

인증서 인증 기반 통합관리 시스템

팀 명 : 섭두리 (서버 두리)

지도 교수 : 유 승 재 교수님

팀 장 : 박현민

팀 원 : 김우람

김재동

안지훈

윤조영

주좌훈

2016. 05

중부대학교 정보보호학과

목 차

1. 서론

- 1.1 연구배경 1
- 1.2 연구 목적 및 주제 선정 1

2. 관련연구

- 2.1 openssl 2
- 2.2 취약점 분석 시스템 3-4
- 2.3 인증서 인증 5-10

3. 본론

- 3.1 시스템 프로세스 11
- 3.1 인증서 생성 방법 11-14
- 3.2 인증서 로그인 방법 14-16
- 3.3 취약점 시스템 16-21

4. 결론 21-22

5. 첨부

- 5.1 웹 프로그램 소스 22-27
- 5.2 발표 자료 27-40
- 5.3 참고 문헌 40-41

1. 서론

1.1 연구배경

매해 정보보호학과의 졸업 작품에 빠지지 않는 서버 취약점 분석 시스템을 구성으로 기존에 서버 관리자 계정 관리가 가지고 있던 취약한 점을 조사해 인증서 인증이라는 인증기법을 추가하여 서버를 관리 하는 환경을 구축하였다.

우선 취약점 분석 시스템은 사용자가 로그아웃 한 시간을 배경으로 사용자가 없을 때 발생하는 취약점을 사용자가 로그인 한 후 바로 자신의 서버의 취약 항목들의 개수를 확인할 수 있고, 상세한 목록까지도 확인 가능하며, 바로 취약점을 줄여 서버의 보안 관리가 손쉬워 질 수 있겠다는 생각을 바탕으로 주요 정보 통신기반 시설 취약점 분석 평가 기준 중에서 상급에 해당되는 것들을 가지고 취약점 분석 시스템을 구현하였다.

또 서버관리자가 서버 계정을 관리하는데 있어 소홀해 질 수 있는 부분과, 기존의 아이디와 패스워드 유출 사례가 많은 것을 생각하여 보안의 취약점을 보완하기 위해 사내에서도 서버 관리자를 분리하여 그 관리자들만 관리 서버의 root 권한에 관해 인증서를 발급 받는다. 이 인증서 안에는 자신의 사원 정보가 들어있어, 그것을 바탕으로 한 개인키로 생성하며, 또한 인증서를 인증을 보강해줄 암호 키를 자신이 직접 입력한 패스워드로 걸어 인증에 대한 보안을 강화 하였다.

1.2 연구 목적 및 주제 선정

인증서란 인터넷 웹상에서 비즈니스 또는 거래를 할 때 거래 상대방을 믿고 신뢰할 수 있도록 하는 일종의 전자 보증서입니다. 현재 게임 상에도 아이디 패스워드와 더불어 2차 비밀번호나, otp 카드 발급을 하는 것처럼, 보통 아이디 패스워드만 사용하기엔 위험성이 너무 크고 누군가 아이디 패스워드를 알게 된다면 악용될 우려가 크기 때문에, 보안성을 더 높이기 위하여 인증서를 도입하게 되었습니다. 인증서 내용에는 이름, 메일, 부서 등의 내용이 첨가되기 때문에, 2차 피해에서도 피의자가 누구인지 쉽게 식별이 가능하게 해놓았고, 관리자가 주의 깊게 관리 한다면, 사용자에게 의해 설정이 변경된다거나 쉽게 취약점에 노출될 수 있기에 각각의 하위서버에 취약점 분석 툴을 설치하여 관리자가 쉽게 사용자의 컴퓨터에 취약점을 파악 할 수 있게 하였습니다. 보안의식이 높지 않아 많은 피해를 입었음에도 아직도 보안의식이 뚜렷하지 않아 사용자에게 비해 관리자가 터무니없이 부족한 현대 사회에 적은 관리자로 보안성을 높이기 위해 이 주제를 선정 하게 되었습니다.

2. 관련연구

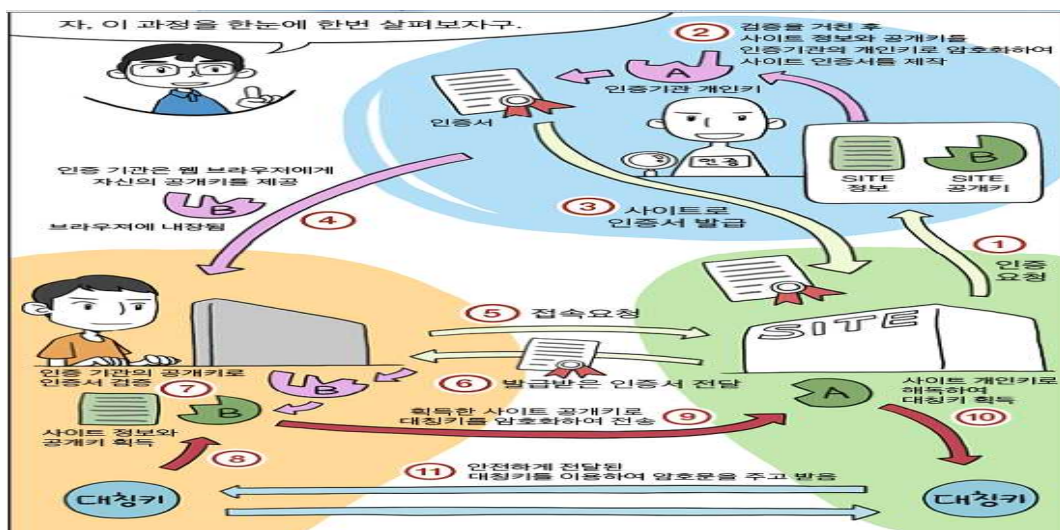
2.1 openssl

OpenSSL은 전송 계층 보안 (TLS) 및 SSL (Secure Sockets Layer) 프로토콜에 대한 강력한 상용 등급, 완전한 기능을 갖춘 툴 키트를 제공하는 오픈 소스 프로젝트입니다. 또한, 범용 암호화 라이브러리이다.

네트워크를 통한 데이터 통신에 쓰이는 프로토콜인 TLS와 SSL의 오픈 소스 구현판으로, C 언어로 작성되어 있는 중심 라이브러리 안에는, 기본적인 암호화 기능 및 여러 유틸리티 함수들이 구현되어 있다.

거의 모든 버전의 유닉스 계열 운영 체제(솔라리스, 맥 OS X, 리눅스, BSD 포함) 및 OpenVMS, 윈도우에서 OpenSSL을 이용할 수 있다. Secure Socket Layer(SSL) 프로토콜은 넷스케이프사에서 웹서버와 브라우저 간의 보안 통신을 위해 만들어졌다. SSL은 통신할 때 인증기관(Certificate Authority, CA)라는 것을 이용해서 서로 인식하게끔 되어 있다. 이 과정을 간단하게 설명하면 다음과 같다.

- 1.[웹브라우저] 보안 페이지를 요청한다. (일반적으로 주소에 https:// 라고 붙는다).
- 2.[웹서버] 자신의 공개키를 인증서와 함께 웹브라우저로 보낸다.
- 3.[웹브라우저] 웹서버의 인증서가 신뢰할 수 있는 제3자(신뢰할 수 있는 루트 인증기관, Trusted root CA)에게 서명되었는지 확인한다. 그리고 인증서가 아직 유효한지, 그리고 접속하려는 사이트와 연관되어 있는지 최종 확인한다.
- 4.[웹브라우저] 최종 확인이 되었으면 웹브라우저는 대칭 암호화키(대칭키)를 생성해서 웹서버의 공개키로 암호화한 후 송신한다. URL이나 기타 HTTP 데이터는 방금 생성한 대칭키를 이용해서 암호화한 후 웹서버로 전송한다.
- 5.[웹서버] 자신의 개인키를 이용해서 수신한 대칭키의 암호를 풀고, 이것을 이용해서 나머지 URL이나 기타 HTTP 데이터의 암호를 푼다.
- 6.[웹서버] 처리 결과(HTML문서+HTTP데이터)를 대칭키를 이용해서 암호화한 후 웹브라우저로 전송한다.
- 7.[웹브라우저] 대칭키를 이용해서 HTTP데이터와 HTML문서의 암호를 풀고 화면에 출력한다.



< 그림 2.1-2 Openssl을 이용한 인증서 >

2.2 취약점 분석 시스템

취약점이란 위협이 발생하는 전제 조건으로, 자산이 가지고 있는 보안상의 결점 또는 취약한 속성을 말합니다. 취약점은 자산이 부당하게 이용될 수 있는 안전장치의 부재 혹은 결점으로 간주되기도 합니다. 서버에 실행되는 오랜 버전의 서비스, 제한 없는 네트워크 접근, 침입 차단 시스템의 열린 포트, 느슨한 통제구역의 물적 보안, 서버와 pc에 취약한 패스워드 설정 등이 있습니다.

취약점을 발생 유형별로 보자면 소프트웨어의 전반적인 성능과 기능 요구 사항 등이 기록된 명세서가 존재하지 않아 원하는 성능과 기능을 충족하지 못하는 경우가 발생할 수 있습니다. 또한 부적절한 패스워드 설정 취약점으로 인해 부정접속이나 2차적으로 정보유출이라는 피해를 야기할 수도 있으며, 백업을 실시하지 않음으로 인해 장애 및 침해사고 발생 시 그에 대한 적절한 대처를 하기 어려운 취약점들도 있습니다.

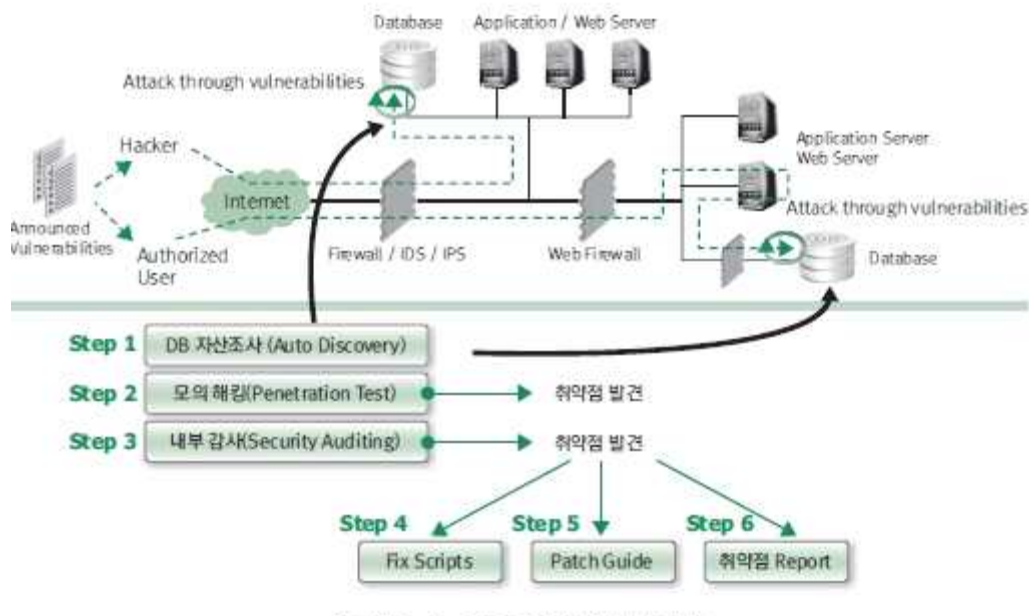
이러한 취약점의 발생 유형을 점검하기 위해 관리적 관점, 기술적 관점, 물리적 관점으로 나누어 볼 수 있습니다. 이 세 가지 분류에 구체적으로 알아보자면 우선 관리적 관점은 정보보호 관리체계 보안 통제에 근거하여 취약점을 점검하고, 조직과 조직에서의 정보자산 운영을 위해 어떠한 취약점이 존재하는지를 바라봐야합니다. 상세 분류에서는 업무 프로세스가 적합한지, 보안에 적절한 예산이 분배되고 있는지, 시설물에 대한 유지보수가 잘 이루어지고 있는지 등의 정보보호 운영에 대한 취약점을 점검합니다. 정보보호 관리의 취약성은 주로 조직 내 정책과 지침의 준수가 잘 이루어지는지, 조직 내 전담 인력이 존재하여 보안 활동을 잘 할 수 행하는지, 정보보호 교육을 통한 임직원들의 정보보호에 대한 의식일 제고하고 있는지 등을 점검합니다. 인적 관리에서는 임직원도 하나의 자원이라는 관점으로 바라보아 인적 자원으로 발생할 수 있는 보안 취약점을 점검합니다. 예를 들어 적합하지 않은 직무 부여, 외부인에 대한 감독 소홀, 정보보호 인식 부재 등이 있습니다.

