Secure DNS의 기능확장을 위한 관리자 시스템의 설계

심희원*°, 김진성*, 심영철*, 임찬순**, 변옥환**
홍익대학교 컴퓨터공학과*, ETRI 슈퍼컴퓨터 센터**

The Design of Administrator Systems for Extending Secure DNS

Heewon Shim*°, Jinsung Kim*, Young-Chul Shim*, Chansun Lim**, Okhwan Byun**
Hongik Univercity Dept. of Computer Science*, ETRI. Supercomputer Center**

요 약

본 연구에서는 안전한 데이터 서비스를 제공하는 기존 Secure DNS를 확장시켜 GUI방식의 관리자 인터페이스를 설계하였다 따라서 시스템 설정에 대한 configuration과 보안관련 도구의 핵심이라 할 수 있는 로그의 관리가 용이해 졌다. 또한 관리자 인터페이스에 의해 각각의 자원레코드에 대한 자동적인 삽입, 삭제가 가능하며 암호화 알고리즘의 추가를 interactive하게 처리한다 그밖에 기존 Secure DNS에서는 새로운 암호화 알고리즘을 추가할 때마다 재 컴파일 해야하는 단점이 있다 이를 해결하기 위해 'Dynamic link Interface'를 설계하였다 이는 암호화 알고리즘의 입출력 표준을 정하고 이를 단일한 시스템 API로 구성하여 Secure DNS가 초기화 될 때 동적 라이브러리를 사용하여 각각의 암호화 알고리즘을 메모리에 적재하는 방식을 택한다. 그밖에 Secure DNS를 이용하여 제공할 수 있는 응용방안으로 개인의 공개키 분배서비스와 X.509 체계를 이용한 인증서를 제공하는 서비스를 제안한다. 따라서 본 연구에서는 인터넷의 기본 인프라스트럭쳐인 DNS를 최대한 활용할 수 있는 여러 가지 방안과 그 해결책을 제시한다

1. 개 요

인터넷의 보급이 전 세계적으로 확산되면서, 인터넷을 통한 다양한 기술들이 개발되고 있다 전자 상거래와 같은 기술은 근래 들어 연구되고 있는 가장 최신의 인터넷 기술중의 하나이다. 이런 인터넷을 통한 기술 중에서 빼놓을 수 없는 기술이 인터넷 보안 관련 기술이다. 인터넷은 기본적으로 개방성을 갖는 네트워크이므로 많은 사람들이 네트워크에 용이하게 접근할 수 있다. 이는 장점이 될 수도 있지만, 한편으로 생각하면 보안상 커다란 단점이 될 수도 있는 것이다. 인터넷 보안의 기술들은 이에 필요하고 개발되어져야 한다

최근 연구되어지고 있는 인터넷 보안 기술의 한 분야는 Secure DNS의 구현이다 Secure DNS는 여러 암호화 알고리즘을 이용하여 DNS가 각 호스트나 존의 개인키로 자원레코드를 서명하고, 서명된 SIG 자원레코드를 리졸버가 공개키에 의해 검증하는 부가적 보안서비스를 제공하는 DNS이다. 이러한 Secure DNS는 크게 3가지의 보안 서비스를 제공하게 된다. 첫째로는 인증된 공개키의 분배 서비스이고, 두 번째로는 데이터 근원에 대한 인증서비스이며, 마지막으로는 질의 또는 응답 메시지의 무결성을 제공하는 것이다 이러한 서비스를 제공하기 위해서 확장된 DNS는 3가지의 새로운 자원 레코드를 정의한다. 공개키의 분배와 저장을 위한 KEY 자원 레코드와 이러한 공개키에 의해 전자 서명된 SIG 자원레코드, 마지막으로 존재하지 않는 자원레코드의 인증을 위한 NXT 자원레코드 등이 이러한 Secure DNS의 핵심이다 [1][2][3][4]

본 연구에서는 이러한 Secure DNS의 구현중 무시되었지만 많은 비중을 차지할 수 있는 몇몇 issue에 대해 연구하고 또한 실제 설계와 구현을 하였다. 따라서 본 논문의 구성은 이러한 issue들은 하나의 단원으로 할당하여 다음과 같이 구성하였다.

