情報保護學會論文誌
第13券 第4號, 2003. 8

안전한 보안패치 분배 구조의 설계 및 구현

손 태 식*, 서 정 우*, 문 종 섭*, 서 정 택**, 임 을 규**, 이 철 원**

Design and Implementation of a Secure Software Architecture for Security Patch Distribution

Tae-Shik Sohn*, Jung-Woo Seo*, Jong-Sub Moon*, Jung-Taek Seo**,
Eul-Gyo Im**, Cheol-Won Lee**

요 약

패치 분배는 시스템의 보안과 네트워크를 구성하는 여러 시스템들에 설치된 소프트웨어의 취약성을 보완하기 위한 가장 중요한 요소 중의 하나이다. 특정 기관이나 조직은 다양한 운영체제 및 그 응용 프로그램을 사용하기 때문에 매번 신속하게 시스템의 취약성에 대해 패치를 적용하는 것은 쉽지 않다. 본 논문에서 우리는 안전한 패치 분배 구조를 제안할 것이며 제안하는 구조는 사용자 인증, 패치 검증, 패치 자동분배 및 설치 등의 다양한 메커니즘을 가진다. 이렇게 제안된 패치 분배 구조는 도메인내의 패치 분배 과정에 있어서의 보안성을 더욱 향상시킬 것이다.

ABSTRACT

Patch distribution is one of important processes to fix vulnerabilities of software and to ensure security of systems. Since an institute or a company has various operating systems or applications, it is not easy to update patches promptly. In this paper, we propose a secure patch distribution architecture with an authentication mechanism, a security assurance mechanism, a patch integrity assurance mechanism, and an automatic patch installation mechanism. We argue that the proposed architecture improve security of patch distribution processes within a domain.

keyword : *security patch, vulnerability, exploit, software distribution*

Ⅰ. 서 론

운영체제 및 여러 응용 프로그램들은 프로그램 개발 과정의 특성상 보안 취약성을 가지고 있기 마련이다. 또한 현재 인터넷이 폭넓게 보급되어, 네트워크를 통한 시스템 침해 사례가 증가하고 있다. 이러한 현실에서 패치에 대한 안전하고 신속한 분배 및 원격지에서의 자동 설치 같은 서비스 제공은 해당 시스템의 보안을 위한 가장 기본적이고 필수적인 요소이다. 패치는 대상 시스템에서 있어 취약성을 보완해주는 일종의 보조 프로그램이라고 할 수 있으며, 이러한 패치는 시스템에 주기적으로 업데이트가 수반되어야 하는 필수적인 요소이다. 하지만 시스템 역공학적인 관점에서 볼 때 해당 시스템에 필요한 패치의 종류, 설치된 프로그램에 대한 부가 사항 등의 패치 정보가 누출된다면, 결국 대상 시스템의 취약한 부분을 그대로 외부에 알려주는 것과 같은 결과를 초래할 수 있다. 또한 악의의 제3자에 의해 트로이안 목마와 같은 백도어 기능이 삽입된 패치가 자신의 네트워크 내부의 중요 시스템에 설치된다면,

* 고려대학교 정보보호기술연구센터, 정보보호대학원({743zh2k, korea002, jsmoon}@korea.ac.kr)
** ETRI 부설 국가보안기술연구소({seojt, imeg, cheolee}@etri.re.kr)

시스템 침해 및 기밀 정보 누출이라는 보안상 치명적인 문제점을 가져 올 수 있다. 즉, 공개 환경에서의 개인 사용자의 경우와 달리 특정 조직이나 기관의 시스템에 대한 패치 분배는 민감한 문제가 아닐 수 없다.[1~4]

그러므로 본 논문에서는, 운영체제 벤더들의 패치 자동분배 방안에 대한 기술 분석을 바탕으로, 특정 조직이나 기관의 단일 도메인 내의 안전한 패치 분배 구조에 대하여 제안한다. 본 논문은 다음과 같이 구성되어 있다. 2장에서는 기존 패치 분배에 관련 연구를, 3장에서는 제안하는 보안패치 분배 구조에 대한 연구를, 4장에서는 제안된 구조에 대한 설계 및 구현에 대한 연구를 그리고 5장에서는 구현 결과 및 검증 예제를 통해 제안시스템을 테스트하며 마지막으로 6장에서 결론 및 향후 연구 방향을 제시한다.

II. 패치 분배 관련 연구

2.1 벤더들의 패치 분배 동향

본 논문에서는 [표 1]과 같이 여러 운영체제 벤더들이 패치를 분배하는 과정에 있어서의 패치 분배 및 인증, 무결성, 기밀성 제공 방안에 대하여 분석하였다. 특히 각 벤더별 분배 과정에서 사용되는 기법과 각 방안 사이의 차이점 그리고 분배 방안에 있어서의 취약점에 초점을 맞췄다. 이러한 본 논문의 분석 내용은 BindView 연구 보고서를 인용하였으며[5~10] 또한 연구 과정에서의 벤더별 사항을 업데이트 하였다.

대부분의 벤더들은 별도의 분배 방안을 적용하지 않고 일반적인 전자메일, FTP 사이트, HTTP 사이트를 이용하여 패치를 분배하였다. 이러한 보편적 분배 방안을 사용하는 이유로는 우선 첫째 벤더들의 보안 패치 분배에 있어서 해당 운영체제를 사용하는 다수의 일반 사용자들을 대상으로 하기 때문이며, 둘째 임의의 다수 사용자에 대한 특정 기법을 사용하는 분배 방안을 적용하는 데에는 기술적으로나 비용적인 측면으로나 많은 어려움이 있기 때문일 것으로 짐작할 수 있다. 또한 위와 같은 분배 과정에서 각 벤더들은 분배되는 패치의 인증, 무결성, 기밀성 보장을 위한 방안으로 [표 1]에서 나타나는 것과 같이 PGP, HTTPs, SSH 등을 주로 적용하는 것을 알 수 있다. 이러한 분배 기법에 대해서 다음절에서 분석한다.[11~22]

〔표 1〕 벤더별 분배 기술 분석

적용기법 / 운영체제	PGP	HTTPs	SSH	Package
Caldera/Open Linux	O			O
Cobalt				O
Compaq/Tru64 unix		O		
Conectiva Linux	O			O
Corel	O			
Debian Linux	O			O
FreeBSD	O			
HP-UX	O	O		
IBM-AIX	O	O		
WireX/Immunix	O			
Mandrake Linux	O			O
MicroSoft Windows	O			
NetBSD	O		O	
Novell/Netware				
OpenBSD			O	
Redhat Linux	O			O
SCO				
SGI/IRX	O			
Slackware Linux				
Sun Solaris	O	O		
Suse Linux	O			O
Trustix	O			O
Turbo Linux	O	O		O

2.2 기존 패치 분배 방안 분석

패치 분배 과정에서는 패치를 분배받는 사용자에 대한 인증, 패치 파일에 대한 무결성 보장 , 전송 과정의 기밀성 보장 등 여러 보안 사항이 요구된다. 본 논문의 기존 패치 분배 동향 분석 결과에 따르면 패치 분배 과정에 있어서 널리 사용되는 기법은 아래와 같은 세 가지 기법으로 크게 나눌 수 있다.

- PGP(Pretty Good Privacy)를 이용
  :첨부되는 전자 서명이 전체 패치 파일 또는 일부 패치 문서에 대한 서명에 대해서도 사용 가능하며 서명된 패치에 대한 인증, 무결성 제공
- HTTPs 연결 설정을 통한 패치 분배
  :SSL(Secure Socket Layer) 기반의 패치 분배 및 기밀성 보장 기능을 제공
- SSH(Secure SHell)을 통한 패치 분배
  :클라이언트 서버간의 보안쉘을 설정하여 패치 분

배 과정에 있어서 기밀성 제공

PGP를 사용하는 경우에는 우선 패치 파일 자체에 대해 서명을 하는 경우와 패치 파일의 소스에 대한 서명, 패치 파일 정보 메시지에 대한 서명 등으로 나뉠 수 있다. 현재 PGP를 사용한 패치 파일의 분배는 보안패치의 분배에 있어서 가장 보편적인 방법이다. 하지만 PGP의 경우 패치 분배시 패치 파일의 서명에 사용된 키의 신뢰성을 사용자가 어느 만큼이나 보장할 수 있는지에 대한 문제점이 남는다. PGP의 경우 가장 일반적인 패치 인증 방법으로서 패치 인증과정 중에 패치에 대한 MD5 체크섬 기능이 포함되기 때문에 PGP의 사용과 함께 패치분배 과정에서의 기밀성을 보장해 줄 수 있는 HTTPs 나 SSH와 함께 사용도 고려되고 있다.

HTTPs는 인증서를 기반으로 하여 서버인증 및 사용자인증 모두를 지원한다. 또한 PGP와는 다르게 공개키를 기반으로 한 키 교환기능을 통하여 기밀 통신 채널을 확립하여 분배되는 패치의 기밀성도 제공할 수 있다. 다만 서버 인증과정에서 서버의 인증서를 신뢰의 문제, 패치 분배 서버와 반복적인 연결 설정이 일어나는 경우 연결 설정과 통신 과정의 암호화 연산에 관련된 오버헤드 그리고 다수의 패치 분배를 위한 미러 사이트가 존재하는 경우에 모든 분배 사이트에 대한 서버 인증서 발급 등의 문제가 남아 있다.