기술적 관점은 주로 IT 인프라 시설에 대한 관리가 잘 이루어지고 있는지 점검하게 됩니다. 컴퓨터/통신관련 취약성은 컴퓨터와 통신이 안전하게 이루어지는지에 대한 내용을 점검하게 되며 인증 매커니즘, 접근통제, 감사 증적, 시스템 패치 등의 취약점을 점검합니다. 정보보호 시스템 관련 취약성은 침입 차단 시스템인 방화벽이나 침입 방지 시스템 IPS, 침입 탐지 시스템 IDS와 같은 보안 솔루션이 올바르게 운영되어 컴퓨터/통신에 대한 보안성 향상을 지원하고 있는지를 점검하게 됩니다. 시스템의 개발과 관련된 취약성이 있으며, 이것은 조직이 사용하는 애플리케이션 개발이 안전하게 이루어지는지를 점검하게 됩니다. 개발 단계에서 보안 측면이 전혀 고려되지 않은 채 최종 애플리케이션이 나오게 된다면 무수히 많은 잠재적인 보안 취약점을 가지게 됩니다. 더구나 개발 완료 후 기능을 수정하는 것에는 막대한 예산이 소모됩니다. 따라서 개발 단계부터 안전하게 개발이 이루어지는지 그 취약점을 점검할 필요가 있습니다.

물리적 관점은 시설물들에 대해 관리 및 설치가 잘 이루어지고 있는지 점검하게 됩니다. 물리적 취약성은 물리적 보안을 위해 출입 통제 시스템을 잘 활용하고 있는지, 올바른 위치에 설치되어 있는지 등 물리 보안에 대한 취약점을 점검하게 됩니다. 환경적 취약점은 환경요인 변화에 따른 보안 취약점을 점검하기 위해 화재 및 수재 장치 등이 올바른 위치에 설치되어

있는지, 정상적인 동작을 하는지에 대해 주기적인 점검을 실시하게 됩니다.

이러한 취약점을 진단 절차는 침입자가 관리자가 운영하는 네트워크와 서버 등에 침입할 수 있는 보안상의 허점이 있는지 기술적으로 분석하는 과정이며, 절차는 자산 조사 및 분석을 하고 진단 대상을 선정하는 등의 절차를 통해 진단을 완료합니다.



< 그림 2.2-1 취약점 분석 절차 >

분류	항목	등급
계정관리	Root계정 원격접속 외 4건	상
파일 및 디렉토리 관리	root 홈 패스 디렉터리 권한 및 패스 설정 외 14건	상
서비스 관리	finger 서비스 비활성화 외23	상
패치 관리	최신보안 패치 및 벤더 권고 사항 적용 1건	상
로그 관리	로그 정기적 검토 및 보고 1건	상

< 그림 2.2-2 취약점 분석 항목>

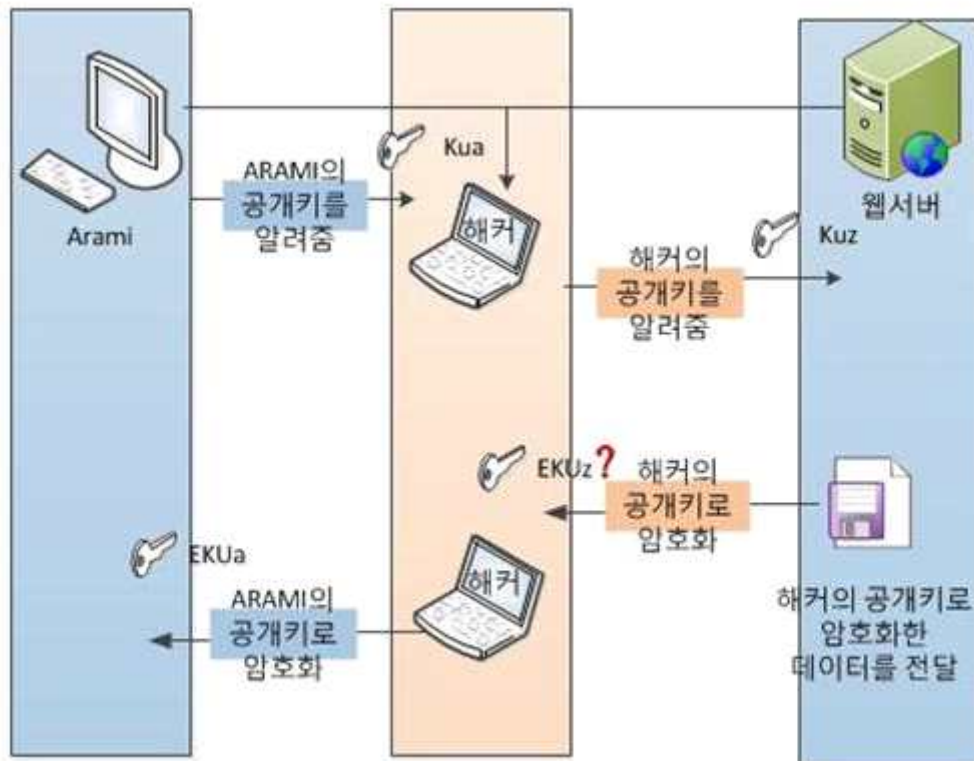
2.3 인증서 인증

*인증서의 정의

인증서란, 서명이나 인감도장과 같은 역할을 하는 전자서명이 특정인에게 유일하게 속한다는 사실을 확인하고 이를 증명하는 전자적 정보를 말하며, 공인인증기관이 발급하는 인증서를 공인인증서라 한다.

공인인증서 내에는 가입자의 전자서명 검증키, 일련번호, 소유자이름, 유효기간 등의 정보를 포함하고 있다. 따라서, 공인인증서는 거래 당사자의 신원 확인은 물론 문서의 위·변조 방지, 거래사실의 부인 방지 등의 기능을 가지며, 안전한 거래를 보장한다.

*인증서의 필요성



<그림 2.3-1 인증서 필요성 >

공개키 기반의 보안 통신은 상대방의 공개키로 데이터를 암호화해서 보내고 이를 받은 유저는 자신의 개인키로 복호화 해야 한다. 하지만 그림 2.3-1에서 "Arami"라는 유저는 웹 서버와 암호화 통신을 하고 싶어 자신의 공개키를 웹서버에게 넘겨 주고자 하지만 중간에 해커가 끼어들어 "Arami"의 공개키를 가로채고 자신의 공개키를 A의 공개키인 것처럼 웹서버에게 보내고 있다. 이를 알 리가 없는 웹 서버는 해커의 공개키가 Arami의 공개키라고 믿고 추후 Arami와 보안 통신을 할 때 해커의 공개키로 암호화해 전달한다. 이를 가로챈 해커는

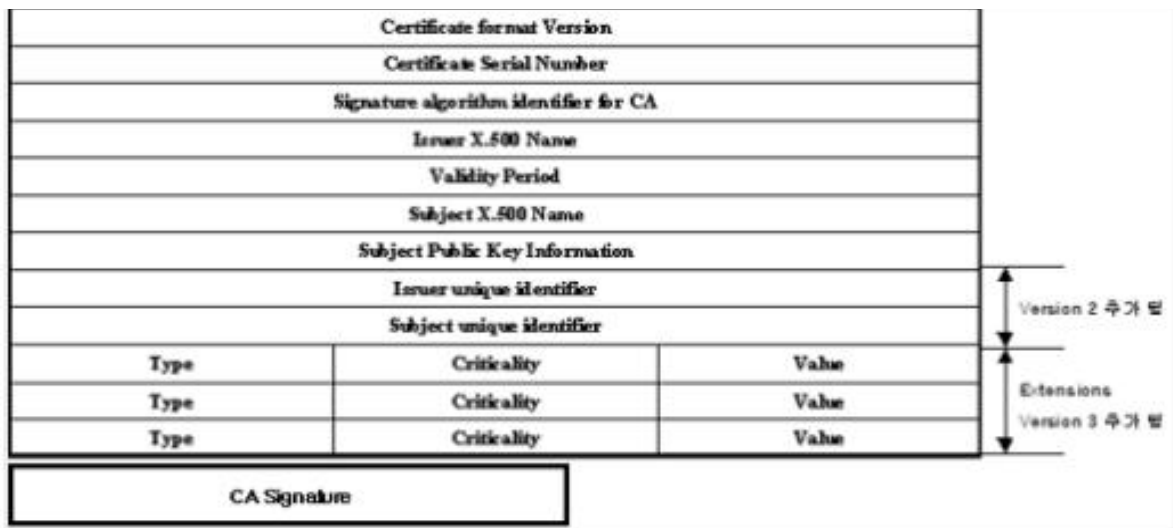
자신의 공개키로 암호화 된 데이터이니 아무런 문제없이 자신의 개인키로 복호화해 데이터를 확인하고 Arami의 공개키를 가지고 있으니 Arami의 공개키로 데이터를 암호화 해 Arami에게 보내 준다.

여기서의 가장 큰 문제점은 웹 서버가 데이터를 확인하고 Arami의 공개키를 믿을 수 없다는 데 있다. 만약 Arami의 공개키를 믿을 수 있고 이 키 값을 가지고 있다면 해커가 중간에 끼어들어도 Arami의 개인키 값을 모르고 때문에 데이터의 복호화 및 릴리즈는 힘들 것 이다. 이러한 문제점을 해결하기 위해 제 3자의 공인된 인증기관에 자신의 공개키를 등록하고 이 공개키가 수납된 공인인증서를 발급 받아 이 키를 필요로 하는 사람에게 배포하는 공인 인증서에 사용자의 공개키, 유효기간, 사용자 ID, CA등의 정보와 인증서의 변조를 방지하기 위한 CA의 개인키에 대한 디지털 서명이 추가되었다.

*X509 인증서

X509는 PKI에서 사용하는 표준 인증서 형식이다. PKI에서 사용하는 공개키, 개인키 같은 비대칭 키를 X509인증서로 관리한다. 그리고 X509인증서는 Thawte, Verisign, RSA와 같은 외부 CA기관에서 승인 과정을 통해 발행된다.

X509인증서는 X500 디렉토리 추천 목록의 한 부분으로 1988년에 처음으로 ITU-T와 ISO/IEC에 의해 발표된 인증서 형식이다. X509(V1)은 1993년 디렉토리 접근 제어를 위한 두 가지 내용을 추가하기 위해 X509(V2)형식으로 개정 되었다. 그리고 이메일의 보안요소등 새로운 개념이 포함된 X509(V3)가 1996년에 발표 되었다.



<그림 2.3-2 인증서 형식>

[

- 1.1. Certificate format Version : 인증서의 버전을 나타냄
- 1.2. Certificate Serial Number : CA에 의해 할당된 정수의 고유 번호로 각각의 인증서의 일련번호는 고유한 숫자이어야 함
- 1.3. Signature algorithm identifier for CA : CA가 인증서 서명하기 위해 사용한 RSA나 DSA 등과 같은 알고리즘을 식별하기 위한 항목
- 1.4. Issuer X.509 Name : 인증서 발급하고 서명한 CA 이름
- 1.5. Validity Period : 인증서가 유효한 기간을 표시한다.
- 1.6. Subject X.509 Name : 인증서의 소유자 즉 인증서의 공개키 항목에 나타난 공개 키

를 소유한 개체이다. 이때 CA에 의해 확인된 각각의 주체명은 고유한 이름이어야 한다.

