DOI: 10.3745/KIPSTC.2008.15-C.4.227

Recursive DNS의 캐쉬 정보 신뢰성 향상 기법

주 용 완[†] · 이 응 재[†] · 남 광 우[††]

요 약

인터넷 인구가 폭발적으로 증가하면서 인터넷 접속 관문이라 할 수 있는 DNS 정보의 위·변조시 엄청난 사회·경제적 피해를 양산할 수 있다. 그러나 DNS 정보의 보호를 위한 기존의 연구들은 현재 구축되어 있는 DNS 시스템 버전의 한계, 관리 부하의 증대 등으로 인하여 실세계 적용에 한계를 보이고 있다.

이 연구는 현재의 DNS 서비스 환경에서 DNS 정보에 대한 위·변조 공격 방법을 분석하고 독립적으로 DNS 정보의 오염 여부를 실시간으로 탐지할 수 있는 방법을 제안한다. 또한 외부 공격에 의한 오염된 DNS 정보뿐만 아니라 정보 획득 시간이 오래되어 유효하지 않은 DNS 정보의 수정을 위하여 Recursive DNS에서 관리되는 캐쉬 정보에 대한 사후 검증을 통하여 Recursive DNS의 신뢰성을 향상시키는 방법을 제안한다.

키워드: 재귀적 DNS, 도메인 네임 서버, 캐쉬 오염 공격, 캐쉬 신뢰도

Cache Reliability Enhancing Method for Recursive DNS

Yong Wan Ju[†] · Eung Jae Lee[†] · Kwang Woo Nam[††]

ABSTRACT

As the internet users rise up rapidly, DNS information forgery can cause severe socio-economic damages. However, most studies on DNS information security reached the breaking point in applying to actual circumstances because of the limit of existing DNS system version, the increasement of management burden and etc. The paper proposes the real-time method for detecting cache poisoning of DNS system independent of analysing the DNS forgery types in the current DNS service environment. It also proposes the method of enhancing the reliability for the cache information of Recursive DNS system by post-verifying the cache information of the DNS system.

Key Words: Recursive DNS, Domain Name Server, Cache Poisoning, Cache Reliability

1. 서 론

인터넷은 1969년 미국방성 고등연구계획국(ARPA: Advanced Research Projects Agency)에 의해 개발된 컴퓨터 네트워크인 ARPANET이 그 시초로서 미국의 연구소와 대학들 간의 컴퓨터를 연결하여 지역적으로 분산되어 있는 방대한 자원을 공유하여 활용할 목적으로 개발되었다. 이후 지식 정보화 사회를 선도하는 핵심수단으로 평가받으며 2004년 기준 전 세계 약 8억 6천만 명의 이용자와 약 2억 6천만 개의 호스트로 연결된 세계 최대의 정보통신네트워크로 자리 잡고 있다[1]. 우리나라의 경우 2005년 말을 기준으로 초고속 인터넷 가입자 수가 약 1천 2백만 명에 달하고 있으며[2], 만 6세 이상 전체 인구의 72.8%인 약 3천 3백만 명이 인터넷을 이용[3]하는 등 인터넷은 우리 생활의 중요한 일부분으로 자리 잡고 있다.

이러한 인터넷은 계속적으로 이용자가 증가하고 인터넷을 중심으로 한 통신, 방송의 융합으로 신규 서비스들이 도입되고 있어 안정적인 인터넷 서비스 제공을 위한 인터넷 기반 시설의 안정성에 대한 중요성도 더욱 커지고 있다. 따라서 현재의 인터넷 서비스에 대한 기반 시설의 안정성 강화 차원에서의 ISP(Internet Service Provider) 기준 강화, IDC(Internet Data Center) SLA(Service Level Agreement) 기준 제시, 그리고 차세대 네트워크 차원에서의 광대역 네트워크에서 분야별 안정성을 지속적으로 강화하는 방안들이 제안되어 구현되고 있다.