HTTPs와 마찬가지로 SSH 역시 공개키를 기반으로 하여 패치분배 과정에서 기밀 통신 채널을 제공해준다. 하지만 SSH 서버에 저장된 SSH 서버의 개인키 누출 등의 염려가 존재한다.

2.3 패치분배 과정에서의 취약점

보안패치 분배 과정에 있어서 사용자들은 일반적으로 패치가 정당한 사이트로부터 인증되어 자신에게 올바르게 배포되었다고 여기게 된다. 이러한 이유는 대부분의 패치 분배는 벤더가 제공하는 자체 사이트를 통해서 분배되기 때문이다. 따라서 이러한 사용자의 잘못된 믿음과 보안패치 분배 과정 자체에서 기인하는 여러 보안 취약성으로 인해서 많은 문제점을 내포하고 있다. 아래의 사항들은 보안패치 분배 과정에서 발생할 수 있는 취약성들을 나열한 것이다.

- 밴더의 패치 서명과 사용자의 패치 검증에 사용되는 키가 취소되는 경우에 그 즉시 사용자가 이러한 키 폐기 사실을 알기 어려움
- 패치 인증에 사용된 서명키가 실제로 벤더의 서명키인지 명확하게 검증하는 것이 어려움
- 전자 서명이 패치 파일 자체에 대한 인증이 아닌 단지 패치 정보의 인증에만 사용되는 경우
- 올바른 서명키로 악의의 목적을 가지고 제작된 패치가 서명되어 분배되는 경우
- 정당한 서명자에 의한 패치의 서명과정 중 서명자의 실수로 인해 잘못된 패치에 대한 서명이 발생하고 분배되는 경우
- 디지털 서명 소프트웨어(예 PGP)나 이 소프트웨어를 신뢰하는 OS의 구성요소에 트로이 목마 프로그램의 배포

2.4 안전한 보안패치 분배 구조 제안 동기

패치란 이미 정식 릴리즈를 거친 소프트웨어나 운영체제, 모듈의 문제점을 보완하고, 이들의 결점을 해결해주는 공급자의 추가 패키지라고 할 수 있다. 이러한 패치는 비단 보안 요소뿐만이 아니라, 응용프로그램의 실행속도 개선, 메모리 누수 개선 등과 같이 소프트웨어의 여러 측면에서의 성능을 향상 시켜주기 위한 것이다. 그러므로 패치 업데이트는 시스템의 보안 및 성능 향상을 위해서 행해져 할 가장 근본적인 사항이며 패치가 담고 있는 정보는 바로 시스템 자체에 대한 취약성을 담고 있으므로 노출시에는 잠재적 시스템 침해의 문제를 야기한다.

앞서 논의된 인터넷 환경에서의 패치분배는 그 대상이나 시스템 구성원의 다양성으로 패치분배 과정에서의 효율성이나 보안성에 문제가 있다. 또한 이외에도 분배되는 다양한 패치 처리의 어려움, 여러 사용자에 대한 관리의 어려움 등의 문제를 가지고 있다.

그러므로 본 논문에서 이러한 패치 분배 환경 고려, 분배되는 다양한 패치 관리, 편리하고 효율적인 자동화된 패치 분배 방안 등에 초점을 맞춘 새로운 보안패치 분배구조를 제안한다.

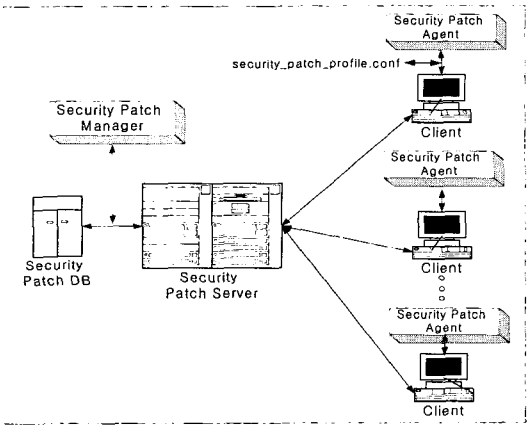
III. 제안하는 보안패치 분배 구조

본 논문에서 제안하는 안전한 보안패치 분배 구조는 특정 조직이나 기업의 제한된 도메인을 그 대상으로 한정한다. 또한 이러한 제한된 도메인을 구성원간 기본적인 신뢰도를 가지는 보안 도메인으로서 가정한다. 따라서 본 제안 구조는 보안 도메인내의 시

스템에 대한 보안패치 자동분배 방안 및 자동분배에 있어서 요구되는 여러 보안 사항을 만족시키는 것을 목표로 한다. 본 장에서는 제안 구조의 전체 구조와 자동분배 메커니즘 그리고 메시지 핸들링을 기반으로 한 전체 시스템 구현 방안에 대해서 서술한다.

3.1 전체 시스템 구조

안전한 보안패치 분배 구조는 [그림 1]과 같은 구성 요소를 가지고 있다. 우선 보안패치를 분배하는 서버 측의 구성요소로는 보안패치 분배서버, 보안패치 DB, 보안패치 매니저가 있으며 보안패치를 분배받는 클라이언트 측 구성요소로는 보안패치 클라이언트, 보안패치 에이전트가 있다.



〔그림 1〕 전체 시스템 구성도

- 패치 분배 서버
  :클라이언트와 보안패치 프로파일에 기반한 분배 메커니즘을 사용하여 실제 패치 분배 과정을 수행하는 구성요소. 보안패치 DB로부터 보안패치 클라이언트에게 필요한 해당 보안패치를 검색하며, 보안패치 클라이언트에 대한 인증과 기밀성 보장을 위한 키교환, 무결성 제공을 위한 보안패치에 대한 체크섬 서명 등의 기능을 수행.
- 보안패치 DB
  :보안 도메인 내에 구성되어 있는 클라이언트에 필요한 보안패치 파일 및 관련정보를 저장. 보안패치 분배과정에서 보안패치 분배서버의 요청에 의해 보안패치를 제공.
- 보안패치 매니저
  :보안패치 분배서버, 보안패치 DB 그리고 보안패치 프로파일에 대한 관리 기능을 수행하며 웹기반의 사용자 인터페이스를 제공.
- 보안패치 클라이언트
  :패치 분배서버와 보안패치 프로파일에 기반한 분배 메커니즘을 사용하여 실제 보안패치 분배 과정을 수행하는 구성요소, 패치분배 서버에 대한 인증과 기밀성 보장을 위한 키교환, 무결성 제공을 위한 패치에 대한 체크섬 검증 등의 기능을 수행.
- 보안패치 에이전트
  :보안패치 클라이언트와 보안패치 프로파일 정보에 대한 관리 기능을 수행하며 보안패치 에이전트는 웹기반의 사용자 인터페이스를 제공.

또한 앞서 열거한 것과 같은 각 구성요소들의 수행 기능은 다음의 [표 2]과 같이 정리된다. 또한 [그림 3]에는 제안하는 시스템에 대한 자동분배 과정에 대한 전체흐름도가 나타나있다.

〔표 2〕 보안기능 요약표

기능명	세부기능명	세부기능 설명
SP1:프로파일 정의	SP_PF_G_01	전역 보안패치 프로파일 정의
	SP_PF_C_02	클라이언트 패치 프로파일 정의
SP2:DB정의	SP_DB_01	보안패치 DB 구성 정의
SP3:자동분배 기능 정의	SP_AD_01	초기화 과정 자동분배
	SP_AD_02	업데이트 과정 자동분배
	SP_AD_03	유저시스템 변화에 의한 자동분배
SP4:자동설치 기능 정의	SP_AI_01	보안패치 자동설치 기능
SP5:사용자 인증 기능 정의	SP_UA_01	보안패치 사용자 인증 기능
SP6:기밀성 기능 정의	SP_SEC_01	보안패치 기밀성 기능
SP7:무결성 기능 정의	SP_INT_01	보안패치 무결성 기능
SP8:매니저 정의	SP_MAN_01	보안패치 서버 관리 기능
SP9:에이전트 정의	SP_AGE_01	보안패치 클라이언트 관리 기능

3.2 보안패치 자동 분배 메커니즘

본 논문에서 제안하는 안전한 보안패치 분배 구조에서는, 각 클라이언트들의 보안패치 프로파일과 보안 도메인내의 구성원 정보를 관리하는 글로벌 보안

패치 프로파일을 사용하여, 도메인내의 각 구성원들에게 패치를 자동으로 분배해준다.

먼저 글로벌 보안패치 프로파일은 보안패치 분배서버가 보안 도메인 안에서 자신이 관리하는 클라이언트의 기본 사항을 설정하고 관리하기 위해 사용하며, 전체 클라이언트의 수, 호스트 이름, 호스트 IP주소, 호스트 운영체제 종류, 호스트 운영체제 버전과 같은 정보를 가지고 있다. 또한 클라이언트 보안패치 프로파일은 각 보안패치 클라이언트가 자신의 보안패치 정보를 설정하기 위해 사용하며, 클라이언트 구성환경 변경 플래그, 클라이언트 시스템 이름, 사용자 ID/PW, 마지막 보안패치 설치 날짜/시간, 설치된 패치들의 개수, 설치된 패치 정보 등을 포함하고 있다.