1.7. Subject Public Key Information : 키가 사용하는 알고리즘의 식별자 및 키 값이 저장됨

1.8. Issuer unique identifier : 발급자명의 재사용을 위한 선택 항목

1.9. Subject unique identifier : 주체명의 재사용을 위한 선택 항목

1.10. Extension : 확장 필드는 확장 형태, criticality indicator 그리고 확장 값 세가지로 구성된다. 확장 형태는 의미와 형태 정보(문자열, 날짜 혹은 복잡한 데이터 구조 등)에 대한 객체 식별자이다. Criticality Indicator는 확장을 인식하지 못한 경우 이를 무시해도 무방한가를 결정해 주는 식별자이다. 그리고 확장 값은 확장 형태에 기술된 자료의 실제적인 자료이다.

1.11. CA Signature : 전자 서명으로 메시지를 해쉬 알고리즘을 사용하여 특정 길이의 값을 생성한다. 그리고 이 값을 발행자의 개인키로 암호화한다.]

*PKI 공개키 기반 구조

공개키 기반(PKI)은 private key(개인키)와 public key(공개키)로 이루어져 있다. 인증서라고 하는 것은 내 공개키가 맞다고 인증기관(CA)이 전자 서명하여 주는 것이며 나와 보안 통신을 하려는 당사자는 내 인증서를 구해서 그 안에 있는 공개키를 이용하여 보안 통신을 할 수 있다.

PKI는 기본적으로 인터넷과 같이 안전이 보장되지 않은 공중망 사용자들이, 신뢰할 수 있는 기관에서 부여된 한 쌍의 공개키와 개인키를 사용함으로써, 안전하고 은밀하게 데이터나 자금을 교환할 수 있게 해준다. PKI는 한 개인이나 기관을 식별할 수 있는 디지털 인증서와, 인증서를 저장했다가 필요할 때 불러다 쓸 수 있는 디렉토리 서비스를 제공한다. 비록 PKI의 구성 요소들이 일반적으로 알려져 있지만, 공급자 별로 많은 수의 서로 다른 접근방식이나 서비스들이 생겨나고 있으며, 그동안에도 PKI를 위한 인터넷 표준은 계속하여 작업이 진행되었다.

PKI는 인터넷 상에서 메시지 송신자를 인증하거나 메시지를 암호화하는데 있어 가장 보편적인 방법인 공개키 암호문을 사용한다. 전통적인 암호문은 대개 메시지의 암호화하고 해독하는데 사용되는 비밀키를 만들고, 또 공유하는 일들이 관여된다. 이러한 비밀키나 개인키 시스템은, 만약 그 키를 다른 사람들이 알게 되거나 도중에 가로채어질 경우, 메시지가 쉽게 해독될 수 있다는 치명적인 약점을 가지고 있다. 이러한 이유 때문에, 인터넷 상에서는 공개키 암호화와 PKI 방식이 선호되고 있는 것이다 (개인키 시스템은 때로 대칭 암호작성법, 그리고 공개키 시스템은 비대칭 암호작성법이라고도 불린다).

PKI는 다음과 같은 것들로 구성된다.

- *디지털 인증서를 발급하고 검증하는 인증기관

- *공개키 또는 공개키에 관한 정보를 포함하고 있는 인증서

- *디지털 인증서가 신청자에게 발급되기 전에 인증기관의 입증을 대행하는 등록기관

- *공개키를 가진 인증서들이 보관되고 있는 하나 이상의 디렉토리

- *인증서 관리 시스템

- 공개키와 개인키 암호화의 동작원리

공개키 암호화에서, 공개키와 개인키는 인증기관에 의해 같은 알고리즘(흔히 RSA라고 알려져 있다)을 사용하여 동시에 만들어진다. 개인키는 요청자에게만 주어져, 공개키는 모든

사람이 접근할 수 있는 디렉토리에 디지털 인증서의 일부로서 공개된다. 개인키는 절대로 다른 사람과 공유되거나 인터넷을 통해 전송되지 않는다. 사용자는 누군가가 공개 디렉토리에서 찾은 자신의 공개키를 이용해 암호화한 텍스트를 해독하기 위해 개인키를 사용한다. 그러므로, 만약 자신이 누구에게나 어떤 메시지를 보낸다면, 우선 수신자의 공개키를 중앙관리자를 통해 찾은 다음, 그 공개키를 사용하여 메시지를 암호화하여 보낸다. 그 메시지를 수신한 사람은, 그것을 자신의 개인키를 이용하여 해독한다. 메시지를 암호화하는 것 외에도, 송신자는 자신의 개인키를 사용하여 디지털 인증서를 암호화하여 함께 보냄으로써, 메시지를 보낸 사람이 틀림없이 송신자 본인이라는 것을 알 수 있게 한다.

* PKI(Public Key Infrastructure)에서 X.509

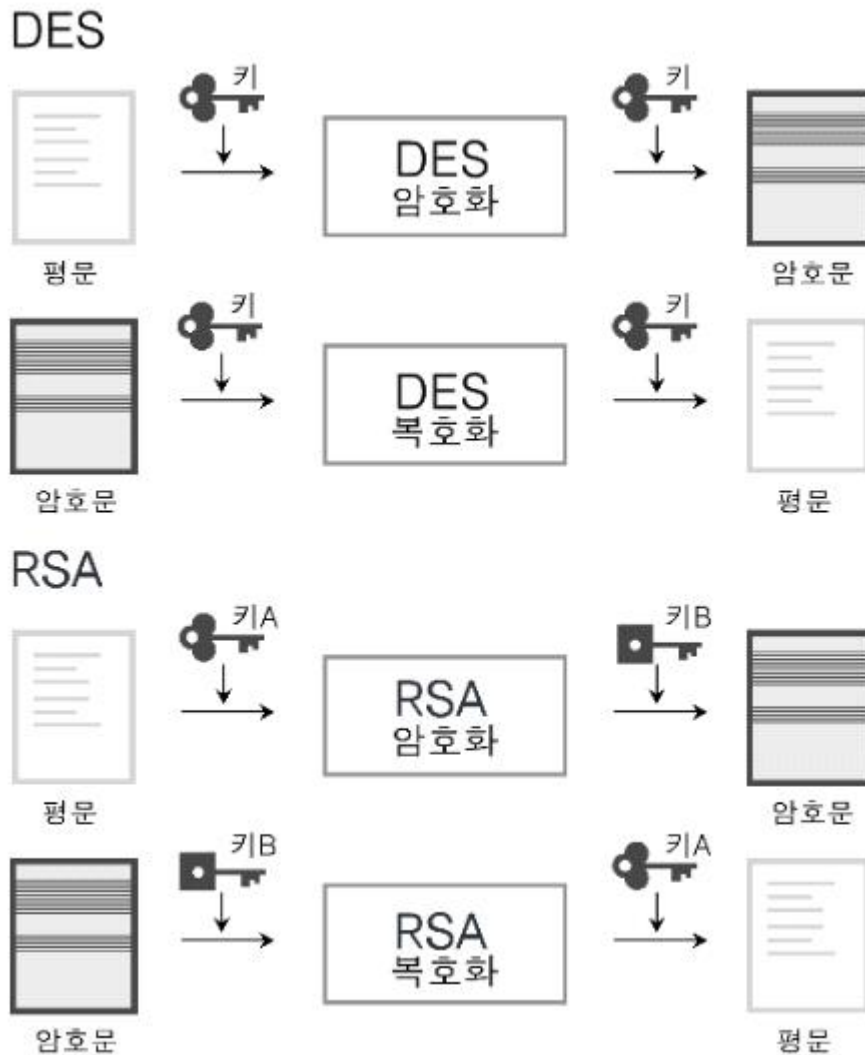
PKI는 공개키와 개인키를 사용하여 메시지를 암호화하고 이 암호화된 메시지 값의 무결성 보장과 메시지 보낸 이에 대한 증명등과 같이 통신상에서 주고받는 메시지를 받는 이에게 메시지의 안정성을 보장해 주는 일련의 과정을 수행한다.

*RSA 암호화 알고리즘

RSA 알고리즘은 미국 MIT의 Rivest, Shamir, Adleman이 발표한 공개키 암호화 방식으로 공개키 암호화 의 개념을 수학적으로 구체화 시킨 알고리즘이다.

이 RSA, 공개키 암호화 알고리즘은 현재 공개키 암호화에서 가장 널리 쓰이고 있는 공개키 알고리즘이다. 그 이유는 RSA 공개키 암호화 알고리즘이 최초로 공개키 암호화 의 개념을 구현한 이유도 있지만, 그 안정성이 십 여년이상을 통해 검증이 되었고, 그동안 발표 되어온 공개키 암호화 알고리즘 중에서 이해와 구현이 쉽게 때문이다.

RSA 공개키 암호화 알고리즘의 기본 형태는 DES 같은 대칭키 암호화와 같다. 키를 사용하여 평문을 암호화 시켜 암호문을 출력 하는 것이다. 대칭키 암호화와 다른 것은 암호화 하는 키와 복호화 하는 키가 다르다는 것 밖에 없다. 이런 기본 암호화 골격은 같지만 암호화 키와 복호화 키를 다르게 하기 위해 RSA 암호화 알고리즘은 DES와 같은 대칭키 암호화 알고리즘과 확연히 다른 내부 알고리즘을 사용하고 있다.



<그림 2.3-3 DES와 RSA >

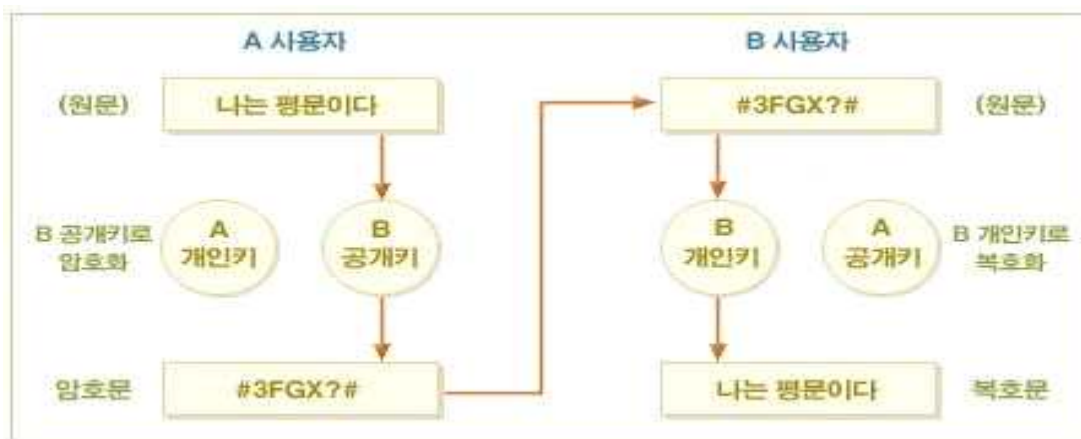
RSA 암호화 기법은 구현이 간단하고, 안전성이 높기 때문에 많이 사용되고 있다. 블록단위로 암호화를 하며, 각 블록은 n (키 값의 곱)보다 작은 바이너리 값으로 이루어져 있다. 블록 사이즈는 $\log_2(n)$ 보다 작거나 같다. 암호화 방법은 $C = Me \bmod n$ 방법으로 되며, 복호화는 $M = Cd \bmod n = (Me)d \bmod n = Med \bmod n$ 의 형태로 된다. 송신자와 수신자는 서로 n 값을 알고 있으며, 공개키는 $\{e, n\}$ 이며, 비밀키는 $\{d, n\}$ 이다. 암호화 방법에 있어서 $Me \bmod n$ 의 연산은 매우 쉽지만, 반대로 그 역 연산은 매우 어렵다. Mod 연산을 통해서 수 없이 많은 값들이 M 값이 될 수 있기 때문이다. RSA에서의 오일러 토션 함수와 오일러 함수의 특성이 들어가 있다.