※ 본 연구는 건설교통부 첨단도시기술개발사업-지능형 국토정보기술혁신 사업과제의 연구비지원(07국토정보C05)에 의해 수행되었습니다.
† 정 회 원: 한국인터넷진흥원 정책기획단
†† 정 회 원: 군산대학교 컴퓨터정보공학과 조교수(교신저자)
논문접수: 2008년 1월 4일
수 정 일: 2008년 4월 29일
심사완료: 2008년 5월 1일

위와 같은 인터넷 서비스의 안정성 강화를 위한 주요한 요소로서 DNS 시설과 DNS에 포함되어 있는 정보에 대한 보안성을 확보하는 등 인터넷의 주요 기반 시설에 대한 보호[4]가 매우 중요하다. 특히 DNS 정보 보호는 2002년에 발생한 전세계 루트 DNS들에 대한 공격으로 인해 그 중요성이 부각되었다. 현재 포털, 교육, 오락 등에 대한 인터넷 매개 서비스의 경우 전 세계적으로 인터넷 이용자를 대상으로 글로벌 서비스를 제공하며 이러한 서비스들은 value-chain화 되어 있어 인터넷 매개 서비스의 접속 관문이라 할 수 있는 도메인 접속 장애 시 엄청난 사회적·경제적 피해를 양산할 수 있다. 이러한 대표적인 사례가 바로 2002년 2월 미국의 Yahoo, eBay, CNN 등에 대한 해커의 집중 공격으로 직·간접적인 피해액이 약 2조 6천억 원으로 추산되었던 경우를 들 수 있다.

국가 인터넷의 주요 기반시설 중의 하나인 DNS의 정보의 위·변조를 방지하여 DNS의 보안성을 향상시키려는 연구는 크게 두 가지 방향으로 진행 중에 있는데 그 중 하나는 DNS 정보에 대한 전자서명 체계를 도입한 '공개키 기반구조(PKI: Public Key Infrastructure)'를 이용하는 DNSSEC(DNS Security Extensions)에 대한 연구로 인터넷에 관한 국제표준화 기구인 IETF(Internet Engineering Task Force)를 중심으로 진행 중이고, 최근에는 Yuan 등이 Recursive DNS 간의 상호 신뢰를 기반으로 협업에 의하여 DNS 공격을 탐지하는 방법[5]을 제안하였다.

그러나 이러한 연구들은 암호화 기법의 적용과 상호 협력에 기반을 두고 있어서 DNS 버전업의 필요, 시스템 성능저하 유발, 관리 부하의 증대 등으로 실세계 적용에 많은 문제점들이 제기되고 있다.

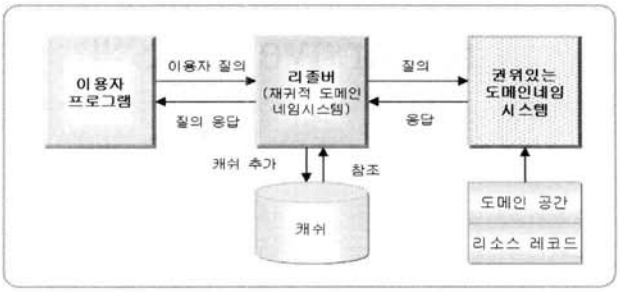
이 논문에서는 DNS 서비스 체계상의 시스템간 수평적, 수직적 상호 의존성으로 인하여 발생하는 문제점들을 개선하고, Recursive DNS가 관리하는 캐쉬 정보의 신뢰성을 강화할 수 있는 방안을 제시한다.

논문의 구성을 살펴보면 2장에서는 DNS 서비스 상의 시스템 체계 및 취약성을 살펴보고 DNS의 캐쉬 정보 신뢰성 향상에 관한 연구들을 기술한다. 3장에서는 캐쉬 오염 공격을 사전에 탐지할 수 있는 기법과 캐쉬 정보의 사후 신뢰성 검증을 위한 기법을 제안한다. 4장에서는 이 논문에서 제안하는 기법의 성능을 평가하고, 마지막으로 5장에서는 결론을 맺는다.

2. 관련 연구

2.1 DNS 서비스 체계

DNS는 인터넷상의 통신기기의 상호간의 인식을 위한 식별체계인 IP(Internet Protocol)주소와 사람이 인식하기 쉬운 도메인 네임을 연결하는 맵핑 기능을 제공하는 계층적인 구조로 구성된 거대한 글로벌 데이터베이스이다[6, 7].