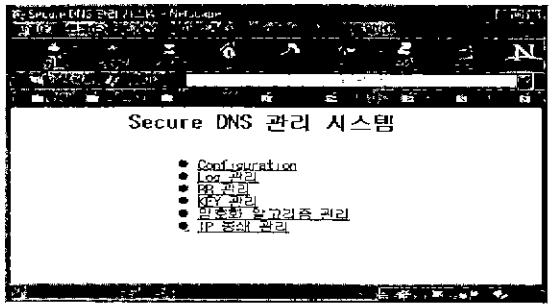
2장은 관리자가 GUI방식으로 DNS의 정책을 관리, 유지할 수 있는 관리자 인터페이스의 설계이며 이러한 관리자 인터페이스를 이용하여 수행 가능한 DNS configuration, 로그 관리, 자원레코드의 관리, KEY의 관리, 그리고 암호화 알고리즘의 관리등에 대해 세부적으로 살펴본다. 3장은 암호화 알고리즘과 Secure DNS API간을 'Dynamic link Interface'에 의해 연결하여 암호화 알고리즘이 추가될 때마다 재 컴파일 하는 오버헤드를 줄일 수 있는 방안에 대해 살펴보며, 4장에서는 이러한 Secure DNS를 이용하여 제공할 수 있는 보안관련 응용서비스에 대해 종합적으로 살펴보겠다 그리고 마지막 5장은 결론으로 본 논문을 끝맺도록 한다.

2. Secure DNS 관리자 인터페이스

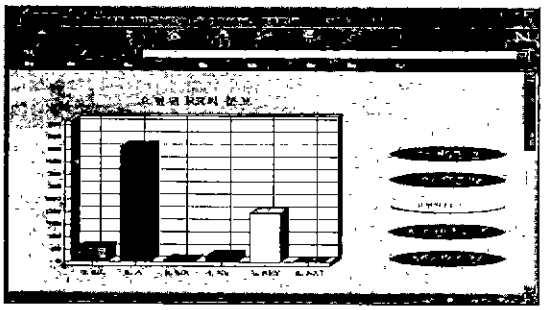
2.1. 관리자 인터페이스의 기능

관리자 인터페이스는 Web 브라우져를 기반으로 관리자와 interactive한 방식으로 여러 작업을 수행한다 로그 정보의 관리와 통계적인 로그 결과에 의해서는 DNS의 운용방침을 결정할 수 있는데 보안 도구의 핵심인 로그 시스템은 뒤에 자세히 살펴보도록 하겠다 그 이외에도 IP 봉쇄 관리와 KEY의 관리, 새로운 자원레코드의 추가등의 작업을 interactive하게 수행할 수 있다.

관리자 인터페이스의 세부적인 항목은 아래 그림과 같이 DNS 설정에 관련된 항목과 로그 관리, 자원 레코드의 관리, KEY 관리, 암호화 알고리즘의 관리, IP 봉쇄관리등이 있다



<그림 1> Secure DNS 관리자 인터페이스



<그림 2> 요청된 RR의 분포 로그

DNS configuration 항목은 로그 파일의 위치, 로그 출력방식, DNS 운영에 따른 configuration 파일을 접근하여 관리자가 직접 DNS를 interactive하게 설정할 수 있도록 한다.

자원레코드의 관리는 특정 자원레코드에 대한 삭제와 등록, UPDATE를 가능하도록 하며, 이에 따른 NXT 자원레코드의 자동 UPDATE를 가능하게 한다. KEY는 새로운 호스트가 추가될 때마다 그에 수반되는 모든 자원레코드를 자동으로 추가시키며, 이에 따른 KEY 자원레코드를 파일로 입력받거나 개인의 암호로 공개키의 개인키쌍을 직접 생성시킬 수 있도록 한다

암호화 알고리즘의 관리는 'Dynamic Link Interface'서 정의된 DB 파일에 접근하여 사용 가능한 알고리즘과 그 번호를 출력한다. IP 봉쇄관리는 특정 IP에 대해 Secure DNS 서비스를 봉쇄하는 등의 정책을 수행할 수 있도록 지원한다.

2.2. 로그 시스템

보안관련 도구의 가장 큰 특징은 로그의 관리이다. 기존 Secure DNS는 로그정보의 중요성에도 불구하고 이러한 로그 시스템의 구현은 되어있지 않은 상태이다. 로그 정보는 Secure DNS의 운영정책에도 많은 역할을 끼칠 수 있다. 실패한 질의의 분포를 분석하여 접근허가 리스트를 관리할 수도 있으며, 보안사고 이후리도 로그 정보를 분석하여 파급효과를 줄일 수 있다. 이러한 로그는 또한 요청된 자원레코드의 분포등과 같은 통계적 결과를 그래프 형태로 출력할 수 있다.