오일러 함수로부터, $a^{\phi(n)+1} = a \pmod n$, where $\gcd(a, n)=1$, $ed=\phi(n)+1 \Rightarrow ed \bmod \phi(n) = 1$ 가 정의되고, 이는 $ed = 1 \bmod \phi(n)$. $d=e^{-1} \bmod \phi(n)$ 형태로도 표현이 가능하다. 역 역산이 어려운 이유도, e 와 d 가 $\bmod \phi(n)$ 에서 많은 inverse가 존재하기 때문이다. 이러한 수학적인 연산의 어려움 때문에(Discrete Logarithm Problem) 키 없이 암호문을 해독하기가 매우 어렵다. 전수 조사를 통한 해결 밖에 방법이 없는데, 이 연산의 지수적으로 표현

되기 때문에 시간이 매우 오래 걸린다. 컴퓨터의 성능이 발달하게 되면서, 연산의 속도가 빨라졌기 때문에, 그 만큼 키 값의 크기도 증가시켜줘야 안전도에 문제가 없다. 그래서 현재는 1024 비트 이상의 키를 사용하도록 권장하고 있다.

RSA 암호화 키 셋업

각 유저는 공개키와 비밀키를 가지고 있다. 우선 매우 큰 임의의 소수 p , q 를 선택한다. 그리고 이 둘의 곱($n = pq$)을 구한다. p 와 q 소수이기 때문에 오일러 토션 함수에 의해서 $\phi(n)=(p-1)(q-1)$ 가 된다. 암호화를 위한 키 e 를 선택한다. 이 때, $1 < e < \phi(n)$, $\gcd(e, \phi(n))=1$ 를 만족해야 한다. 즉, $\phi(n)$ 보다 작은 양수이며, $\phi(n)$ 과도 서로소가 되어야 한다. 이제 복호화 키 d 를 구해야 한다. 이 때, $ed = 1 \pmod{\phi(n)}$ and $0 \leq d \leq n$ 식을 만족하는 d 를 구해야 한다. 그러면 키 셋업은 끝이 난다. 공개키는 $\{e, n\}$ 이 되며, 비밀키는 $\{d, p, q\}$ 가 된다.

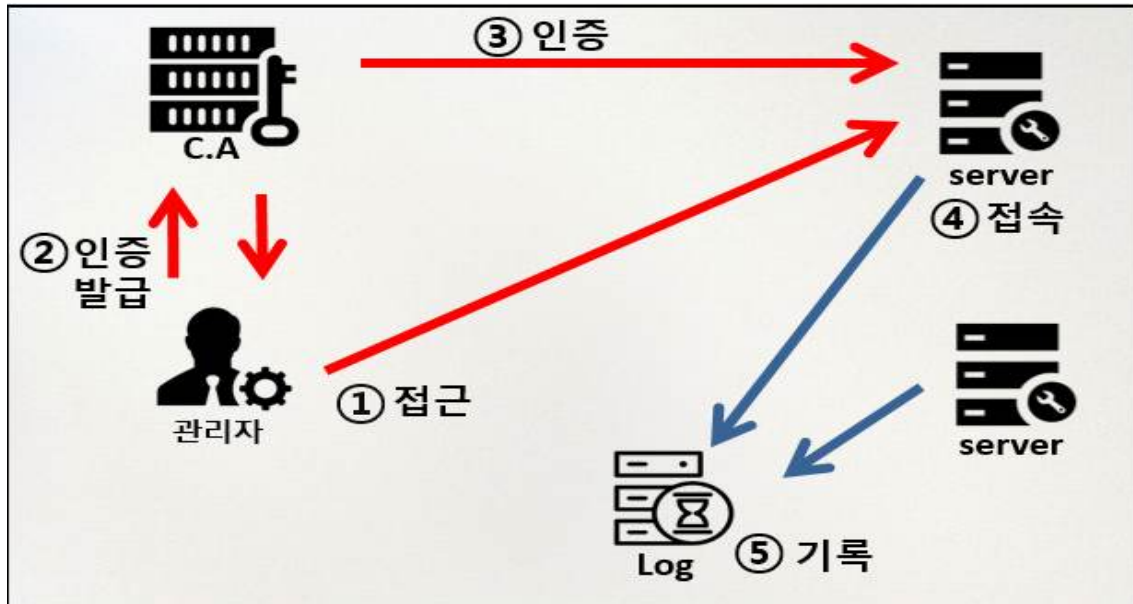


<그림 2.3-4 RSA 알고리즘>

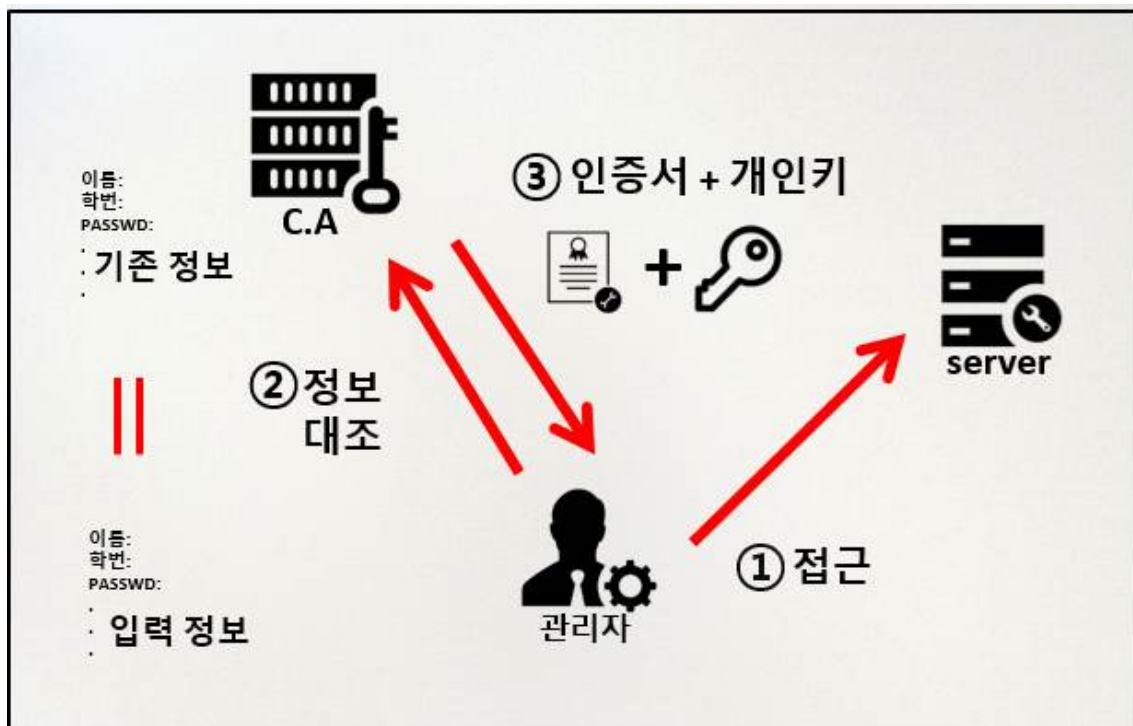
3. 본론

3.1 시스템 프로세스

< 전체 구성도 >



3.1 인증서 생성 방법



ROOT 권한을 얻기 위해 ROOT계정의 ID/Password로 접속을 시도한다.

사용자 이름:

취소 다음

암호:

취소 로그인

ID/Password 입력 후 인증서 발급, 인증서 로그인을 위한 GUI프로그램이 실행된다.

아래 목록에서 항목을 선택하십시오.

check	menu
<input type="radio"/>	인증서 발급
<input type="radio"/>	인증서 로그인

취소(C) 확인(O)

인증서 발급 메뉴선택 시 입력창을 통해 정보를 입력하게 된다.

정보입력

이름	ahn
ID	mama2233
Password (8글자 이상)	●●●●●●●●
Country (korea)	KR
Student Number (9xxxxxxx)	91317000
Name (honggildong)	min
Mail Address (xxx@xxx.xxx)	parkih@naver.com


취소(C) 확인(O)

입력한 정보 값이 인증서 서버를 통해 대조가 이루어진다. 입력 값이 유효할 경우 관리자가 USB를 통해 인증서를 저장가능, 유효하지 않을 경우 오류 메시지와 함께 초기 화면으로 돌아간다.


USB선택

SanDisk	15G
---------	-----

취소(C) 확인(O)

 오류가 발생했습니다.

확인(O)

 인증서 발급 완료

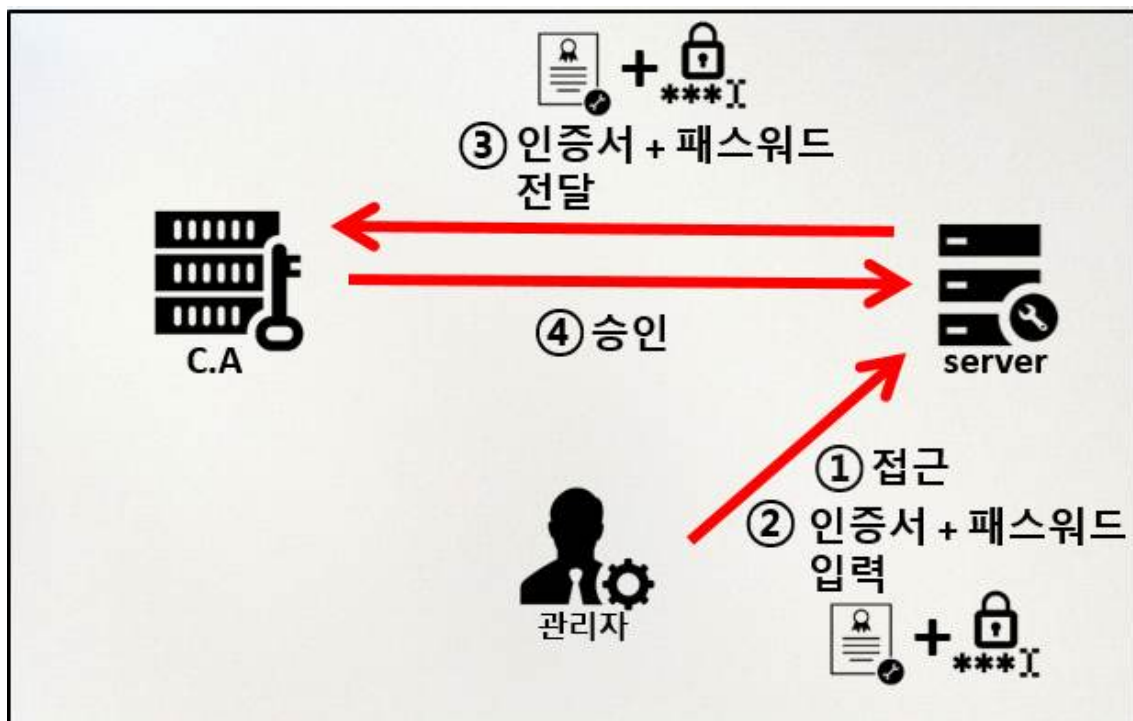
확인(O)

```
root@localhost:/home/centos1/openssl
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
a
a
a
a
a
a
a
a
a
a
^CKilled by signal 1.

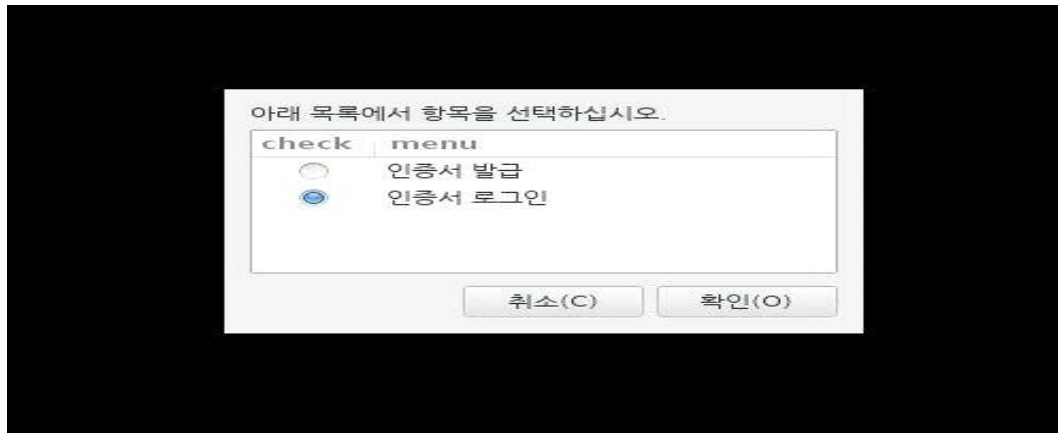
[root@localhost openssl]# sh cat.sh
a
Generating RSA private key, 2048 bit long modulus
.....+++
.....+++
e is 65537 (0x10001)
writing RSA key
Signature ok
subject=/C=KR/ST=91317000/L=ahn/O=woanhon/OU==woanhon/emailAddress=min/CN=minh1p
ark@naver.com
Getting CA Private Key
```

인증서버에서 인증서 생성이 정상적으로 이루어지는 것을 알려주는 셸이다.