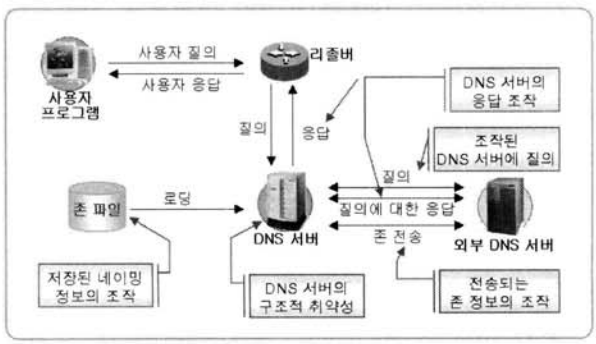
(그림 1) DNS 서비스 체계도

이는 (그림 1)과 같이 계층적으로 구성되어 하위단의 주소 정보를 가지고 있는 Authoritative DNS와 서비스적 차원에서 이용자로부터 질의를 받아서 Authoritative DNS에 계층적인 질의를 수행하고 최종 답신 전달해 주는 Recursive DNS로 구성된다. 그리고 이러한 Recursive DNS는 효율적인 DNS 질의·응답 서비스를 수행하기 위하여 한번 조회한 DNS 정보를 캐쉬 데이터베이스에 저장하여 관리한다.

2.2 DNS 서비스의 취약성

DNS는 개발 당시부터 오픈 서비스를 지향하고 있어 보안을 고려하지 않았기 때문에 끊임없이 보안의 위협성에 대한 논의가 제시되어 왔다. (그림 2)는 이러한 DNS에서 발생할 수 있는 취약성을 보여준다[8].

(그림 2)와 같이 DNS의 취약성은 DNS의 구조적 취약성, DNS의 응답 조작, 조작된 DNS로의 질의, 저장된 네이밍 정보의 조작 등으로 나누어진다. DNS의 구조적 취약성은 DNS 질의를 처리하는 방법 및 DNS 서버의 구조적인 문제로 인하여 발생하며, DNS 응답 조작은 질의 데이터가 전송되는 과정에서 해커의 데이터 조작에 의해 발한다. 조작된 DNS 및 저장된 네이밍 정보의 조작은 네임 서버 자체를 해킹함으로 인해 발생되는 문제를 의미한다. 이 논문에서는 특히 서비스형 시스템인 Recursive DNS에 초점을 맞추며 이에 대한 DNS의 구조적 취약성과 저장된 네이밍 정보의 조작을 중점적으로 다룬다. 또한 DNS 시스템에 대한 물리적인 접근에 의한 정보 조작과 DNS 서버의 해킹 등에 의한 도메인 정보 조작은 DNS 서비스 영역 외적으로 발생할 수 있는 취약성이므로 이 논문에서는 논하지 않는다.



(그림 2) DNS 서비스의 구조적 취약성

2.3 기존연구 고찰

DNSSEC은 IETF의 DNSEXT 워킹그룹에서 DNS의 취약성을 극복하기 위하여 DNS 데이터에 대한 인증과 무결성 서비스를 제공하여 DNS에 보안요소를 추가 확장하는 것으로 이를 위해서 DNSSEC 프로토콜은 새로운 자원 레코드 유형의 정의와 각 구성요소들의 안전한 상태(Secure Status)에 대한 요구사항들을 정의한다. DNS 보안 확장기술은 RFC2535에서 기본적인 내용이 기술되어 있으며 표준화 대상이 계속 추가되면서 수정·보완이 진행되고 있다[9].

DNSSEC은 Authoritative DNS에 설정된 권한 있는 데이터에 대하여 공개키 암호 방식의 전자서명 체계를 적용함으로써 Recursive DNS 데이터가 전달되는 과정에 있어서 DNS 데이터의 위·변조 여부를 Recursive DNS가 서명검증을 통해서 검증할 수 있는 보안 체계를 구현하는데 있다. DNSSEC 표준은 RFC4033, RFC4034, RFC4035의 3개의 문서로 채택되어 자세히 기술되어 있다.

DNS 서비스의 기반이 되는 소프트웨어들 중에서 BIND의 경우 이러한 표준들을 수용하여 BIND 버전 8.2부터 DNSSEC을 어느 정도 지원하였으나 BIND9에 와서 제법 사용이 가능한 수준으로 구현되었다[10, 11, 12, 13].

DNSSEC은 이러한 공개키 암호 기반의 전자서명 체계의 도입을 위하여 DNS의 리소스 레코드(이하 RR)에 새로이 DNSKEY(DNS Public Key), RRSIG (Resource Record Signature), DS(Delegation Signer), NSEC(Next Secure) 등 총 4개의 RR을 추가적으로 정의하였다.