위와 같은 보안패치 프로파일을 사용한 보안패치 자동 분배 메커니즘은 다음의 [표 3, 4, 5]와 같은 방법으로 동작한다. 즉, 글로벌 보안패치 프로파일과 클라이언트 보안패치 프로파일들을 사용하는 [표 3, 4, 5]의 분배방안을 통하여 관리자와 클라이언트들은 부가적인 작업 없이 패치 프로파일들을 참조하여 자동화된 분배가 가능하다.

다음의 [표 3]의 보안패치 초기화 메커니즘에서는, 안전한 보안패치 분배 구조의 초기화 과정에서, 분배서버는 보안패치 클라이언트에게 클라이언트 보안패치 프로파일을 제공받고 또한 글로벌 보안패치 프로파일을 업데이트 함으로써 초기화 과정에서 보안패치를 자동으로 분배하는 방안을 서술하고 있다.

〔표 3〕 보안패치 초기화 메커니즘

1. 보안패치 클라이언트가 보안패치 분배서버에게 보안패치 분배를 요청
2. 보안패치 클라이언트와 보안패치 분배서버간의 상호 인증과정을 수행
3. 보안패치 클라이언트와 보안패치 분배서버간의 기밀통신을 위한 키 교환
4. 보안패치 분배서버는 보안패치 분배를 요청한 클라이언트의 정보를 글로벌 보안패치 프로파일에서 참조
5. 보안패치 분배서버는 현재 클라이언트에게 필요한 보안패치 분배를 위해 클라이언트의 보안패치 프로파일을 요청
6. 보안패치 분배서버는 클라이언트 보안패치 프로파일을 참조하여 해당 보안패치 분배
7. 보안패치 클라이언트는 분배 받은 보안패치를 설치
8. 클라이언트 보안패치 프로파일에 보안패치 설치를 통한 패치 설치 변경 사항을 적용 후 보안패치 분배서버에게 전송
9. 보안패치 분배서버는 전송 받은 클라이언트 보안패치 프로파일을 업데이트

다음의 [표 4]의 보안패치 Push 메커니즘에서는 새로운 보안패치가 보안패치 DB에 업데이트 되는 경우에, 글로벌 보안패치 프로파일에 등록된 보안 도메인 내의 클라이언트 중 각 클라이언트 보안패치 프로파일을 참조하여 해당 클라이언트에게 보안패치를 자동으로 분배하는 방안을 서술하고 있다.

〔표 4〕 보안패치 Push 메커니즘(By Manager Request)

1. 새로운 보안패치가 보안패치 DB에 업데이트
2. 보안패치 분배서버는 글로벌 보안패치 프로파일에 등록된 클라이어트들의 클라이언트 보안패치 프로파일을 참조하여 보안패치 DB에 업데이트된 보안패치가 필요한 클라이언트 선별
3. 패치 분배서버는 선별된 클라이언트에게 패치 분배
4. 보안패치 클라이언트는 분배 받은 보안패치를 설치
5. 클라이언트 보안패치 프로파일에 패치 설치 변경 사항을 적용 후 보안패치 분배서버에게 전송
6. 보안패치 분배서버는 전송 받은 클라이언트 보안패치 프로파일을 업데이트

다음의 [표 5] 보안패치 Pull 메커니즘에서는 클라이언트 시스템에 새로운 설정이 적용되어 보안패치 프로파일이 수정되는 경우 수정된 클라이언트 보안패치 프로파일을 보안패치 분배서버에게 업데이트 하도록 요청하여 보안패치 분배서버가 업데이트 된 클라이언트 보안패치 프로파일을 기반으로 보안패치들을 자동 분배하는 방안을 서술하고 있다.

〔표 5〕 보안패치 Pull 메커니즘(By Client Request)

1. 새로운 프로그램의 설치 등으로 인한 클라이언트 시스템의 구성 변경 발생
2. 클라이언트 보안패치 프로파일에 새로운 프로그램 설치 후 변경 사항을 적용
3. 보안패치 클라이언트는 보안패치 분배서버에 보안패치 프로파일을 전송하고 보안패치 분배서버는 전송 받은 클라이언트 보안패치 프로파일을 업데이트
4. 업데이트 된 클라이언트 보안패치 프로파일을 참조하여 보안패치 분배서버는 클라이언트에게 필요한 보안패치를 선별하여 분배
5. 보안패치 클라이언트는 분배받은 보안패치를 설치
6. 클라이언트 보안패치 프로파일에 패치 설치 변경 사항을 적용 후 보안패치 분배서버에게 전송
7. 패치 분배서버는 받은 보안패치 프로파일을 업데이트

Ⅳ. 보안패치 분배 구조의 구현

본 논문에서 제안하는 보안패치 분배 구조는 Redhat linux 7.3 기반에서 자바(JDK 1.3.1_05)를 사용하

여 구현되었다. 보안패치 분배서버는 자바 멀티쓰레딩 서버로 동작하도록 구성되었으며, 보안패치 클라이언트는 역시 자바 클라이언트로 구현되었다. 또한 보안패치 매니저와 에이전트는 아파치 웹서버를 사용하며 PHP(ver 4.1.2-7)로 구현되었고 보안패치 DB는 Mysql(ver 11.28)을 사용하였다. 암호화 관련 구현은 JCA를 따르는 JCSI(ver 1.0b)를 사용하였다. 그러면 각 구성요소들에서 사용하는 세부 기능의 구현에 대하여 알아보겠다.

4.1 사용자 인증 스킴

보안패치 자동분배 프레임워크의 클라이언트는 서버의 인증서를 기반으로한 클라이언트·서버 상호인증 기능을 수행한다. 보안패치 자동분배 프레임워크의 사용자 인증 과정이 수행되기 전에 아래와 같은 두 가지 사항을 만족해야 한다.

- 인증에 사용할 보안패치 서버의 인증서를 미리 발급(OFF-Line 분배 권장)
- 사용자들은 서버의 인증서를 발급받는 과정에서 사용자 ID와 PW를 등록

인증 메커니즘에 사용된 프로토콜은 다음과 같으며 본 프로토콜에 나오는 기호는 [표 6]에 설명되어 있다.

[표 6] 인증 프로토콜 기호 설명

기호	설명
SP_Server	보안패치 서버
SP_Client	보안패치 클라이언트
T	타임스탬프
R	랜덤 넘버
Epub/Dpri	공개키암호화/개인키복호화
ID/PW	사용자 식별자/비밀번호

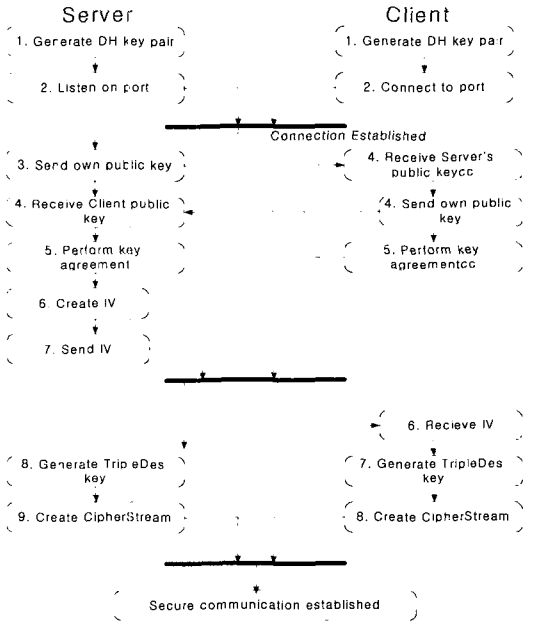
SP_Client->SP_Server : Send Msg_Distribution
SP_Server->SP_Client : Send Msg_Distribution|T
SP_Client : Generating R
SP_Client->SP_Server : $E_{pub}[R|T]$
SP_Server : Verify $D_{pri}[E_{pub}[R|T]]$
SP_Server->SP_Client : $E_R+1[T]$
SP_Client : Verify $D_R+1[E_R+1[T]]$
SP_Client->SP_Server : ID | $E_{pub}[PW]$
SP_Server : Verify $D_{pri}[E_{pub}[PW]]$
SP_Serve->SP_Client : Send Msg_Auth_success

먼저 클라이언트는 패치분배를 서버에게 요청하고, 클라이언트의 요청을 받은 서버는 응답 메시지와 함께 T를 전달한다. 클라이언트는 R을 생성하고 생성한 R과 T를 서버의 공개키로 암호화하여 서버에게 전송하고, 서버는 R을 복호화한 후 다시 R+1으로 T를 암호화하여 전송한다. 클라이언트는 이 값을 확인함으로써 서버에 대한 인증을 수행한다. 이후 클라이언트는 자신의 ID/PW를 전송하여 자신에 대한 인증과정을 수행한다. 이러한 서버/클라이언트 상호간의 인증과정이 수행된 후 본 분배 과정이 시작된다.