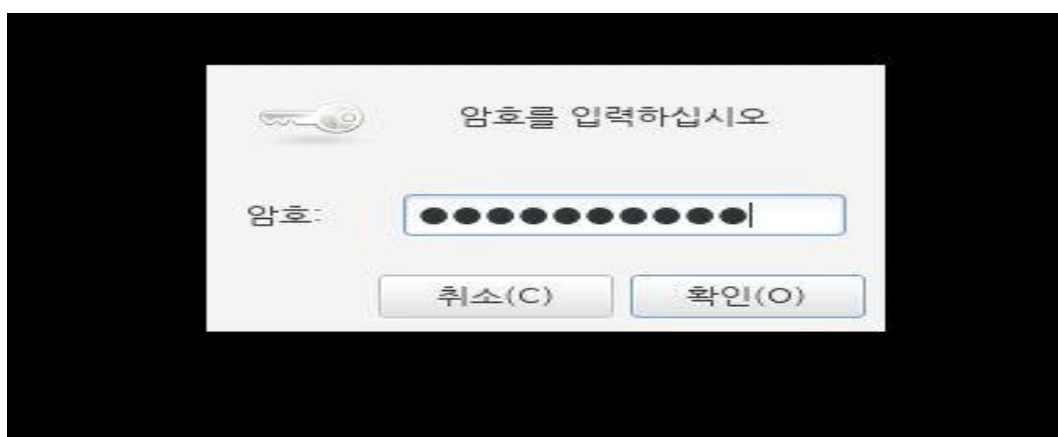
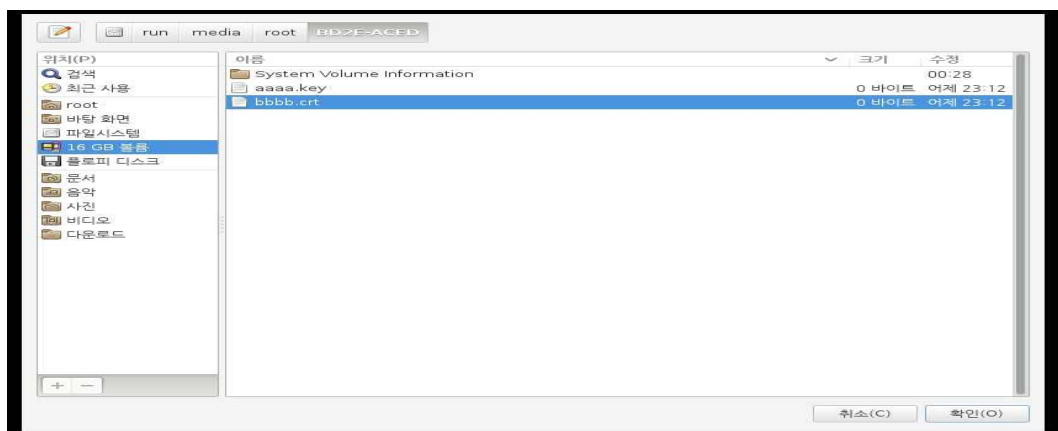
3.2 인증서 로그인 방법



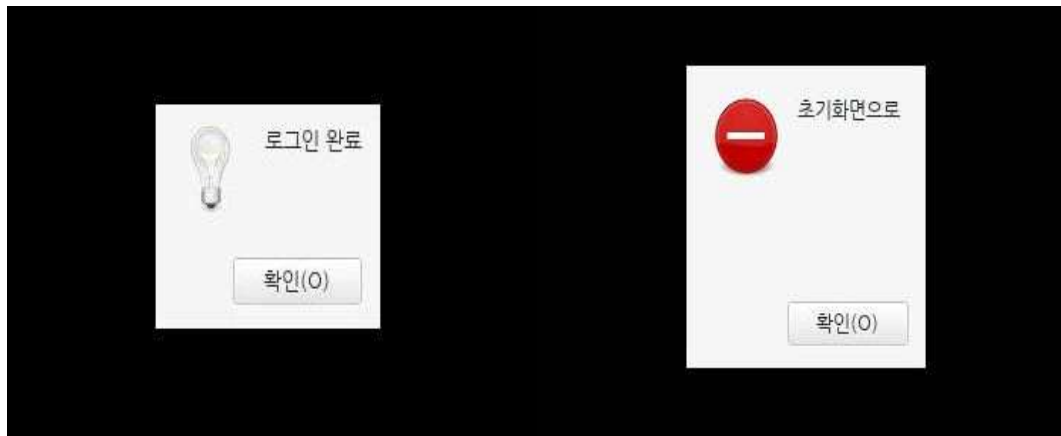
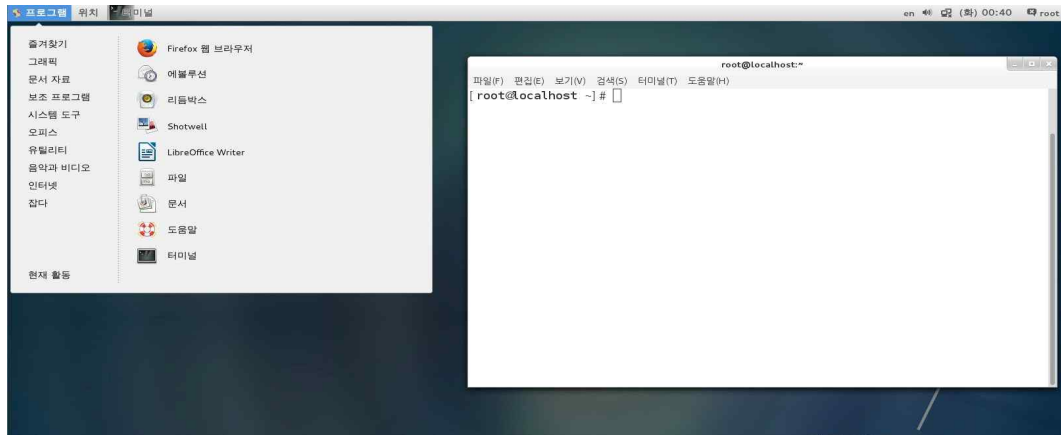
인증서를 정상적으로 발급 받은 후 인증서 로그인 메뉴를 통해 인증서를 통한 로그인을 시도한다.



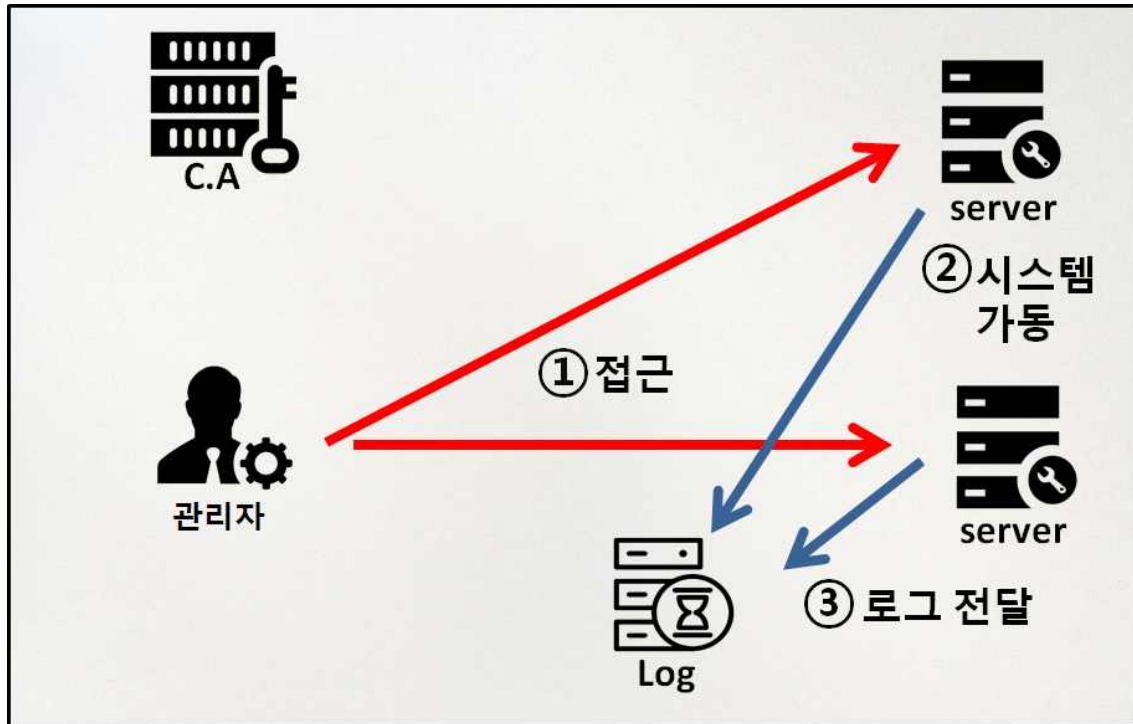
인증서 로그인을 위해 USB에 저장된 인증서를 선택하고 설정한 비밀번호를 입력한다.



인증서와 비밀번호가 유효할 경우 정상적인 ROOT권한으로 로그인이 완료 되고, 유효하지 않을 경우 오류 메시지와 함께 로그인에 실패하고 초기화면으로 돌아간다.



3.3 취약점 시스템



각 로그들을 확인 한다.

```
root@localhost:/sh

파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
20160516- 19: 46-----
-----
현재 접속 중인 사용자-----
-----
19: 47: 08 up min, average:
USER TTY IDLE
centos : 0 ?xdm? [pam/gdm-autologin]
centos pts/0 5: 24

secure log-----
-----
-----

mmessages log-----
-----
-----
May 16 19: 46: 47 localhost NetworkManager: DHCPREQUEST on ens33 to 255.255.255.25
5 port 67 (xid=0x39fdb287)
May 16 19: 46: 54 localhost dhclient[1469]: DHCPREQUEST on ens33 to 255.255.255.25
5 port 67 (xid=0x39fdb287)
May 16 19: 46: 54 localhost NetworkManager: DHCPREQUEST on ens33 to 255.255.255.25
5 port 67 (xid=0x39fdb287)

lastlog-----
-----
-----
사용자 이름      포트      어디서      최근 정보
root            pts/0
gdm             : 0
centos          : 0
월 5월 16 19: 31: 44 +0900 2016
금 11월 28 21: 09: 31 +0900 2014
월 5월 16 18: 53: 30 +0900 2016

last-----
-----
-----
reboot    system boot  3. 10. 0- 123. el7. x  Fri Nov 28 14: 37 - 14: 38  (00: 01)
centos    : 0      : 0      Fri Nov 28 11: 48 - 13: 17  (01: 28)
(unknown : 0      : 0      Fri Nov 28 11: 46 - 11: 48  (00: 02)
reboot    system boot  3. 10. 0- 123. el7. x  Fri Nov 28 20: 45 - 13: 17  (- 7: - 28)
reboot    system boot  3. 10. 0- 123. el7. x  Fri Nov 28 20: 44 - 11: 45  (- 8: - 58)

wtmp begins Fri Nov 28 20: 44: 12 2014

cron log -----
-----
-----
May 16 19: 06: 03 localhost anacron[2844]: Job `cron.daily' terminated
May 16 19: 06: 03 localhost anacron[2844]: Normal exit (1 job run)
May 16 19: 10: 01 localhost CROND[3194]: (root) CMD (/usr/lib64/sa/sa1 1 1)
May 16 19: 20: 01 localhost CROND[3316]: (root) CMD (/usr/lib64/sa/sa1 1 1)
May 16 19: 30: 02 localhost CROND[4027]: (root) CMD (/usr/lib64/sa/sa1 1 1)
```

취약점시스템 실행

=====1. 계정 관리=====

1. root계정 원격 접속 제한
2. 패스워드 복잡성 설정
3. 계정 잠금 임계값 설정 점검
4. 패스워드 파일 점검
5. 패스워드 최소 길이 설정
6. 패스워드 최대 사용기간 설정
7. 패스워드 최소 사용기간 설정

=====2. 파일 및 디렉터리 관리=====

1. 환경 변수 파일 점검
 2. 파일 및 디렉터리 소유자 점검.
 3. /etc/passwd 파일 소유자 및 권한 설정
 4. /etc/shadow 파일 소유자 및 권한 설정
 5. /etc/hosts 파일 소유자 및 권한 설정
 6. /etc/xinetd.conf 파일 소유자 및 권한 설정
 7. /etc/services 파일 소유자 및 권한 설정
 8. SetUID, SetGID, Sticky bit설정 파일 점검
-
9. 사용자 시스템 시작파일 및 환경파일 소유자 및 권한 설정
 10. world writable 파일 점검
 11. /dev에 존재하지 않은 device 파일 점검
 12. /root/.rhosts, hosts.equiv 사용금지
 13. 접속 IP 및 포트 제한

=====3. 서비스 관리=====

1. Finger 서비스 점검
 2. Anonymous FTP 점검
 3. r계열 서비스 점검.
 4. cron파일 소유자 및 권한 설정값
 5. DOS공격에 취약한 서비스 점검
 6. NFS서비스 실행상태 점검
 7. NFS 접근 통제
 8. automountd 점검
 9. RPC 서비스 확인
 10. DNS 보안 버전 패치
-

실행 결과 취약 항목 검출

15. Apache 파일 업로드 및 다운로드 제한

16. Apache 웹 서비스 영역의 분리

=====4. 로그 관리=====

로그 파일 확인

취약점의 총 개수는 18개 입니다.