RRSIG RR의 서명과 서명대상 DNS 데이터를 가지고 서명검증을 하기 위해서는 이 서명에 사용된 개인키에 대응하는 공개키가 공개 배포되어야 하는데 이 공개키는 DNSKEY RR에 설정되어 배포되고 이때 공개키의 소유 주체는 해당 도메인 자체가 된다. 즉, 도메인 영역 단위로 공개키가 배포된다는 것을 의미한다.

이러한 DNSKEY RR에 설정된 공개키 역시 그 자체에 대한 위·변조 가능성이 존재하므로 이 공개키에 대한 인증이 필요하게 되는데 이를 위해서 DNS 서비스 체계상에 인증체인이 형성될 필요가 있으며 이 인증체인은 하위 도메인 영역의 DNSKEY RR에 대한 상위 도메인 영역에서의 DS RR을 사용하여 구성한다. DNSSEC에서의 부모 도메인과 자식 도메인간의 DNSKEY RR과 DS RR을 사용한 인증관계는 DNS 구성 체계를 따라서 하나의 트리 구조를 인증체인으로 형성하는데 이상적으로는 최상위 루트 도메인 영역에서부터 끝까지 이어지는 인증체인이 형성될 수 있다.

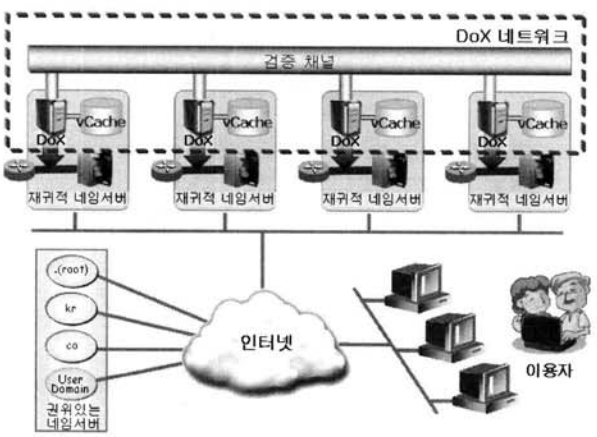
그러나 암호화되어 추가된 DNS 정보로 인하여 각 DNS들이 처리하여야 할 데이터량이 급격히 증가하기 때문에 이를 최상위 루트 DNS에 적용하였을 때 시스템의 성능이 저하되는 문제가 발생한다. 또한 DNS 정보가 변경될 때 마다 DNS 정보의 기본 단위인 Zone 파일 전체에 대하여 전자서명을 매번 갱신하여야 하고, .kr, .com, .net, .org와 같이 대량의 정보를 가지는 zone 파일에 서명할 때에 많은 양의 연산을 필요로 하기 때문에 전체적인 데이터 처리시간이 증가하여 DNS의 응답 속도를 저하시키는 문제를 발생시킨다. 그리고 DNSSEC의 적용을 위해서는 모든 레벨의 DNS들에 대하여 BIND 버전 수정 등의 작업이 필요하기 때문에 실제 적용을 위해서는 많은 시간이 요구된다.

Recursive DNS에 대한 캐쉬 오염 공격 방법은 하나의 DNS에 대해서는 쉽게 공격이 이루어지지만 다수의 DNS들에 대하여 동시에 공격하는 것은 어렵다. 즉, DNS를 구성하는 시스템들은 각기 다른 운영체제와 DNS 프로그램을 사용하고, 매우 짧은 시간에 DNS를 공격해야 하기 때문에 여러 개의 DNS를 동시에 공격하여 캐쉬 정보를 오염시키는 것은 매우 어렵다는 것이다.

Lihua Yuan[5] 등은 이러한 DNS의 특성을 이용하여 여러 개의 Recursive DNS들이 상호 협업하여 캐쉬 오염 공격을 막는 Domain Name Cross Referencing(DoX) 시스템을 제안하였다.

DoX 시스템은 기존의 DNS에는 어떠한 수정도 가하지 않는 클라이언트 측의 솔루션으로 Recursive DNS가 송·수신하는 DNS 질의 및 응답 메시지를 모니터링하여 캐쉬에 저장되는 DNS 정보를 검증하는 체계이다. 그리고 'DoX 네트워크'라 불리는 캐쉬 오염 공격에 대응하기 위하여 협업하는 시스템들은 서로 Peer-to-Peer (P2P)로 연결되어 각자가 보유하고 있는 DNS 캐쉬 정보를 공유한다.