4.2 기밀성 제공 스킴

안전한 보안패치 분배 구조에서 보안패치 서버와 보안패치 클라이언트간의 상호인증 과정이 수행된 후에 SKIP(Simple Key management for Internet Protocol) 모듈러스 기반의 DH 키교환을 통해 기밀 통신채널 확립에 필요한 비밀키를 공유하게 된다. 이렇게 공유된 비밀키를 사용하여 분배서버와 클라이언트간의 패치분배 과정에서 패치파일에 대한 암호화 서비스를 제공한다.

이러한 SKIP 모듈러스 기반의 DH 키교환 과정은 그림에서 보이는 바와 같다.[23~24] 여기서 SKIP 모듈러스란 SKIP에서 정의하고 있는 여러 암호화 알고리



[그림 2] SKIP 모듈러스 기반의 DH 키분배

즘의 파라미터 중 DH 키분배에 사용되는 파라미터를 가리키며 본 논문에서는 512, 1024, 2048bits 비밀키 중 1024bits의 키를 생성하기 위해 1024bits SKIP 스펙을 사용하였다.

4.3 패치파일 인증 및 무결성 제공 스킴

패치파일에 대한 인증 및 무결성 서비스를 제공하기 위하여 패치서버는 클라이언트에게 제공하는 패치파일에 대하여 먼저 SHA 알고리즘을 사용하여 160bits 메시지 다이제스트 값을 계산한다. 이렇게 생성된 패치 파일에 대한 메시지 다이제스트 값은 패치 파일에 대한 무결성을 제공하며 다시 이 계산값에 대한 패치서버의 서명을 첨부함으로써 패치파일에 대한 인증 기능을 제공 할 수 있다. 다음에 설명되어 있는 프로토콜은 패치파일에 대한 인증 및 무결성을 제공하기 위한 것이며 다음과 같다.

먼저 클라이언트는 패치서버로부터 DH 키분배를 통해 생성된 비밀키로 암호화 된 값을 전송 받는다(아래 프로토콜의 1번 과정). 그 후 클라이언트는 먼저 자신과 패치서버가 공유하고 있는 비밀키를 사용하여 암호화된 값을 복호화한다(아래 프로토콜의 2번 과정). 복호화가 성공하면 다시 패치서버의 공개키로 패치서버의 서명값을 검증함으로써 패치파일에 대한 인증기능을 수행한다(아래 프로토콜의 2.1번 과정). 패치파일 인증과정이 성공하면 다시 패치파일에 대한 SHA 메시지 다이제스트 계산값을 검증한다(아래 프로토콜의 2.2번 과정). 이러한 모든 과정이 모두 성공하면 클라이언트는 패치서버에게 전송 받은 패치가 정상임을 알리며 자신의 시스템에 설치하기 위한 과정에 들어간다(아래의 3번 과정).

1: SP_Server->SP_Client : $E_k[P|S_k[H(P)]]$
2: SP_Client : $D_k[E_k[P|S_k[H(P)]]]$
2.1: : P, V[S[H(P)]]
2.2: : P, C(P)

〔표 7〕 무결성 프로토콜 기호 설명

기호	설 명
P	보안패치 파일
S	패치 서버의 서명 생성
V	서명값에 대한 검증 값 생성
H(P)	패치파일에 대한 메시지다이제스트
Ek/Dk	비밀키를 사용하여 암호화/복호화

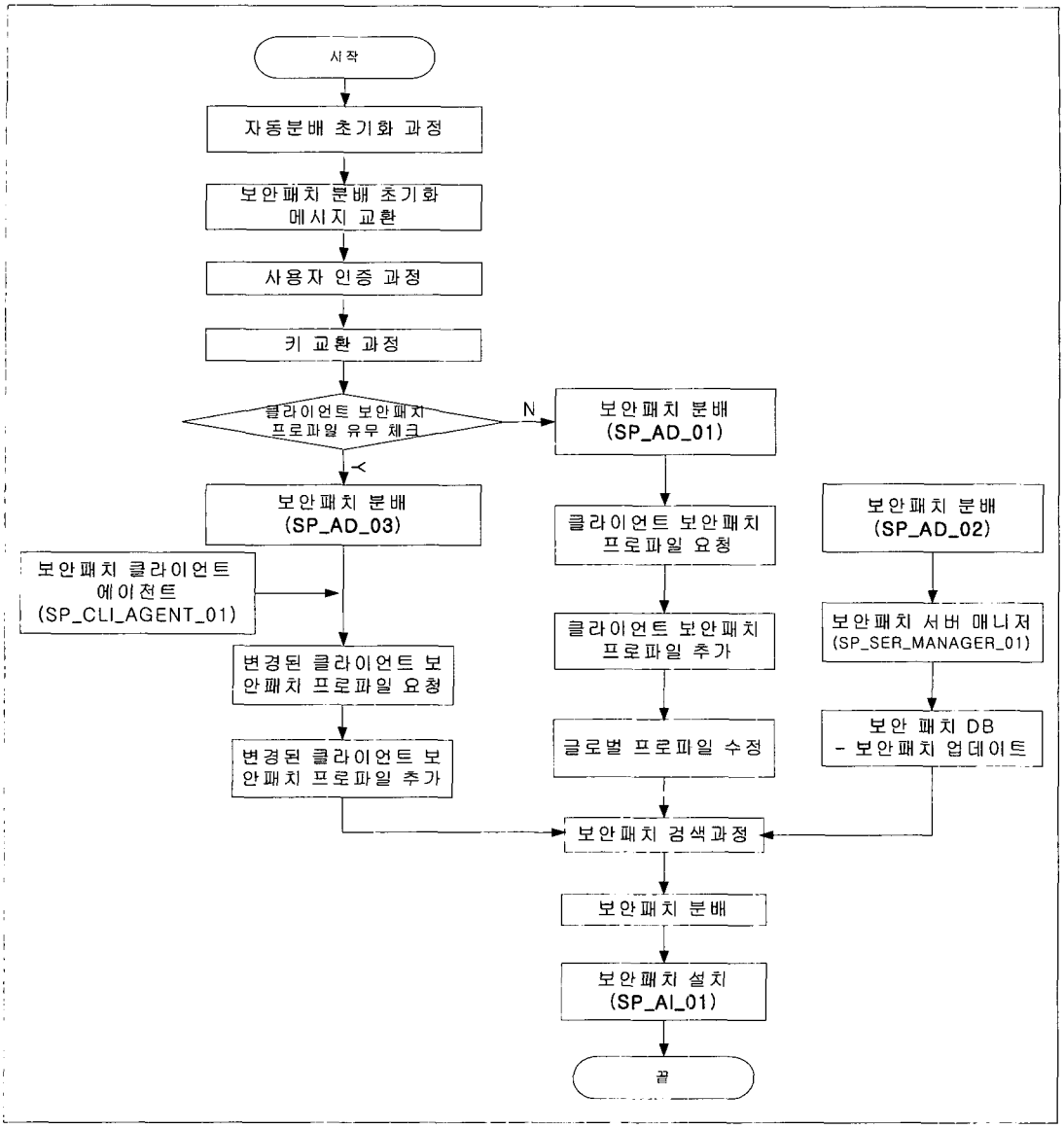
3: SP_Client->SP_Server : if Patch is verified,
Send Msg_Patch_verified

다음의 [표 7]은 위의 패치파일 인증 및 무결성 제공 프로토콜에 대한 기호 설명이다.

4.4 자동 분배 스킴

제안 시스템에서는 전역 보안패치 프로파일과 클라이언트 보안패치 프로파일을 사용하여 보안패치들을 자동으로 분배한다. 우선 클라이언트 보안패치 프로파일은 앞서 서술된 것과 같이 클라이언트에 설치된 보안패치 정보를 가지고 있으며 이러한 보안패치 정보들은 클라이언트가 최초로 분배 서버에 인증되는 과정에서 분배 서버에 전송된다. 이때 인증된 클라이언트들 정보와 그 클라이언트의 시스템 정보를 전역 보안패치 프로파일에 등록한다. 그 후 전역 보안패치 프로파일을 참조하여 보안패치 DB를 검색하여 필요한 보안패치를 분배하는 것이다([그림 3]의 SP_AD_01 참고). 그리고 보안패치 DB에 새로운 보안패치가 추가되는 경우에, 역시 먼저 전역 보안패치 프로파일을 참조하여 등록된 클라이언트의 보안패치 프로파일을 참조하게 되고 이 과정에서 현재 새롭게 추가된 보안패치가 필요한 클라이언트를 찾아내어 분배한다.([그림 3]의 SP_AD_02 참고), 마지막으로 클라이언트 시스템에 새로운 소프트웨어가 설치되는 경우 현재 소프트웨어의 설치 정보를 클라이언트 보안패치 프로파일에 업데이트하게 되면, 수정된 이 클라이언트 보안패치 프로파일을 분배서버에게 전송하여 분배서버가 보관하고 있는 클라이언트 보안패치 프로파일을 업데이트 하게 되고, 이에 맞는 보안패치를 보안패치 DB에서 검색하게 된다([그림 3]의 SP_AD_03 참고). 그러므로 이와 같은 클라이언트 시스템의 패치 설치 정보를 관리하는 프로파일의 지속적인 갱신을 통하여 서버는 부가적인 관리자의 노력 없이 새로운 보안패치가 발표될 때 신속 정확하게 분배할 수 있다.