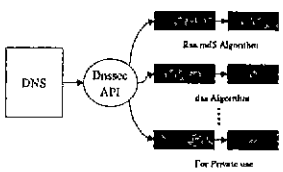
보통 로그정보는 평균 1초에 5번의 질의가 있을 때 한 개의 로그당 200-500byte가 소요되므로, 엄청난 디스크 용량이 필요하다. 따라서 효율적인 로그관리를 위해서 제한된 로그 정보의 정의가 필요하다 성공적인 질의에 대한 로그는 질의한 주체의 IP(4byte). 요청한 자원 레코드 종류(4byte), 시간(HH:SS). 성공여부 플래그(1, 1byte) 등을 저장한다 반면 실패한 질의에 대한 로그는 성공적인 질의보다 비교적 많은 정보가 필요하다. 따라서 접근한 주체의 IP(4byte), 목적지의 도메인 네임(255byte), 요청한 자원레코드 종류(3byte), 오류코드(2byte), 시간(HH:SS), 성공여부 플래그(0, 1byte)등이 로그 정보로 필요하다. 모여진 로그정보는 체계적인 관리가 필요하다 또한 로그 정보를 분석하여 부분적인 DNS 정책에 적용이 가능하다. 일단 모여진 로그 파일은 1일 단위로 24시에 파일을 닫는다 이렇게 닫혀진 파일은 분석기에 의해 다시 열려져 분석된다. 분석기는 전체 질의의 개수, 실패한 질의의 개수, 요청된 자원레코드의 분포, 질의의 시간별 분포, 실패한 질의에 대한 IP분포와 횟수, 날짜등을 분석 파일에 저장한다. 분석 작업이 끝난 로그 파일중 성공한 질의에 대한 항목은 디스크의 용량 낭비를 막기 위해 삭제되지만 성공한 항목은 보안사고 이후에 중요한 자료가 되므로 남겨둔다. 분석기는 1일, 1달, 1년 단위로 분석 파일을 재 분석하여 통계적 정보를 파일에 출력한다.

이렇게 관리된 통계정보는 관리자의 요청에 의해 그래프 형태로 화면에 출력하거나 파일에 출력할 수 있으며, 부가기능으로 이러한 통계정보에 의해 실패한 질의의 횟수가 많은 호스트의 IP는 DNS에 접근하는 것을 봉쇄할 수도 있다.

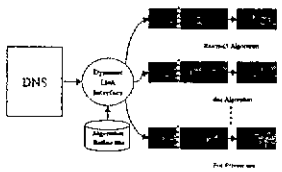
로그 정보는 bind의 내부에서 처리된 결과를 돌려줄 때 가로채어 필요한 정보를 파일에 남기도록 한다. 필요한 정보는 앞서 말한바와 같이 주체의 IP, 목적지의 도메인 네임(255byte), 요청한 자원 레코드 종류, 시간, 성공여부(오류코드)등을 저장하여야 한다. 이들을 얻기 위한 함수는 Secure bind에서 기본으로 제공하므로 이를 이용하여 로그를 남긴다.

3. Dynamic Link Interface

현재 RFC2065에서 정의된 공개키의 암호화 알고리즘은 얼마되지 않는다. 실제 Secure DNS에 포함되어질 암호화 알고리즘은 RSA/MD5, Diffie-Hellman, DSA, Elliptic curve등과 간접키, 사용자 정의키등이 있다.[3][4][8][9] 하지만 일라되지 않은 암호화 알고리즘에 비해 사용할 수 있도록 예약되어진 알고리즘 번호는 0~255까지로 총 256개이다. 이중 위 암호화 알고리즘과 기타 예약되어진 것을 제외해도 5~251번까지의 암호화 알고리즘이 사용 가능하다. 이는 앞으로 여러개의 암호화 알고리즘이 더 추가되어질 수 있다는 것을 의미한다 만약 Secure DNS가 암호화 알고리즘의 확장성을 고려하지 않고 설계되어졌을 경우, 전 세계에 있는 수많은 DNS가 알고리즘이 추가될 때마다 새 컴파일을 해야 할 것이다. 이에 본 연구에서는 확장성을 고려하여 Secure DNS의 암호화 알고리즘을 구현하는 방안을 제안하려 한다



<그림 3> 기존 암호화 알고리즘의 운영방식



<그림 4> 수정된 Dynamic Link 인터페이스

3.1. 기존 Secure DNS의 암호화 알고리즘 모듈구성

기존 암호화 알고리즘의 운용방식은 위의 같이 각각의 암호화 알고리즘에 독립적으로 Dnssec API를 이용하여 DNS는 각각의 암호화 알고리즘을 사용할 수 있다. 즉 RSA의 경우 직접 rsaref의 세부적인 함수를 사용하지 않고도 적판적으로 DNS에서 한정된 함수들을 이용하여 암호화 알고리즘 모듈을 이용할 수 있다 또한 각각의 암호화 알고리즘은 DNS에 사용될 목적으로 만들어진 알고리즘이 아니므로 DNS 목적에 맞도록 API와 세부 알고리즘간의 lnk layer를 두었다. 이러한 배려에도 불구하고 이 방식의 단점은 알고리즘의 추가시마다 재 컴파일을 해야 한다는 것이다.