아래의 (그림 3)은 DoX 시스템의 전체적인 구조를 보여준다. 그림과 같이 협업하는 시스템들의 모임인 'DoX 네트워크'를 구성하는 Recursive DNS에서 DoX 시스템은 DNS와 Authoritative DNS 시스템 간의 질의 메시지를 모니터링하여 질의에 대한 응답이 올바른지를 검사한다.



(그림 3) DoX 시스템의 구조

이러한 DoX 시스템들은 서로 P2P로 연결되어 있으며, 대량의 DNS 정보를 신속하게 전송하기 위하여 상호 'Verification Channel'이라 불리는 독립적인 데이터 전송 통로를 유지한다. 그리고 검증된 DNS 정보들은 Verification Cache(vCache)라 불리는 별도의 저장 공간에 반 영구적으로 저장되며, 여기에 저장된 정보들은 다른 DoX 시스템과의 캐쉬 정보 비교를 통하여 캐쉬의 오염 및 잘못된 정보의 유지 여부를 결정하는데 사용된다.

그러나 Lihua Yuan 등이 제안한 DoX 시스템은 상호 협업하는 Recursive DNS 간의 DNS 정보 교환을 위하여 Validation Channel을 구성하는 등의 추가적인 작업을 필요로 하며, 이것은 반드시 Recursive DNS 간의 상호 신뢰 관계를 유지하여야 한다. 그리고 협업하는 시스템 간의 빈번한 데이터 전송으로 인하여 부가적인 네트워크 트래픽이 발생되며, 이는 협업하는 시스템의 수에 따라 전체 DNS 서비스의 성능이 크게 저하되는 문제를 발생시키는 제약점을 가지고 있다.

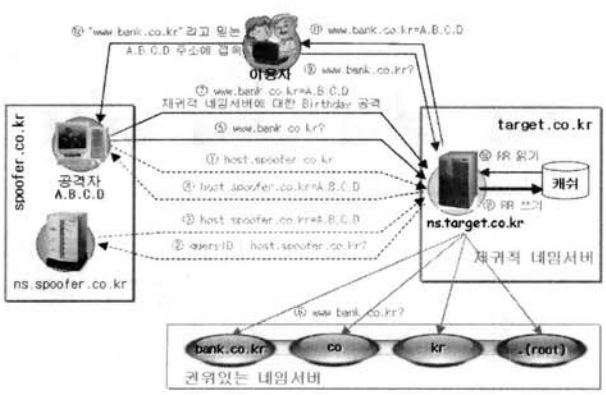
그리고 추가적으로 DoX에 대한 연구는 현재 서비스에 대한 기본적인 설계와 기법이 제안되어 있는 실정이고 이에 대한 명확한 상세명세로서 프로토콜 등에 대한 구현과 검증은 되어있지 않은 상태이다.

3. 캐쉬 신뢰성 향상을 위한 제안기법

3.1 캐쉬 오염 공격 분석

Recursive DNS의 구조적인 취약점을 이용한 캐쉬 정보를 위·변조하기 위한 대표적인 유형이 Birthday 공격으로 (그림 4)와 같은 형태로 공격이 이루어진다.

공격자는 먼저 사전 질의 과정을 거쳐 Recursive DNS가 사용하는 IP 주소와 포트 정보를 알아낸다. Recursive DNS에 대한 기초 정보를 알아낸 공격자는 DNS에게 오염시키려는 도메인 정보를 질의한다. 공격자의 질의를 받은 DNS는 자신의 캐쉬 정보를 검색하고, 캐쉬 내에 해당 정보가 존재하지 않는다면 Authoritative DNS에게 정보를 요청한다. 그러나 이 과정에서 공격자는 Recursive DNS가 Authoritative DNS로부터 올바른 질의 결과를 받기 전에 Authoritative DNS가 응답을 보낸 것과 동일한 형태의 패킷 정보를 생성하여 Recursive DNS에게 전송한다. 이때 Recursive DNS는 자신의 질의에 대한 응답이 자신이 전송한 IP 주소, 포트 번호, 그리고 TID(Transaction ID)를 기준으로 올바르다고 판단하기 때문에 Authoritative DNS가 보낸 패킷이 아닌 응답이라 하더라도 식별기준이 되는 정보가 동일하다면 올바른 응답으로 인식할 수 있다. 그러나 TID의 경우 Recursive DNS가 정보를 요청할 때마다 다른 값을 생성하기 때문에 공격자가 사전에 TID 정보를 감지하기가 불가능하다. 따라서 공격자는 사전 작업을 통하여 알아낸 Recursive DNS의 IP 주소와 포트 번호와 함께 임의의 TID 값을 생성하여 전송한다. 또한 TID 정보를 사전에 알 수 없기 때문에 동일한 패킷 정보에 TID 정보만을 변경하여 수백 건의 응답 메시지를 Recursive DNS에게 전송한다.