자동 분배과정에서 DB 검색과정은 현재 보안패치 프로파일의 필드 중 운영체제 종류와 각 운영체제의 버전에 따라 구분되지만, 본 구현에서는 단일 운영체제(리눅스)만을 고려하였기 때문에 큰 의미가 없다. 하지만 향후에는 대상 환경의 보안 정책에 따른 그룹화 기능을 추가하여 보다 효율적 패치 DB검색 기능을 포함할 것이다.

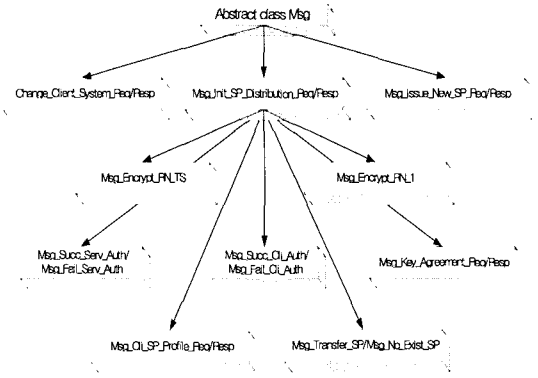


〔그림 3〕 보안패치 분배 구조 전체 흐름도

4.5 보안패치 프로파일 스킴

보안패치 자동분배 시스템에서 사용되는 보안패치 프로파일은 전역 보안패치 프로파일과 클라이언트 보안패치 프로파일로 구성되며 각 프로파일의 기능과 구성요소는 다음과 같다.

- 전역 보안패치 프로파일
  :보안패치 자동분배 시스템의 분배 서버가 자신이 관리하는 클라이언트의 기본 사항을 설정하기 위해 사용하며 전체 클라이언트 사용자 수, 클라이언트 시스템 이름, 클라이언트 시스템의 IP 주소, 클라이언트 시스템의 운영체제, 운영체제 버전으로 구성.
- 클라이언트 보안패치 프로파일
  :보안패치 자동분배 시스템의 클라이언트가 자신의 보안패치 정보를 설정하기 위해 사용하며 클라이언트 시스템 이름, 클라이언트 사용자 ID/PW, 마지막 보안패치 설치 날짜/시간, 설치된 패키지 개수, 설치된 패키지 목록으로 구성.



〔그림 4〕 메시지 계층 구조도

4.6 분배 메시지 핸들링 스킴

본 논문에서 제안하는 안전한 보안패치 자동분배 구조는 자바언어로 구현하였으며 자바의 객체 지향적인 특성에 따라 분배 구조에 사용되는 모든 구성요소들간 이벤트들은 모두 추상 메시지 클래스에서 앞서 서술된 분배 메커니즘들에 대한 세 가지 계열의 메시지로 구분되며 다시 각 메시지들은 각 메커니즘에 필요한 인자들을 포함하여 이벤트를 수행하도록 구성되었다. 또한 자바의 플랫폼 독립적인 실행 특성에 의해 현재 리눅스 기반의 분배 구조는 추후 부가적인 수정 없이 상이한 플랫폼으로의 적용이 가능하다.

〔표 8〕 메시지 구성 및 기능

Abstract class Msg : 보안패치 자동분배 구조의 추상 메시지 클래스 Msg_Init_SP_Distribution_Req/Resp : 클라이언트의 보안패치 초기화 요청/응답 메시지 Msg_Encrypt_RN_TS : 클라이언트의 랜덤 넘버와 수신한 타임스탬프를 서버의 공개키로 암호화한 서버인증 요청 메시지 Msg_Encrypt_RN_1 : 클라이언트로부터의 메시지로부터 랜덤 넘버를 복호화하고 랜덤 넘버를 증가시킨 값으로 다시 타임스탬프를 암호화하여 전송한 서버 인증 요청 메시지 Msg_Succ_Serv_Auth/Msg_Fail_Serv_Auth : 서버인증이 성공/실패 전송 메시지 Msg_Succ_Cli_Auth/Msg_Fail_Cli_Auth : 클라이언트인증이 성공/실패 전송 메시지 Msg_Key_Agreement_Req/Resp : 키교환 요청/응답 메시지 Msg_Cli_SP_Profile_Req/Resp : 보안패치 프로파일 요청/응답 메시지 Msg_Transfer_SP/Msg_No_Exist_SP : 보안패치 전송 중/발견 못함 알림 메시지 Msg_Issue_New_SP_Req/Resp : 새로운 보안패치 추가/응답 메시지 Change_Client_System_Req/Resp : 클라이언트 시스템 구성 변경 알림 요청/응답 메시지

[표 8]에서는 추상 메시지 클래스를 상속한 각 메시지들에 대한 설명을 하고 있으며 [그림 4]에서는 각 메시지들의 계층 구성도와 그 메시지들을 사용하여 필요한 이벤트를 수행하는 분배 메커니즘과의 상관관계를 보여주고 있다. 다음의 [표 8]은 각 메시지들에 대한 처리 이벤트 및 그 속성에 대하여 설명한다.

V. 제안시스템 구현 결과 및 검증예제

5장에서는 본 논문에서 제안한 시스템의 구현 결과를 소개하고 구현된 결과를 검증하기 위한 테스트를 수행한다. 결과 테스트는 앞의 3.2에서 서술한것과 같이 패치 분배가 일어나는 세 가지 경우 중 시스템 초기화 과정에서의 분배과정을 통해 실시된다.

먼저 제안 시스템 구현 결과는 다음과 같다. 전체 시스템의 관리를 위한 사용자 인터페이스를 가지는 보안패치 매니저와 에이전트는 아파치(Apache) 서버(ver 1.3.23-11)와 php(ver 4.1.2-7)를 통하여 구현되어 원격에서도 웹기반으로 편리하게 처리할 수 있게 구현되었다.

보안패치 매니저는 보안패치 서버와 보안패치 DB를 관리하며 다음과 같은 기능을 제공한다.

- 보안패치 서버의 기본 정보를 제공
  :운영체제 종류, 버전, 개발 언어 정보, 보안패치 DB 기본 정보, 보안패치 자동분배 프레임워크의 클라이언트 정보 제공
- 전역 보안패치 프로파일 정보
  :등록된 호스트에 대한 클라이언트 보안패치 프로파일 정보 또한 리스트
- 보안패치 정보 제공
  :현재 보안패치 DB에 저장된 보안패치 관련 정보 리스트
- 보안패치 DB에 새로운 보안패치 업데이트 제공
  :패치 추가, 제거, 업데이트 메시지 전송 기능

보안패치 에이전트는 클라이언트에 설치된 보안패치 정보를 관리하며 다음과 같은 기능을 제공한다.

- 보안패치 클라이언트의 기본 정보를 제공
- 개발 정보, 클라이언트 시스템의 보안패치 프로파일 정보

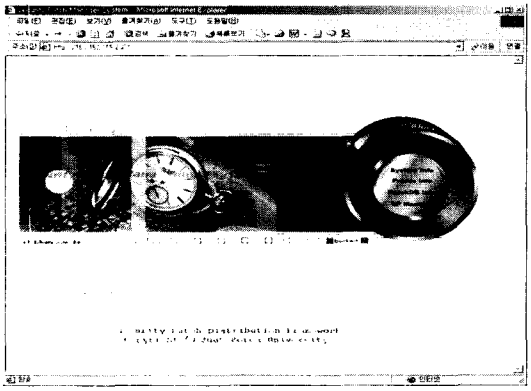
- 보안패치 프로파일 관리 기능 제공
  :새로운 프로그램 추가, 삭제, 보안패치 클라이언트 재실행

보안패치 DB는 Mysql(ver 11.18)을 사용하여 구현되었으며 구성요소는 다음과 같다.

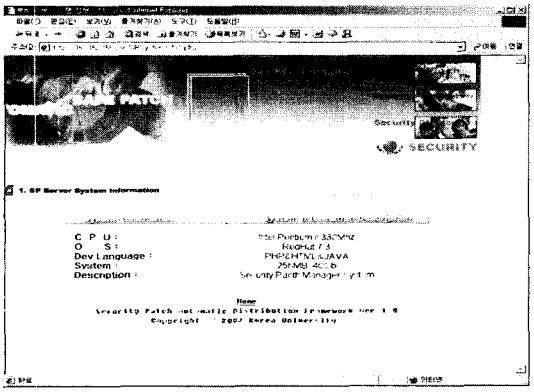
- 보안패치 대상 파일 이름/버전, 보안패치 파일, 보안패치 체크섬 서명값, 보안패치 관리자, 마지막 업데이트 날짜, CVEs

다음에서는 실제 보안패치 자동분배 시스템의 각 구성요소의 기능을 알아본다. 보안패치 자동분배 매니저는 [그림 5]와 같은 4가지 기능을 수행한다.