취약점 부분만 지금 바로 보시겠습니까?

y

```
[취약] => 원격 접속 포트가 열려 있습니다.
[취약] => 패스워드가 너무 짧습니다.
[취약] => 계정 잠금 임계값이 설정되어 있지 않습니다.
[취약] => 패스워드 파일이 보안에 취약합니다.
[취약] => 현재 패스워드 최소길이(5글자)로 너무 짧습니다.
[취약] => /etc/hosts 파일 소유자 설정이 취약합니다.
[취약] => /etc/xinetd.conf 파일 소유자 설정이 취약합니다.
[취약] => /etc/xinetd.conf 파일 권한 설정이 취약합니다.
[취약] => world writable 파일이 5개 이상 존재합니다..
[취약] => finger 서비스가 작동 중 입니다.
[취약] => Anonymus Ftp 계정이 존재합니다.
[취약] => cron 파일의 소유자 및 권한 설정이 취약합니다.
[취약] => NFS 서비스가 작동하지 않습니다.
[취약] => automount가 작동 중입니다.
[취약] => RPC 서비스가 작동 중 입니다.
[취약] => Apache 웹 프로세스 권한이 취약합니다.
[취약] => Apache 상위 디렉토리 접근이 취약합니다.
[취약] => Apache 파일 업로드 및 다운로드가 취약합니다.
```

로그 서버에 남는 기록

2. 파일 및 디렉터리 소유자 점검.
[양호]

3. /etc/passwd 파일 소유자 및 권한 설정
[양호]

4. /etc/shadow 파일 소유자 및 권한 설정
[양호]

5. /etc/hosts 파일 소유자 및 권한 설정
[취약] => /etc/hosts 파일 소유자 설정이 취약합니다.
[양호]

6. /etc/xinetd.conf 파일 소유자 및 권한 설정
[취약] => /etc/xinetd.conf 파일 소유자 설정이 취약합니다.
[취약] => /etc/xinetd.conf 파일 권한 설정이 취약합니다.

7. /etc/services 파일 소유자 및 권한 설정
[양호]

8. SetUID, SetGID, Sticky bit 설정 파일 점검
SetUID 설정 파일 점검

합계 28

홀 디렉토리 /perm/suid 저장

SetGID 설정 파일 점검

합계 14

홀 디렉토리 /perm/sgid 저장

Sticky bit 설정 디렉토리 점검.

9. 사용자 시스템 시작파일 및 환경파일 소유자 및 권한 설정
[양호]

10. world writable 파일 점검.
[취약] => world writable 파일이 5개 이상 존재합니다..

11. /dev에 존재하지 않은 device 파일 점검
[양호]

12. /root/, rhosts, hosts.equiv 사용 금지
[양호]

13. 접속 IP 및 포트 제한
양호

=====3. 서비스 관리=====

1. Finger 서비스 점검 (3항목 체크)

[양호]

[취약] => finger 서비스가 작동 중입니다.

[양호]

2. Anonymous FTP 점검

[취약] => Anonymus Ftp 계정이 존재합니다.

3. r계열 서비스 점검 (3항목 체크)

[양호]

[양호]

[양호]

4. cron파일 소유자 및 권한 설정값

[취약] => cron 파일의 소유자 및 권한 설정이 취약합니다.

5. DOS공격에 취약한 서비스 점검

[양호]

6. NFS서비스 실행상태 점검

[취약] => NFS 서비스가 작동하지 않습니다.

7. NFS 접근 통제

[양호]

8. automountd 점검

[취약] => automount가 작동 중입니다.

9. RPC 서비스 확인

[취약] => RPC 서비스가 작동 중입니다.

10. DNS 보안 버전 패치

[안전]

11. Apache 디렉토리 리스팅 제거

[양호]

12. Apache 웹 프로세스 권한 제한

root 7595 7206 0 23:04 ? 00:00:00 grep httpd

[취약] => Apache 웹 프로세스 권한이 취약합니다.

4. 결론

취약점 분석 시스템을 연구하면서 리눅스에는 CUI 터미널 창 과 같은 곳이 아닌 GUI 즉 그래픽을 기반으로 실행이 되는 프로그램을 알게 되었고, 또 그것을 로그인하고 바로 실행되게 하기 위해 리눅스의 부팅 순서나 리눅스에 프로세스나 설정 파일을 공부하였다.

관리자가 로그아웃을 하거나 사용을 중지한 상태가 길어져 화면보호기가 켜져 있을 때 생길 수 있는 취약점을 주요 정보 통신기반 시설 취약점 분석 평가 보고, 취약점 분석 항목이 왜 생길 수 있는지 연구하고, 각 항목에 대해 쉘 프로그램으로 작동 할 수 있도록 알맞은 명령어를 공부하였다. 이때 이 항목들 중 상 중 하로 등급을 나누어 취약점 분석 툴에 빠지지 말아야 할 상급의 항목들만 작업하여 로그보고서를 받는 작업을 이뤘다. 작성한 취약점 시스템들 중 root 권한이 필요한 항목들이 포함되어 있는 것을 포함한 다는 것을 전제하여 진행하고, 진행이 끝나면 로그인이 된 사용자 배경화면에 보고서를 띄우고, 취약 항목과 개수를 바로 알려줌으로써 사용자는 자신의 서버에 문제점을 보다 정확하고, 신속하게 알 수 있을 것이다.

또한 안전과 취약의 항목을 모아 로그 서버에 보내주면서 지난 기록도 다시한번 관리자가 찾아 볼 수 있다.

사용자 로그인 문제에서 일어날 수 있는 password 유출과 계정 아이디 유출 문제를 보안하고자. 인증서 인증을 생각하였다. 공인인증서와 같은 원리의 개인 인증서를 작성하고, 개인 CA를 생성 구축하여, 인증서버의 root 개인키를 이용해 발급해주는 방법으로 접근 하였다.

인증서 알고리즘은 openssl을 전격 활용하여, PKI공개키 기반의 RSA암호화 인증서를 제작하여 해당 관리자에게 전달하였고, 이 과정에서 관리자 정보를 인증 서버가 가지고 있어, (원래인사 DB가 따로 있어 인사 정보를 가져와 대조하는 방법) 들어오는 정보와 대조, 결과 값이 참일 경우 입력받은 내용을 기반으로 개인키를 생성하고, 인증서를 발급 해주는 방식으로 진행 하였다. 이 과정이 모두 끝나면 관리자는 자신이 사용하는 usb를 불러와, usb안에 발급 받은 개인키와 인증서를 저장하는 방식으로 이루어진다.

이 발급받은 인증서는 인증서 로그인 할 시에 인증서와 같이 정보입력란에 입력되어 있는 password를 같이 입력하여, 인증서버에 password로 묶여진 자신의 인증서의 해시 값 파일을 풀고 그 안의 내용과 지금 들어오는 인증서의 해시값을 도출해 그 두 개를 비교하는 방식으로 인증서 대조가 이루어지며, 인증서 대조가 이루어 진 후에는 정상적으로 root권한을 회복할 수 있다.

해시값을 password로 한번더 묶은 이유는 기존의 password를 관리하는 방법 중 연구하여, 해커가 인증서 해시 값의 파일을 발견하더라도, password가 없는 이상 안의 내용을 확인할 수 없다는 전제하에 진행하였다.

5. 첨부 자료

5.1 셸 프로그래밍 소스

< 최초 메인 GUI셸코드&인증서 정보값 받는 셸코드 >

```
#!/bin/bash

gui=`zenity --title "관리자 로그인" --height=200 --width=300 --list --radiolist
--column "check" --column "menu" FALSE "인증서 발급" FALSE "인증서 로그인"`

if [[ $gui = "인증서 발급" ]]
then
    sh /home/centos/openssl/newuser.sh 2> /dev/null
    if [[ $? = '0' ]]
    then
        (
            rm -rf log.txt
            user=$(head -n 1 newuser.txt)
            cat newuser.txt >> $user.txt
            echo "60" ; sleep 3
            echo "# 정보 전송 중"; sleep 5
            sshpass -p centos scp -o StrictHostkeyChecking=no
            ./newuser.txt centos1@10.100.114.78:/home/centos1/openssl/user
            echo "100"; sleep 1
        )|
        zenity --progress --auto-close \
        --title="Information send" \
        --text="전송중...." \
        --percentage=0
        rm -rf newuser.txt
        rm -rf /home/centos/openssl/$user.txt
        check=$(find /home/centos/openssl -name log.txt)
        .
        .
    elif [[ $? = '1' ]]
    then
        zenity --error --text "초기화면으로"
        sh /home/centos/openssl/test.sh
    fi
```

< 발급받은 인증서, 비밀번호 입력해서 인증서 로그인하는 쉘 >

```
#!/bin/bash

SELECT=$(zenity --file-selection --title="인증서 선택")

case $? in
    1)
        zenity --error --title "취소" --text "초기화면으로"
        sh /home/centos/openssl/test.sh
        ;;
    0)
        sshpass -p centos scp -o StrictHostkeyChecking=no $SELECT
centos1@10.100.114.78:/home/centos1/openssl/user/
        PASS=$(zenity --title "인증서 로그인" --password)
        case $? in
            1)
                zenity --error --text "초기화면으로"
                sh /home/centos/openssl/test.sh
                ;;
            0)
                echo $PASS >> pass.txt
                sshpass -p centos scp -o StrictHostkeyChecking=no pass.txt
centos1@10.100.114.78:/home/centos1/openssl/user/
                rm pass.txt
                sleep 5
                check=$(find /home/centos/openssl/ -name success.txt)
                check1=$(find /home/centos/openssl/ -name fail.txt)

                if [ -f "$check" ]
                then
                    zenity --info --text="로그인 완료"
                    exit
                .
                .
                .
                fi
                ;;
            esac
        ;;
esac
```

<인증서 USB로 저장하는 쉘>

```
#!/bin/bash
while [ : ]
do
    lsusb > /home/centos/openssl/lsusb.txt
    check=$(cmp -i 18 /home/centos/openssl/lsusb.txt
/home/centos/openssl/pass1.txt)
    if [ -z "$check" ]
    then
        next=$(zenity --info --title "인증서 발급" --width=300 --text '<span
foreground="blue" font="14">\nUSB가 필요합니다.\t\t</span>\n<i>(USB삽입 후 확인
입력)</i>')
    else
        break
    fi
done
sleep 3
name=$(fdisk -l | tail -n 1 | awk '{print$1}')
mkdir /root/usb
mount $name /root/usb/
select=$(ls /run/media/root/)
move=$(find /run/media/root/ -name $select)
store=$(df -h | grep $name)
.
.
.
sh /home/centos/openssl/test.sh
fi
fi
```

