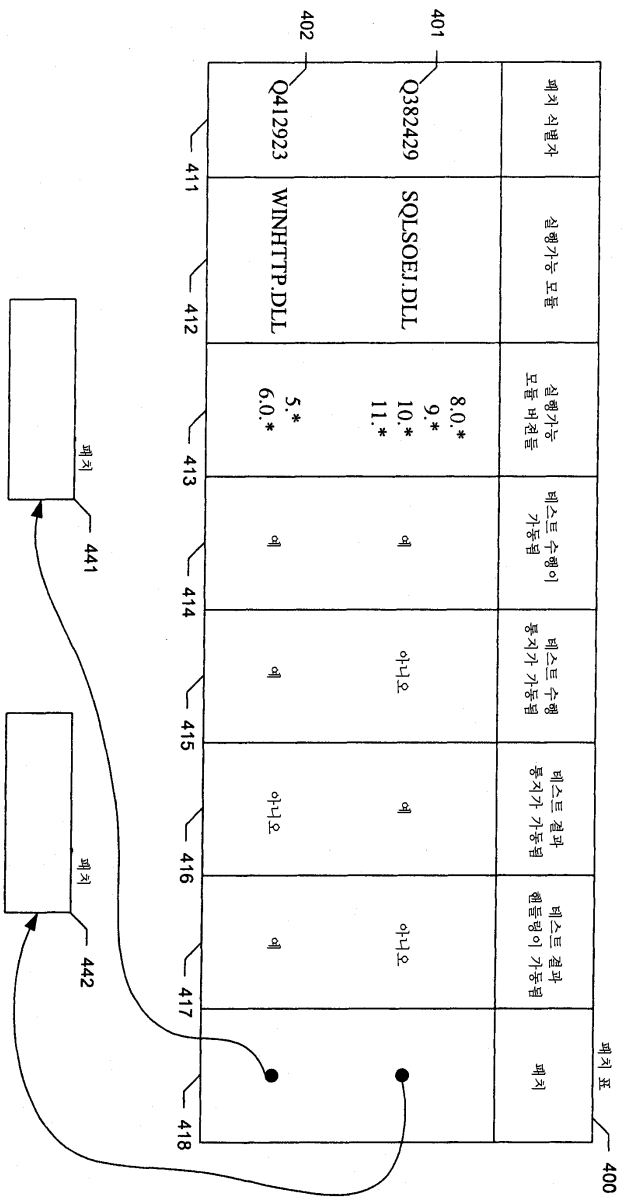
도면2


패치 배포 서버
210
패치들
201
관리 서버
220
통지
패치 구성들
패치들
203
202
201
타겟 컴퓨터 시스템
221
복귀들
복귀 구성들
통지
222
관리 서버
230
231
232

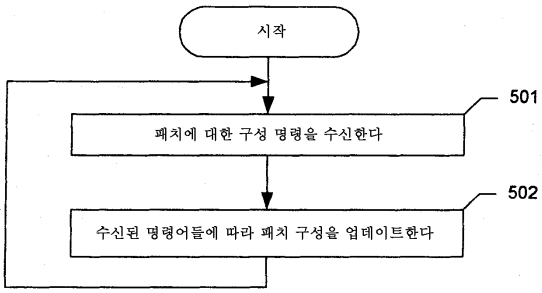
도면3


시작
301
패치를 수신한다
아니오
패치의 서명이
검증가능한가?
예
303
로컬 패치 표에 패치를 추가한다
304
패치 구성을 초기화한다
305
패치를 적용한다