- System Info : 분배 서버 시스템의 정보 표시[그림 6]
- Profile Info : 글로벌 보안패치 프로파일 정보를 표시하며 해당 호스트를 선택하면 해당 호스트의 클라이언트 보안패치 프로파일 정보 표시 [그림 7], [그림 8]

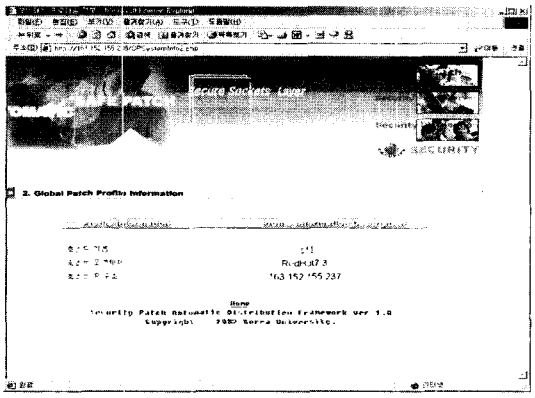


〔그림 5〕 보안패치 자동분배 매니저

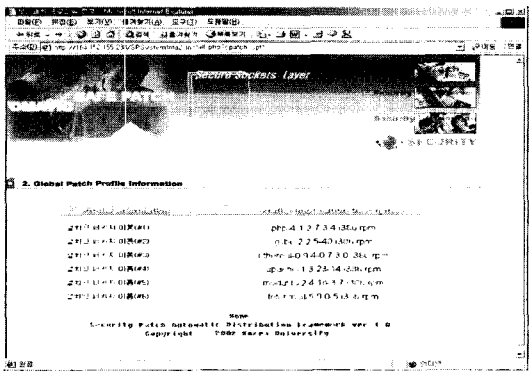


〔그림 6〕 보안패치 분배서버 시스템 정보

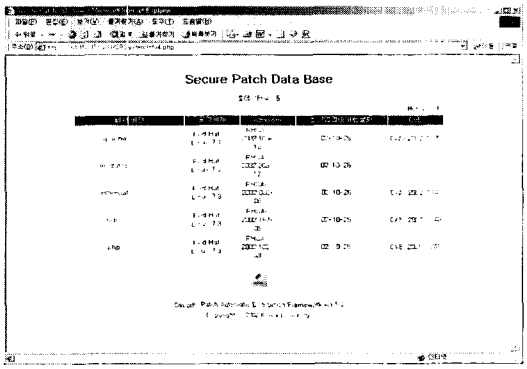
- PatchDB Info : DB에 저장된 보안패치 정보 표시 [그림 9]
- SP Manager : 보안패치 프로파일과 보안패치 DB를 관리 [그림 10], 새로 발표된 보안패치를 추가하는 기능 및 수정하는 기능 그리고 보안패치 서버에게 알려주는 기능을 수행 [그림 11]



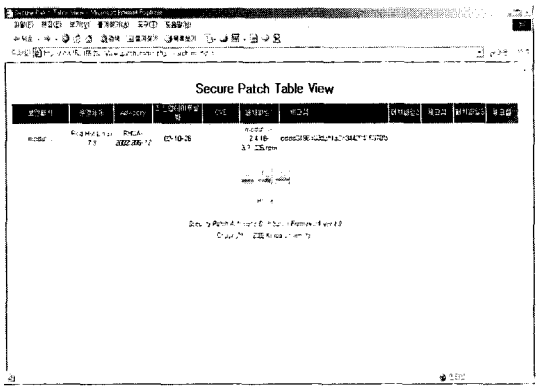
〔그림 7〕 글로벌 보안패치 프로파일 정보 표시



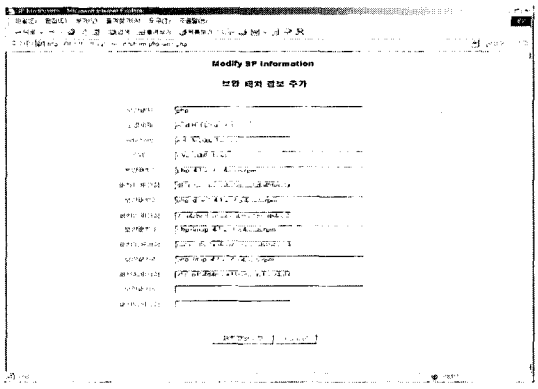
〔그림 8〕 글로벌 프로파일에 등록된 사용자 보안패치 프로파일 정보



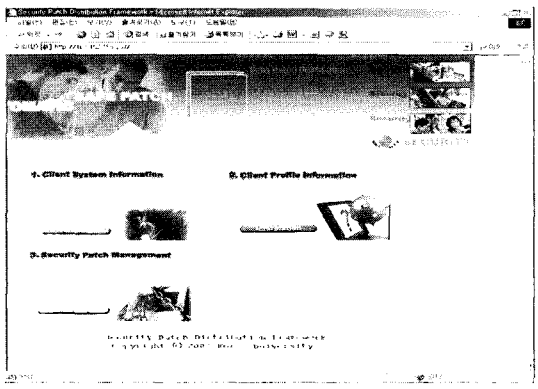
〔그림 9〕 보안패치 DB 정보 표시



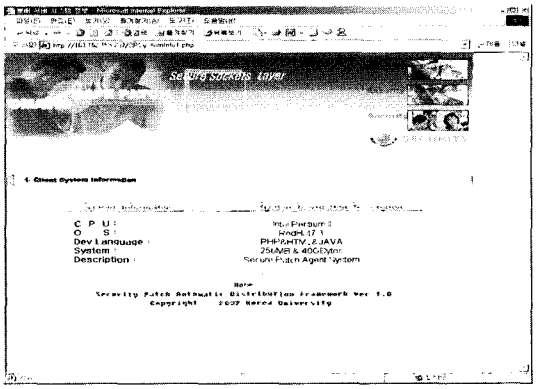
〔그림 10〕 보안패치 DB 세부 정보 표시



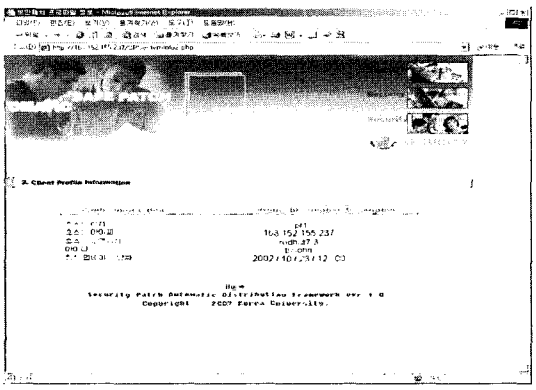
〔그림 11〕 보안패치 DB 수정 정보 표시



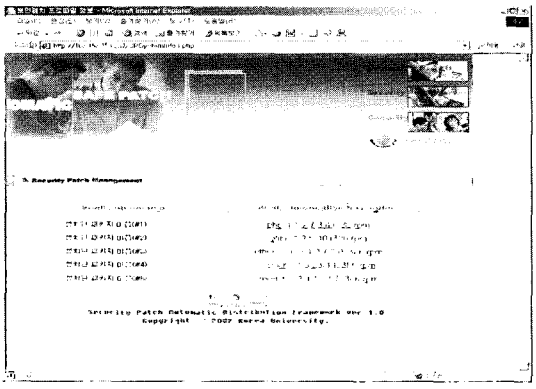
〔그림 12〕 보안패치 에이전트



〔그림 13〕 보안패치 에이전트 시스템 정보 표시



〔그림 14〕 보안패치 사용자 프로파일 정보 표시



〔그림 15〕 보안패치 프로파일의 인스톨 정보 표시

보안패치 자동분배 에이전트는 [그림 12]와 같은 3가지 기능을 수행한다.

- System Info : 분배 클라이언트 시스템의 정보를 표시[그림 13]
- Profile Info : 글로벌 보안패치 프로파일 정보를 표시하며 해당 호스트를 선택하면 클라이언트 보안패치 프로파일 정보를 보여준다.[그림 14]
- Security Patch Management : 보안패치 프로파일의 보안패치 정보를 표시하며, 수정, 추가 메뉴를 제공.[그림 15]

지금까지 본 논문에서 제안된 안전한 보안패치 분배 시스템의 구현 결과를 알아보았다. 또한 이렇게 구현된 시스템에 대한 테스트를 위해 앞서 서술된 3.2와 같은 세가지 분배 상황을 가정할 수 있으며,

이러한 세가지 분배 상황중에서도 본 장에서는 보안패치 자동분배 시스템의 테스트를 위해 시스템 초기화 과정에서의 분배 과정 예제를 수행한다. 이 경우 클라이언트는 먼저 최초 등록 과정에서 자신의 보안패치 프로파일을 서버에게 등록하며, 이때 클라이언트 보안패치 프로파일을 받은 보안패치 분배 서버는 보안패치 DB의 패치 파일의 목록과 현재 클라이언트로부터 전송 받은 보안패치 프로파일을 검색하여 해당 패치가 존재하는 경우 패치를 분배하게 된다.

우선 보안패치 자동분배 검증을 위한 준비 과정으로 다음과 같은 상황을 가정한다.