< test.sh이랑 한쌍 정보입력창 쉘 >

```
#!/bin/bash
zenity --forms --title="인증서 발급" --text "정보입력"\
--separator="
" \
--add-entry="    이름    " \
--add-entry="    ID    " \
--add-password="    Password    (8글자 이상)    " \
--add-entry="    Country    (korea)    " \
--add-entry="    Student Number    (9xxxxxxx)    "
```

```
--add-entry="    Name    (honggildong)    " \
--add-entry="    Mail Address    (xxx@xxx.xxx)    " \ >> newuser.txt
```

< 기존 관리자 DB 비교 >

```
#!/bin/bash

host=`sed -n '1p' /home/centos1/openssl/user/newuser.txt`;

a=$(cmp/home/centos1/openssl/user/newuser.txt/usr/local/src/openssl/${host}.t
xt);
if [ -z "$a" ]
then
`sh /home/centos1/openssl/create.sh `;
echo "성공하였습니다." >> /home/centos1/openssl/log.txt;
else
echo "실패하였습니다." >> /home/centos1/openssl/log.txt;
`sshpass      -p      centos      scp      /home/centos1/openssl/log.txt
centos@10.100.114.85:/home/centos/openssl`;
fi
```

<인증서 생성>

```
#!/bin/bash

host=`sed -n '1p' /home/centos1/openssl/user/newuser.txt`;

# 파일 이름

L=`sed -n '2p' /usr/local/src/openssl/${host}.txt`;
.
.
.
#개인키 생성

cp /home/centos1/openssl/user/${host}.key /etc/pki/tls/private/${host}.key.enc;
openssl rsa -in /etc/pki/tls/private/${host}.key.enc -passin pass:${passwd} -out
/home/centos1/openssl/user/${host}.key ;

openssl      x509      -req      -days      1825      -extensions      v3_user      -in
/home/centos1/openssl/user/${host}.csr      -CA      /etc/pki/tls/certs/woanhon-rootca.crt
-CAcreateserial      -CAkey      /etc/pki/tls/private/woanhon-rootca.key      -out
```

```

/home/centos1/openssl/user/${host}.crt -extfile /etc/pki/tls/host_openssl.conf;

openssl md5 /home/centos1/openssl/user/${host}.crt
>>/usr/local/src/openssl/md5.txt;

#해시값 출력

openssl aes-256-cbc -pass pass:${passwd} -in /usr/local/src/openssl/md5.txt -out
/usr/local/src/openssl/md5.enc

#해시값 pass 압축

rm /usr/local/src/openssl/md5.txt

#삭제

mv /home/centos1/openssl/user/${host}.csr /usr/local/src/openssl/user/${host}.csr:#
인증서 요청서 서버로 옮기기
.
.
.

```

<인증서 대조>

```

#!/bin/bash

openssl aes-256-cbc -d -pass -f /home/centos1/openssl/user/pass.txt -in
/usr/local/src/openssl/md5.enc -out /usr/local/src/openssl/md5.txt

#해독

md5=`awk '{print$2}' /usr/local/src/openssl/md5.txt`;

# 해시값만 출력하기
.
.
if [ "$md5" = "$crt1" ]
then
echo "인증되었습니다." >> /home/centos1/openssl/user/success.txt;
`sshpass -p centos scp /home/centos1/openssl/success.txt
centos@10.100.114.85:/home/centos/openssl`;
else
echo "인증 실패 하였습니다." >> /home/centos1/openssl/user/fail.txt;
`sshpass -p centos scp /home/centos1/openssl/suceess.txt
centos@10.100.114.85:/home/centos/openssl`;
fi

```

5.2 발표 자료



📌 조원 소개

💡 팀 명 : 섭두리 (서버 두리)

조원	역할
박 현 민	팀장 및 인증 서버 구현 및 전체 시스템 구축
김 우 람	로그인 프로그램 및 취약점 분석 툴 구축
김 재 동	취약점 분석 툴 구축 및 자료 조사
안 지 훈	로그인 프로그램 및 취약점 분석 툴 구축
윤 조 영	취약점 분석 툴 구축 및 자료 조사
주좌훈	자료조사

💡 개발 환경 : Cent Os 7, 셸 스크립트

📌 주제 선정

- 주제 선정 이유

사용자가 로그인한 시간에 발생할 수 있는 취약점을 로그인할 때 가동되는 취약점 분석 툴을 통해 검사하고, 취약 항목을 관리자가 서버를 부팅하고 처음 화면에 보여줌으로 서버의 보안관리를 손쉽게 할 수 있다.
또 서버관리자가 서버 계정을 관리 하는데 있어 소홀해 질 수 있는 부분과, 기존의 아이디와 패스워드의 보안 취약점을 보완하기 위해 사내의 관리자를 구분하고 인증서를 발급하여 인증 과정에서 인증 할 수 있도록 함으로써 관리 계정의 보안을 강화 할 수 있다.

- 주제(프로젝트 요약)

- Root권한을 가지기 위해 로그인 후 인증서를 통한 인증을 한단계 더 거치게 된다.
- 로그인 후 root 권한을 가지게 된 서버의 취약점 분석 툴이 돌게 되고, 취약 항목만 관리자에게 알려 주게 된다.
- 취약 항목을 포함한 취약 툴 보고서는 로그 서버에 따로 저장된다.

주제 선정 이유1

기업 관리자 계정 지킴이 2차 인증수단 이젠 '필수'

입력일자 : 2014-04-10
 공유 : 0 | 댓글 : 0 | 추천 : 0

아이디와 패스워드로만 할 수 있는 환경, Oauth 인증수단 적극 검토해야

기업관리자 계정 지킴이 2차 인증수단 이젠 '필수'로 꼽히고 있다. 기업에 보안에 대한 관심이 높아져 가고 있고 이를 위한 인증수단도 다양해지고 있다. 보안 시스템들도 한층 업그레이드되고 있다.



아이디와 패스워드의 안전한 사용

그러나 최근의 개인정보 유출 사고를 보면, 패스워드 해킹은 빈번하게 발생하고 있다. 기업 관리자에게는 보안인증서명이 필수적인 사항이 되고 있다. 기업의 관리자 계정과 관리자 계정에서 접속이 가능하고 인증된 사용자만 접근이 가능하게 하는 것이 중요하다. 패스워드와 인증수단은 사고로 자주 발생하는데, 최근에는 관리자 계정에 유출된 비밀번호를 통해 아이디와 패스워드를 유출시키고도 발생한다. 문제는 관리자 계정의 비밀번호 유출로 유출된 비밀번호로 로그인 및 관리자 계정 유출과 같은 사고가 발생할 수 있다.

현재는 이러한 위험이나 패스워드의 유출이 발생한 것을 방지하기 위해 일어난다. 이는 관리자 계정에 로그인 할 수 있는 것은 관리자 계정 유출과 같은 것이다. 특히 관리자 계정에 로그인 할 때 관리자 계정 유출을 방지할 수 있으며, 보안등급이 매우 높은 기업이나 기관에서도 관리자 계정에 로그인하는 사고가 발생한다.



주제 선정 이유 2

취약점 관리 솔루션 시장, 후끈 달아오르나?

입력일자 : 2014-04-10

공유 : 0 | 댓글 : 0 | 추천 : 0

Qualys 사 국내 본격 진출...지니온 'Qualys Guard' 공식 론칭
 컴플라이언스 준수 비용 절감 위한 취약점 관리 솔루션 시장 기대

[eGSEC 2016] 투씨메시지, 보안 취약점 통합 관리 플랫폼 전시

입력일자 : 2016-04-10

공유 : 0 | 댓글 : 0 | 추천 : 0

보안 취약점 통합 관리 플랫폼 '투씨메시지' 소개

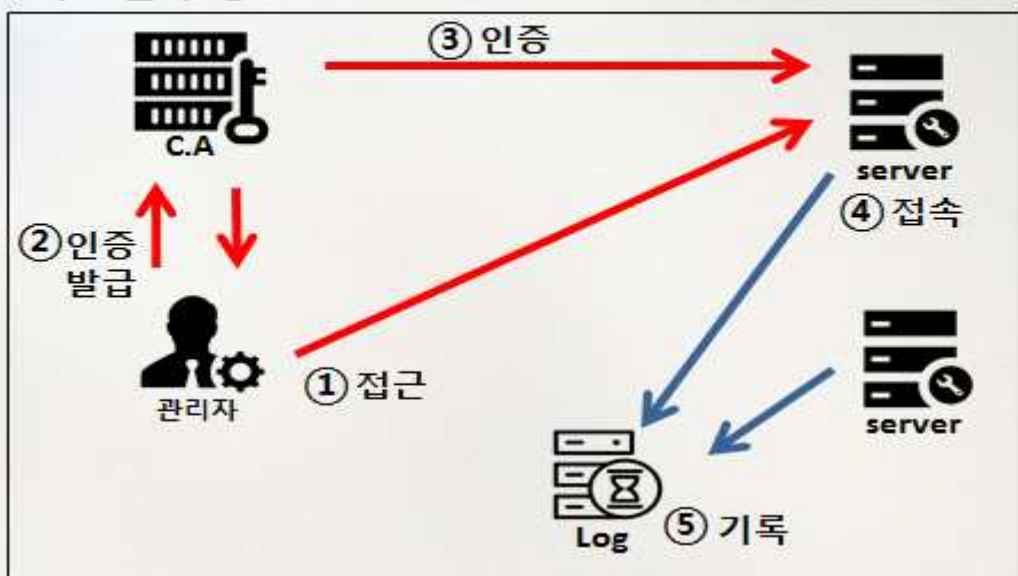
[보안뉴스 김대현] 3월 18일, 부산 컨퍼런스에서 개막한 '국제정보보안산업전 2016(eGSEC 2016, www.egsec.org)'에 참가한 투씨메시지는 자사의 보안 솔루션 브랜드 TSC Beyond the Security를 소개했다.

지니온 취약점 관리 컨설팅팀의 박관순 부장은 "기업이나 기관에서 취약점 관리가 필요한 이유는 해커의 위협으로부터 자산을 보호할 수 있는지 확인이 필요하기 때문이다. 현재 400,000개 이상의 알려진 취약점이 존재하고 하루에 10개 이상의 취약점이 발표된다"면서 "취약점은 지속적으로 증가하기 때문에 지속적으로 이를 제거하지 않으면 실제적인 위협에 대응할 수 없다. 모든 취약점을 다 제거할 수는 없지만 한정된 자원으로 효율적인 제거가 필요하다"고 설명했다.

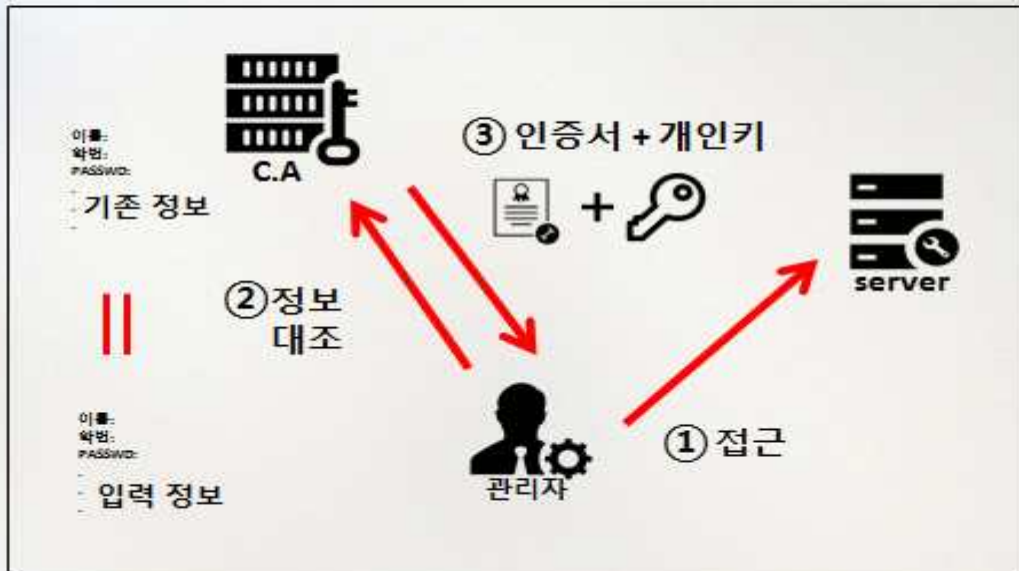
추진 일정

내용	월 별 진 행 계 획									
	3월			4월				5월		
	2주	3주	4주	1주	2주	3주	4주	1주	2주	3주
하위 서버 서비스 실행										
하위 서버 root 권한 제거										
메인 서버 로그인 웹 연구										
인증서 발행 연구										
인증서 로그인 방법 연구										
취약점 분석 결과 처리 문제										

시스템 구성도



시스템 구축 (인증서 생성)



시스템 구축 (인증서 생성)

The screenshot shows a user login interface with a dark background. The text "사용자 이름:" (User Name:) is displayed above a text input field containing the text "root". Below the input field are two buttons: "취소" (Cancel) and "다음" (Next). At the bottom of the interface, the text "사용자가 root권한으로 로그인 시도." (User attempts to login with root privileges.) is displayed.