3.2. 확장성을 고려한 Dynamic Link 인터페이스

수정된 알고리즘은 위와 같이 암호화 알고리즘을 lnk와 연결하여

하나의 모듈로 만든다 이 모듈은 Dynamic Link Interface에 의해 하나의 독립된 프로그램으로서 실행되며 DNS에서는 이러한 암호화 알고리즘을 호출하는 API를 표준화시켜 암호화 알고리즘의 앞부분에서 이들 API를 해석한다

```
-int  dnssec_check()  Function to check if certain alg is suppored
  ex) switch(alg) { .. case KEY_RSA: return 1 ..
-int  dnssec_sign() :  Incremental signing routine takes flags to
                       select which steps to perform
  ex) switch(key1->alg) {    case KEY_RSA: rsa_cmp( ....
- int  dnssec_verify()  Incremental verify routine
  ex) switch(key1->alg) {    case KEY_RSA: rsa_sign( ....
- int  dnssec_genkey()  Function to generate new KEY for dns
                        : rsa_verify등의 rsaref_link에 정의된 함수를 사용한다.
- KEY *dnssec_getkey()  Function to retrive private KEY from storage private
                        keys are stored internaly and are only decoded once
- void dnssec_putkey()  Function to write out a private key
- KEY *dnssec_pubkey()  Function to convert KEY RR into a KEY structure
- KEY *dnssec_readpubkey()  Funcation that reads in public key
- int dnssec_writepubkey()  Function to write out public key
- int dnssec_encpubkey(). Function to return a public key in DNS format base64 encoded
- void dnssec_freekey()  Releases all memory pointed to by key structure
```

<표 1> 암호화 알고리즘 표준 API

DNS에서 필요로 하는 암호화 알고리즘 표준 API는 위의 <표 1>과 같다. 이러한 표준 API는 Secure DNS에 의해 호출되어 Dynamic link 인터페이스(이후 DLI)를 통과하여 암호화 알고리즘과 표준화된 통신규약에 의해 직접 각 알고리즘의 공통 API에 전달된다. 이때 세부적인 KEY나 SIG의 구조체는 표준 구조체로 캐스팅되어 호환성을 유지한다. 각 암호화 알고리즘의 link는 이러한 API를 해석하여 세부적인 암호화 알고리즘의 함수들을 호출하여 결과를 리턴한다.

암호화 알고리즘을 동적으로 연결하기 위해서는 DNS에서 세부 암호화 알고리즘의 번호를 DLI로 전달하면 DLI는 특정 암호화 알고리즘이 있는 파일을 Algorithm define DB에서 찾아내고 그 파일을 지정하여 동적 라이브러리를 메모리에 적재한다.

Algorithm define DB의 구성은 알고리즘 번호와 그 알고리즘이 제공하는 암호화 함수의 이름을 가지고 있으며, 알고리즘 번호를 return한다. 따라서 DLI는 이 이름으로 암호화 알고리즘을 호출할 수 있다 예를들어 RSA의 key를 생성시키는 작업을 수행하려면 DNS는 DLI에 RSA의 알고리즘 번호와 KEY를 생성하라는 액션코드를 보낸다 DLI는 알고리즘 번호를 Algorithm Define DB에서 찾아 RSA 알고리즘을 수행하는 동적 라이브러리 파일을 찾아낸다. DLI는 찾아낸 동적 라이브러리 파일을 호출하여 표준 프로토콜을 이용하여 공통 API에 dnssec_genKey()를 호출한다 특정 알고리즘 모듈은 공통 API에서 넘어온 함수 이름을 RSA_keygenerate()로 바꾸어 RSA 암호화 알고리즘을 호출하여 결과를 받아온다. 결과는 역시 공통 API를 통해 DLI로 전달되고 캐스팅에 의해 원하는 결과를 DNS에 리턴한다.