- 가정 : 두 클라이언트를 SPF1, SPF2로 가정
- 보안패치 매니저에서 글로벌 보안패치 프로파일에 등록된 사용자 확인(초기화 상태이므로 등록된 사용자 없음, [그림 16])
- 보안패치 매니저에서 보안패치 DB에 등록된 보안패치 확인. [그림 17]
  - apache-1.3.23-14.i386.rpm
  - ethereal-0.9.4-0.7.3.0.i386.rpm
  - glibc-2.2.5-40.i386.rpm
  - php-4.1.2-7.3.4.i386.rpm
  - modutils-2.4.18-3.i386.rpm
- 각 클라이언트의 보안패치 에이전트에서 자신의 보안패치 프로파일에 등록된 인스톨 패치 정보 확인. [그림 18], [그림 19]

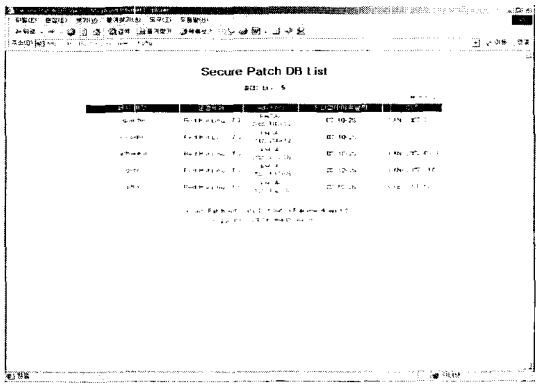
  SPF1(테스트 클라이언트 1)
  - installed_pkg_1=php-4.1.2-7.3.4.i386.rpm
  - installed_pkg_2=glibc-2.2.5-40.i386.rpm
  - installed_pkg_3=ethereal-0.9.4-0.7.3.0.i386.rpm
  - installed_pkg_4=apache-1.3.23-14.i386.rpm
  - installed_pkg_5=modutils-2.4.18-3.7.i386.rpm
  - installed_pkg_6=fetchmail-5.9.0-5.i386.rpm

  SPF2(테스트 클라이언트 2)
  - installed_pkg_1=php-4.1.2-7.3.4.i386.rpm
  - installed_pkg_2=glibc-2.2.5-40.i386.rpm
  - installed_pkg_3=ethereal-0.9.4-0.7.3.0.i386.rpm
  - installed_pkg_4=apache-1.3.23-14.i386.rpm
  - installed_pkg_5=fetchmail-5.9.0-5.i386.rpm

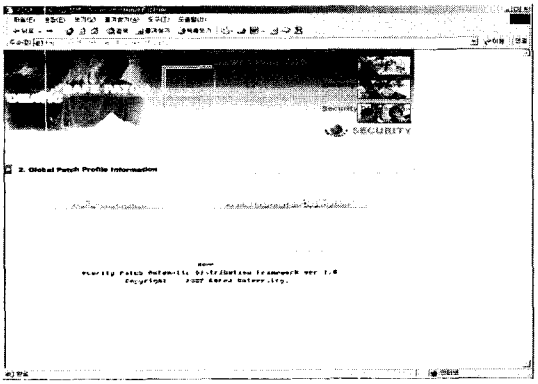
지금까지 보안패치 자동분배 검증을 위한 준비과정을 수행하였으며 이후에는 초기화과정 분배 시나리오를 수행한다. 분배 시나리오는 다음과 같이 진행된다.

1단계 : 보안패치 자동분배 서버프로그램 및 SPF1 클라이언트 프로그램 구동. [그림 20]

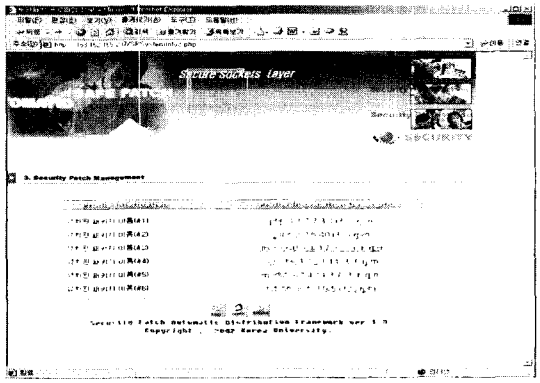
2단계 : 콘솔 화면에서 보안패치 자동분배 프레임워크의 구동 메시지 확인(초기 접속 메시지, C/S 상



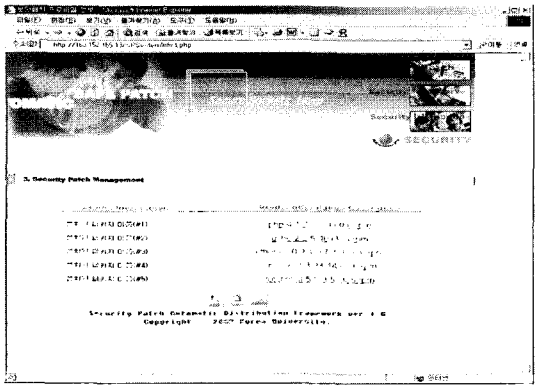
〔그림 17〕 보안패치 매니저에서 패치DB 확인



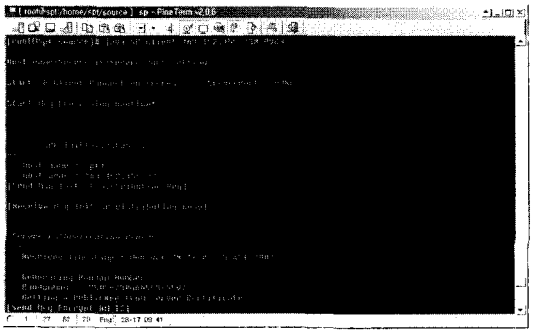
〔그림 16〕 보안패치 매니저에서 글로벌프로파일 확인



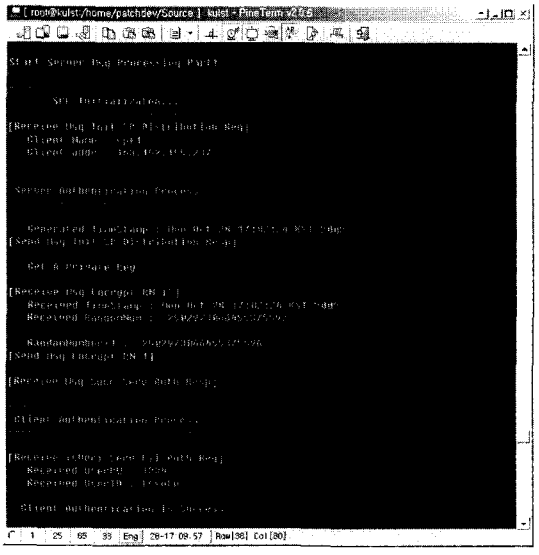
〔그림 18〕 에이전트에서 사용자 프로파일 확인(spf1)



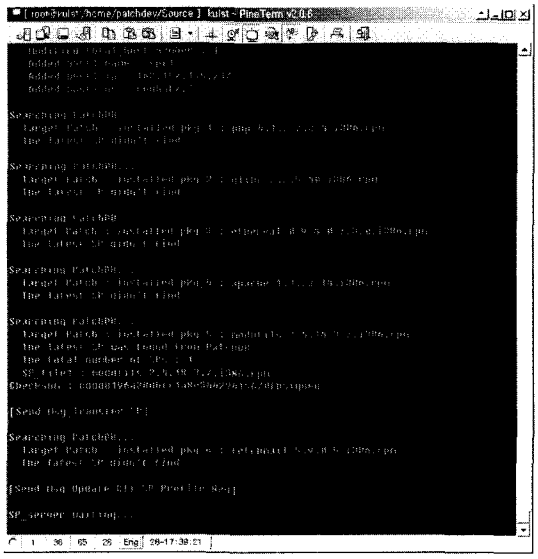
〔그림 19〕 에이전트에서 사용자 프로파일 확인(spf2)