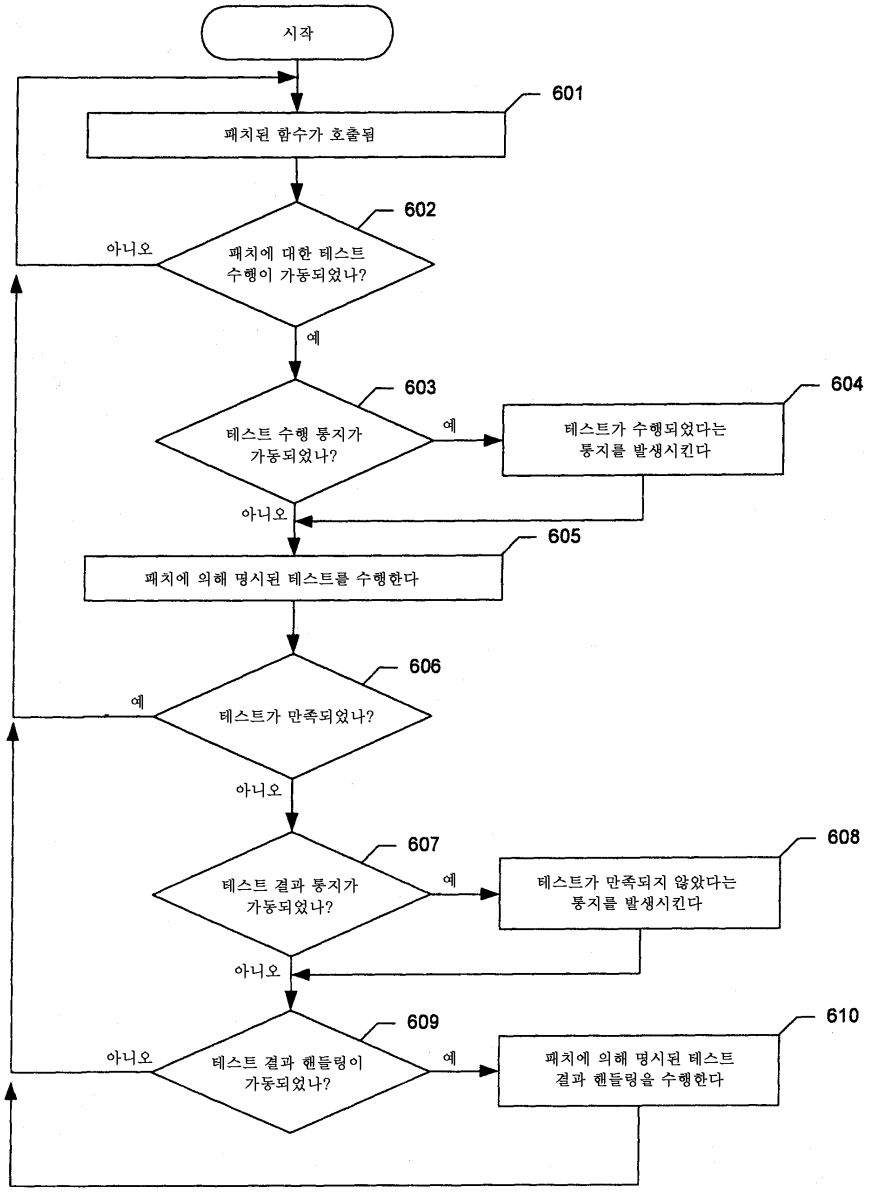
도면4


패치 식별자
실행가능 모듈
실행가능 모듈 버전들
테스트 수행이 가능됨
테스트 수행 롤지가 가능됨
테스트 결과 롤지가 가능됨
테스트 결과 핸들링이 가능됨
패치
Q382429
SQLSOEJ.DLL
8.0.*
9.*
10.*
11.*
예
예
아니오
예
아니오
예
아니오
Q412923
WINHTTP.DLL
5.*
6.0.*
예
예
아니오
예
아니오
패치
패치
401
402
411
412
413
414
415
416
417
418
400
441
442
패치 표

도면5


시작
패치에 대한 구성 명령을 수신한다
수신된 명령어들에 따라 패치 구성을 업데이트한다
501
502

도면6


시작
601
패치된 함수가 호출됨
602
패치에 대한 테스트 수행이 가동되었나?
아니오
예
603
테스트 수행 통지가 가동되었나?
예
604
테스트가 수행되었다는 통지를 발생시킨다
아니오
605
패치에 의해 명시된 테스트를 수행한다
606
테스트가 만족되었나?
예
아니오
607
테스트 결과 통지가 가동되었나?
예
608
테스트가 만족되지 않았다는 통지를 발생시킨다
아니오
609
테스트 결과 핸들링이 가동되었나?
아니오
예
610
패치에 의해 명시된 테스트 결과 핸들링을 수행한다