다음은 기존 DNSSEC_API의 KEY를 생성하는 역할을 수행하는 함수로 동적인 Link 인터페이스에 사용될 code의 일부분이다.

```
int dnssec_genKey (name, bits, exp, flags, protocol, alg)
{ ...
  if (!exits_alg(alg)) {     : return 0; }
  else {
    libName = search_ADDB(alg),
    gLibDLL = LoadLibrary(libName),
    if (gLibDLL != NULL) { . ;
      getKey = (KEY)GetProc(gLibDLL, GENKEY),
      ... ; }
. .
```

<표 2> Dynamic Link Interface code segment

4. Secure DNS의 응용방안

Secure DNS는 본래의 기능인 네이밍 서비스외에도 KEY 자원레코드의 저장과 분배의 역할을 수행한다 하지만 이러한 DNS에서 사용되는 KEY는 호스트 level의 KEY이므로 활용도가 그리 높지 않았다. 따라서 이러한 KEY자원 레코드를 user level의 공개키 저장이 가능하도록 재정의 해야 한다. 개인에 대한 명세는 TXT 자원 레코드를 이용하고 KEY의 식별자는 개인의 e-mail로 한다

Secure DNS는 KEY 자원 레코드의 분배뿐만 아니라 사용자의 KEY지원 레코드를 제 3자에게 인증하는 인증서를 제공할 수 있다 현재 CERT라는 자원 레코드를 이용하여 이를 수행하고 있으며 이는 전자상거래에 직접 활용할 수 있음을 의미한다.[5] 이때 많은 사용자들은 자신의 공개키를 직접 DNS 관리자에게 전달하기 보다는 on-line상에서 등록하고 싶어할 것이다. 따라서 안전한 channel이 없는 on-line상에서 사용자의 공개키를 DNS에 등록하려면 그 사용자는 Secure DNS에 등록한 호스트에서 등록을 해야만 할 것이다. 이때 보안상 필요한 내용은 등록하고 있는 호스트의 이름과 사용자의 이름, 공개키, 그리고 그밖에 자세한 명세만이 필요할 뿐이다 또한 메시지에 전자서명을 하여 응답 메시지의 무결성을 제공하며 현재 사용되고 있지 않은 Secure DNS의 기밀성 서비스를 이용할 수도 있다. 이후에 DNS 관리자는 로그에서 호스트 위치를 파악하여 사용자의 진위성을 판단할 수 있다.[7]

5. 결론

본 연구에서는 안전한 네이밍 서비스를 제공하는 SecureDNS에 부가적인 기능을 부여하여 DNS를 확장시키거나 관리자와의 인터페이스를 제공하는 방안에 대해 연구하였다.

DNS는 Internet 환경의 가장 중요한 인프라 스트럭처이며 어떤 응용 프로그램보다 가장 많은 활용도를 보이고 있다. 이러한 DNS에 보안상의 해결책을 제시하는 것만으로 그치지 않고 이를 활용하여 인증서의 배분이나 user-level의 key의 분배등의 문제에 사용하면 DNS의 특성상 많은 장점이 있을 것이다. 따라서 이러한 서비스를 제공하려면 관리가 용이하고 online상으로 등록이 가능해야 하며, 사용자나 관리자에게 편의를 제공하는 인터페이스가 필수이다 본 연구에서는 이러한 DNS의 활용을 가능하도록 다음의 기능을 확장시켰다

확장된 기능은 크게 3가지로 나뉜다 첫째로는 관리자가 GUI환경에서 DNS를 쉽게 관리할 수 있도록 하는 환경을 제공하는 것이며, 이를 이용해 이전의 DNS보다 복잡해진 자원레코드의 관리를 interacuve하게 관리할 수 있다. 또한 로그 시스템의 도입으로 Secure DNS의 운용에 적용할 수 있게 되었다. 둘째로는 알고리즘 추가시마다 컴파일 해야하는 오버헤드를 줄이기 위해 Dynamic Link 인터페이스를 제공하는 것이고, 마지막으로는 SecureDNS를 이용하여 user-level의 key 분배서비스나 인증서등에 이용하는 활용방안에 대해 살펴보았다

참고 문헌

[1] RFC 1032 : M.Stahl, "Domain administrators guide", 1987 11
[2] RFC 1033 : M.Lottor, "Domain administrators operations guide", 1987 11
[3] RFC 2065 : D.Eastlake, C.Kaufman, "Domain Name System Security Extensions", 1997 3
[4] Draft : D.Eastlake, "Domain Name System Security Extensions", 1998 3
[5] Draft : D.Eastlake, Olafur Gudmundsson, "Storing Certificates in the Domain Name System", 1998 3
[6] Draft : D.Eastlake, "RSA/MD5 KEYs and SIGs in the Domain Name System", 1998 1
[7] Draft : D.Eastlake, "DNS Operational Security Considerations", 1998 3
[8] 한국정보보호센터, "정보보호현황 -정보보호 기관 및 기술을 중심으로", 1996.10
[9] 한국정보보호센터, "정보보호총서", 1996 12