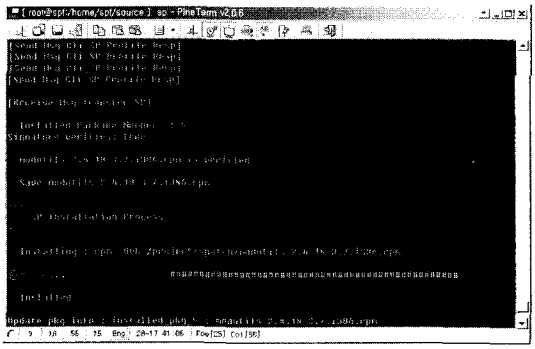
〔그림 20〕 보안패치 서버 및 클라이언트 구동



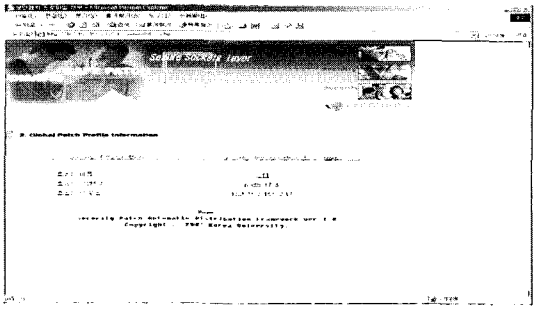
〔그림 21〕 인증 및 키교환 메시지 확인



〔그림 23〕 4단계 : 사용자 등록, 패치 검색, 분배 과정



〔그림 24〕 사용자는 패치를 분배받고 체크섬 검사 후 설치 과정을 수행

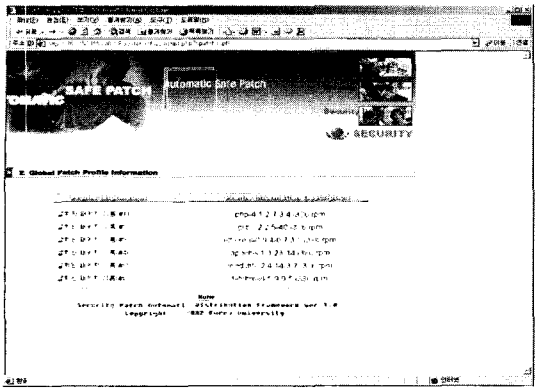


〔그림 25〕 글로벌프로파일에 등록된 사용자 확인

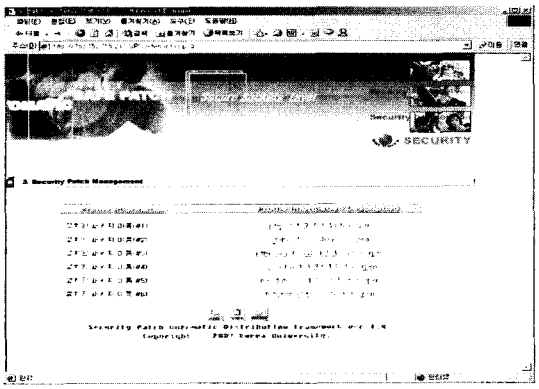
상호 인증 메시지, DH 키 교환 메시지, 자동 분배 과정 메시지, 자동 설치 과정 메시지, [그림 21, 22, 23])

3단계 : SPF1의 콘솔에서 보안패치 전송 확인, 설치 확인[그림 24]

4단계 : 보안패치 매니저의 프로파일 정보 확인(SPF1 등록 확인 [그림 25], SPF1의 보안패치 프로파일 정보 확인. [그림 26])



〔그림 26〕 글로벌 프로파일에 등록된 사용자(spf1)의 인스톨 패치 정보 확인



〔그림 27〕 사용자 보안패치 프로파일에 등록된 업데이트된 인스톨 패치 정보 확인

5단계 : SPF1 보안패치 에이전트의 보안패치 프로파일 업데이트 정보 확인[그림 27]

Ⅳ. 결 론

보안패치는 시스템의 소프트웨어나 운영체제의 문제점 중에서도 보안에 관련된 중요 부분의 안전한 관리를 위해서 필수적인 요소이다. 그러므로 안전하고 신속하게 보안패치를 분배받아 설치하는 것이 중요하며, 또한 분배 과정에서의 보안 역시 꼭 필요하다.

본 논문에서는 보안패치 분배에 있어서의 자동화, 기밀성, 무결성, 인증 요구사항을 충족시킬 수 있는 보안패치 분배구조를 제안하고 설계하였다. 설계된 사항의 구현은 자바를 사용하여 리눅스기반에서 구현되었으며 초기화 과정에서의 분배, 새로운 보안패치의 추가 후 분배, 클라이언트 시스템의 소프트웨어 설치 시 필요한 보안패치 분배의 세 가지 시나리오를 통해 구현된 보안패치 분배 구조를 테스트하였다.

추후에는 클라이언트 시스템에서의 설치 과정에서 발생할 수 있는 의존성에 따른 에러 발생 처리와 리눅스이외의 다양한 시스템으로의 적용 가능성에 대한 연구가 필요하다.

참 고 문 헌

[1] M. A. Bashar, "Low-threat security patches and tools", *1997 International Conference on Software Maintenance(ICSM '97)*, October, 1997.

[2] LLNL, "Secure Software Distribution Architecture : SafePatch", *Lawrence Livermore National Laboratory*, May, 1999.

[3] G. Caronni, M. Waldvogel, D. Sunand, and B. Plattner. "Efficient Security for Large and Dynamic Multicast Groups", *In Workshop on Enabling Technologies(WETICE 98)*, IEEE Comp Society Press, 1998.

[4] C. Kaufman, "DASS(Distributed Authentication Security Service)", *Network Working Group Request for Comments: 1507*, September, 1993.

[5] BindView TR, "Vulnerabilities in Operating-System Patch Distribution", *http://razor.bindview.com/*, 2000.

[6] BIRMAN, K. P., AND JOSEPH, T., "Exploiting virtual synchrony in distributed systems", *In 11th Annual Symosium on Operating Systems Principles*, pp. 123-138, November, 1987.

[7] BIRMAN, K. P., AND RENESSE, R. V. "Reliable Distributed Computing with the Isis Toolkit", *IEEE Computer Society*, March 1994.

[8] Mc DANIEL, P., PRAKASH, A., AND HONEYMAN, P., "Antigone: A flexible framework for secure group communication", *In Proceedings of the 8th USENIX security Symposium*, pp.99~114, August, 1999.

[9] Tim Polk. "Automated tools for testing computer system vulnerability", *Technical Report NIST SP800 -6, National Institute of Standards and Technology*, 1993.

[10] Cliff Stoll,. "Telling the goodguys: Disseminating information on security holes", *In Proceedings of the Fourth Aerospace Computer Security Conference*, pp.216~218, 1988.

[11] Joey Hess, "A comparison of the deb, rpm, tgz, slp, and pkg package formats", GNU, 1999
[12] BSDI/BSD/OS, https://www.bsdi.com/
[13] Caldera / OpenLinux, https://www.calderasystems.com /security
[14] Compaq/Tru64 UNIX, http://www.compaq.com/
[15] Debian GNU/Linux, http://www.debian.org/
[16] FreeBSD, http://www.freebsd.org/
[17] Hewlett-Packard / HP-UX, http://www.itrc.hp.com/ common/bin/doc.pl
[18] IBM/AIX, http://techsupport.services.ibm.com/
[19] Novell / Netware, https://support.novell.com/filefinder/
[20] SGI / IRIX, http://www.sgi.com/
[21] Slackware Linux, http://www.slackware.com/lists/
[22] Sun Microsystems / Solaris, http://sunsolve.sun.com/
[23] Jess Garms, Java Security, WROX, 2002
[24] SKIP, ISOC http://www.isoc.org/HMP/PAPER/244/ abst.html

〈著者紹介〉

손 태 식 (Tae-Shik Sohn)
2000년 2월 : 아주대학교 정보 및 컴퓨터 공학부 학사
2002년 2월 : 아주대학교 정보통신전문대학원 정보통신공학과 석사
2002년 3월 : 고려대학교 정보보호대학원 정보보호학과 박사과정
한국정보보호교육센터 강사,
ICU부설 정보통신교육원 강사
<관심분야> 시스템/네트워크 보안

서 정 우 (Jung-Woo Seo)
2002년 2월 : 호남대학교 정보통신 공학부 졸업(공학사)
2002년 3월~현재 : 고려대학교 정보보호대학원 석사과정 재학중
<관심분야> 시스템/네트워크 보안, 생체인식

문 종 섭 (Jong-Sub Moon)
1981년 2월 : 서울대학교 계산통계학과 학사
1983년 2월 : 서울대학교 계산통계학과 석사
1992년 2월 : Illinois Institute of Technology 박사
1993년~현재 : 고려대학교 전자 및 정보공학부 교수
고려대학교 정보보호대학원 겸임 교수
<관심분야> IDS, 신경망, 생체인식, 운영체제

서 정 택 (Jung-Taek Seo)
1999년 2월 : 충주대학교 컴퓨터공학과 졸업(공학사)
2001년 2월 : 아주대학교 대학원 컴퓨터공학과 졸업(공학석사)
2000년 11월~현재 : ETRI 부설 국가보안기술연구소 연구원
<관심분야> 정보전, 시스템/네트워크 보안, 취약점 분석 · 평가

임 을 규 (Eul-Gyu Im)
1992년 2월 : 서울대학교 컴퓨터공학과 졸업(공학사)
1994년 2월 : 서울대학교 대학원 컴퓨터공학과 졸업(공학석사)
2002년 2월 : Univ. of Southern California Computer Science Dept 박사 졸업
2000년~2002년 : WiseNut Inc
2002년 4월~현재 : ETRI 부설 국가보안기술연구소 선임연구원
<관심분야> 시스템 및 네트워크 보안, 취약점 분석 · 평가

이 철 원 (Cheol-Won Lee)
1987년 : 충남대학교 수학과 졸업(학사)
1989년 : 중앙대학교 대학원 전자계산학과(석사)
2001년 : 아주대학교 대학원 컴퓨터공학과 박사과정 수료
1989년~1996년 : 한국전자통신연구원 선임연구원
1996년~2000년 : 한국정보보호센터 선임연구원/통신모델링 과제책임자
2000년~현재 : ETRI 부설 국가보안기술연구소 팀장
<관심분야> 컴퓨터 및 네트워크 보안, 정보통신 기반보호, 정보보호시스템 평가기준