시스템 구축 (인증서 생성)

아래 목록에서 항목을 선택하십시오.

check	menu
<input type="radio"/>	인증서 발급
<input type="radio"/>	인증서 로그인

인증서 발급 선택

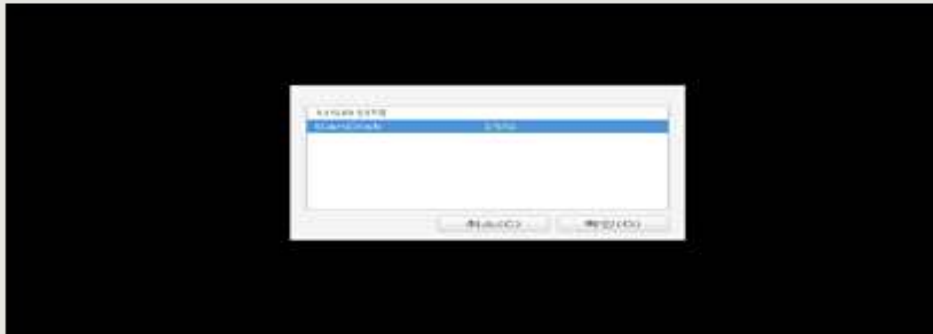
시스템 구축 (인증서 생성)

정보입력

이름	ahn
ID	mama2233
Password (8글자 이상)	●●●●●●●●
Country (korea)	KR
Student Number (9xxxxxxxx)	91317000
Name (honggildong)	min
Mail Address (xxx@xxx.xxx)	parkih@naver.com

서버 관리자 계정 사용자의 정보 대조

시스템 구축 (인증서 생성)



발급된 인증서는 usb에 보관 가능하다.

시스템 구축 (인증서 생성)



개인 정보가 불일치할 때

시스템 구축 (로그인)



시스템 구축 (로그인)

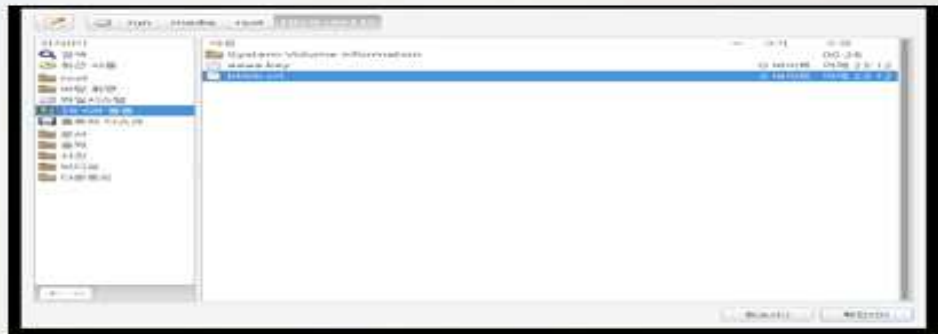
아래 목록에서 항목을 선택하십시오.

check	menu
<input type="radio"/>	인증서 발급
<input checked="" type="radio"/>	인증서 로그인

취소(C) 확인(O)

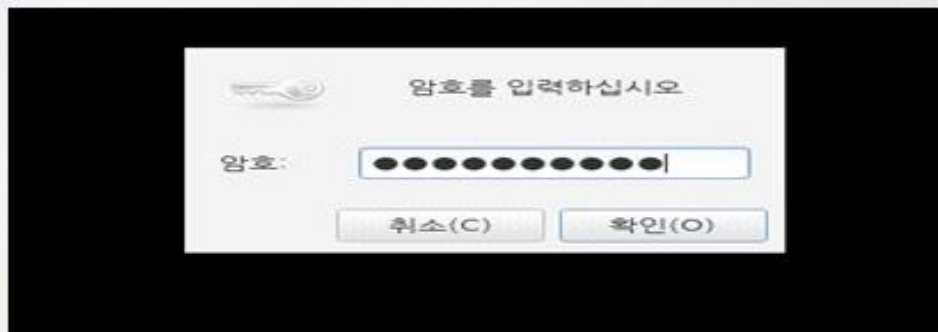
인증서 로그인을 선택한다.
로그인은 생성 후 바로 실행도 한다.

시스템 구축 (로그인)



인증서를 사용자 usb에서 찾는다.

시스템 구축 (로그인)



인증서에 대한 암호를 입력한다.

시스템 구축 (로그인)



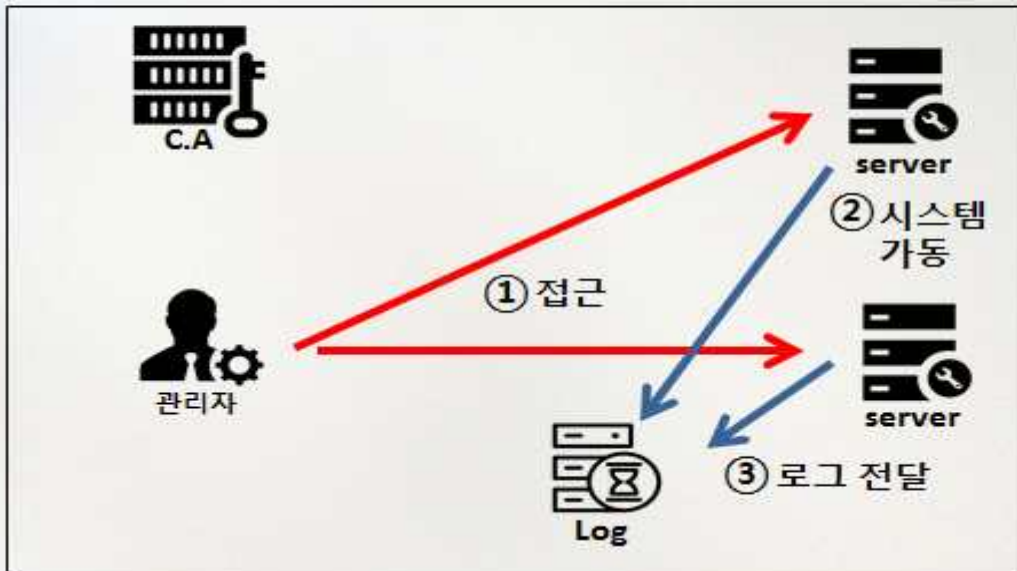
인증서 인증이 성공하면 서버를 사용할 수 있다.

시스템 구축 (로그인)



인증서 인증이 실패 할 경우 초기 화면으로 돌아간다.

시스템 구축 (취약점 분석 시스템)



시스템 구축 (취약점 분석시스템)

15. Apache 파일 업로드 및 다운로드 제한

16. Apache 웹 서비스 영역의 분리

로그 파일 확인

취약점의 총 개수는 18개 입니다.

취약점 부분만 지금 바로 보시겠습니까?

y

[취약] => 원격 접속 포트가 열려 있습니다.

[취약] => 패스워드가 너무 짧습니다.

[취약] => 계정 이름 일체값이 설정되어 있지 않습니다.

[취약] => 패스워드 파일이 보안에 취약합니다.

[취약] => 현재 패스워드 최소길이(5글자)로 너무 짧습니다.

[취약] => /etc/hosts 파일 소유자 설정이 취약합니다.

[취약] => /etc/xinetd.conf 파일 소유자 설정이 취약합니다.

[취약] => /etc/xinetd.conf 파일 권한 설정이 취약합니다.

[취약] => world writable 파일이 5개 이상 존재합니다.

[취약] => finger 서비스가 작동 중입니다.

[취약] => Anonymus Ftp 계정이 존재합니다.

[취약] => cron 파일의 소유자 및 권한 설정이 취약합니다.

[취약] => NFS 서비스가 작동하지 않습니다.

[취약] => autofs가 작동 중입니다.

[취약] => RPC 서비스가 작동 중입니다.

[취약] => Apache 웹 프로세스 권한이 취약합니다.

[취약] => Apache 상위 디렉토리 접근이 취약합니다.

[취약] => Apache 파일 업로드 및 다운로드가 취약합니다.


사용자 로그인 후
보이는 취약 항목


시스템 구축 (취약점 분석시스템)

```
2. 파일 및 디렉터리 소유자 점검.  
[양호]  
3. /etc/passwd 파일 소유자 및 권한 설정  
[양호]  
4. /etc/shadow 파일 소유자 및 권한 설정  
[양호]  
5. /etc/hosts 파일 소유자 및 권한 설정  
[취약] => /etc/hosts 파일 소유자 설정이 취약합니다.  
[양호]  
6. /etc/xinetd.conf 파일 소유자 및 권한 설정  
[취약] => /etc/xinetd.conf 파일 소유자 설정이 취약합니다.  
[취약] => /etc/xinetd.conf 파일 권한 설정이 취약합니다.  
7. /etc/services 파일 소유자 및 권한 설정  
[양호]  
8. SetUID, SetGID, Sticky bit 설정 파일 점검  
SetUID 설정 파일 점검  
합계 28  
폴디렉토리 /perm/suid 저장  
SetGID 설정 파일 점검  
합계 14  
폴디렉토리 /perm/sgid 저장  
Sticky bit 설정 디렉토리 점검.
```


로그 서버에 쌓이는
원본

결론


 CUI작업이 아닌 GUI 작업으로 셸을 처리

 취약점 항목


주요 정보 통신 기반 시설 취약점 분석 평가 보고, 취약점 분석 항목들 중 상급 항목으로 셸 코드 작성

 서버의 취약점

- 사용자가 로그인 한 직후 바로 볼 수 있게 취약 항목의 개수와 선택에 따라 항목을 구체적으로도 알 수 있다.

 인증 방식 강화

- 기존의 계정보안 관리에 있던 취약점을 인증서 인증으로 보완
- Openssl을 이용해 pki기반의 rsa암호화 알고리즘의 인증서를 구축

 십두리 만의 CA를 구축하고 root 역할을 해 개인키를 생성하고, 인증서를 발급 과정 연구



Q&A

5.3 참고 자료

* 인증과 공개키 기반구조 2011.11.22.

-중부대학교 정보보호학과 이병천 교수님

* <http://sol9501.blog.me/70105358014>

-x509 pki기반 알고리즘

* <http://firstboos.tistory.com/entry/openssl-101i-%EC%84%A4%EC%B9%98-on-CentOS-65>

http://zetawiki.com/wiki/리눅스_개인서명_SSL_인증서_생성#.EC.9D.B8.EC.A6.9D.EC.84.9C_.EC.83.9D.EC.84.B1

-openssl 개인키 생성하는 방법

- openssl업데이트

*취약점 분석 항목 가이드 2015년 판

-취약점 분석 항목

28일 동안 배우는 리눅스 서버 관리-카사노 히데마츠 저

유닉스 리눅스 시스템 관리자를 위한 쉘 스크립트 활용 가이드 - 정해주 저

뇌를 자극하는 리눅스 서버 & 네트워크- 우재남 저