(그림 4) Birthday 공격에 의한 캐쉬 오염 공격 패턴

이러한 공격 방법을 분석하면 Recursive DNS 관점에서 다음의 두 가지 중요한 공격 패턴을 확인할 수 있다.

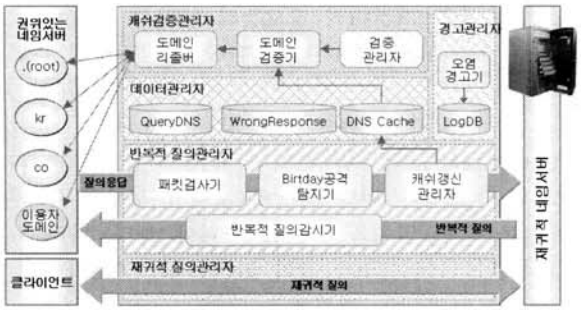
- Recursive DNS는 동일한 도메인 정보에 대하여 자신이 요청하지 않는 TID를 갖는 DNS 응답 메시지가 갑자기 급증하여 수신한다.
- 캐쉬 오염 공격이 성공한 경우 Recursive DNS는 공격자와 자신이 정보를 요청한 Authoritative DNS로부터 올바른 DNS 응답 메시지를 두 번 전송받는다.

따라서 이러한 두 가지 공격 패턴을 이용하여 캐쉬 오염 공격을 사전에 확인 할 수 있다. 또한 저장된 캐쉬 정보에 대하여 사후 검증 과정을 통하여 사전 탐지 과정에서 검출하지 못한 오염된 캐쉬 정보와 Recursive DNS의 캐쉬 정보 갱신 주기로 인한 잘못된 캐쉬 정보를 탐지하도록 하여 Recursive DNS의 신뢰성을 향상시킬 수 있다.

3.2 제안 기법의 구성

(그림 5)는 이 논문에서는 DNS 서비스 체계 상의 다른 시스템과는 독립적으로 제안하는 기법(CPDM : Cache Poisoning Detection Method)의 전체적인 구조를 보여준다. (그림 5)와 같이 제안하는 기법은 크게 재귀적 질의 관리자, 반복적 질의 관리자, 데이터 관리자, 캐쉬검증 관리자, 경고관리자로 구성된다.

Recursive DNS에 대한 캐쉬 오염 공격은 재귀적 질의가 아닌 반복적 질의 수행 과정에서 발생한다. 따라서 제안된 기법은 Recursive DNS가 요청하는 재귀적 질의에 대해서는 단지 향후 통계 자료 산출을 위한 모니터링 기능만을 수행한다. 반복적 질의관리자는 Recursive DNS에 대한 캐쉬 공



(그림 5) 제안된 시스템의 전체 구조

격을 사전 탐지하며, 크게 반복적 질의감시기, 패킷 검사기, Birthday 공격 탐지기, 캐쉬 갱신 관리기로 구성된다. 반복적 질의감시기는 향후 캐쉬 공격 감지를 위한 기초적인 작업을 수행하며, 캐쉬 검증 관리자는 Recursive DNS에 저장된 캐쉬 정보에 대한 사후 검증 작업을 수행한다. 데이터 관리자는 제안하는 기법 내에서 사용되는 데이터를 관리하기 위한 모듈이며, 경고 관리자는 캐쉬 오염 공격이 탐지되었을 때 이를 시스템 운영자에게 알려주기 위한 작업을 수행한다.

데이터 관리자에서 관리하는 데이터들 중에서 QueryDNS에 저장되는 데이터의 스키마 구조는 QueryDNS = <IP, Port, TID, QueryName>으로 구성된다. 여기서 IP와 Port는 Recursive DNS 서버의 IP 주소와 응답 메시지를 수신할 포트 번호를 의미한다. 그리고 TID는 Recursive DNS가 수신하는 응답 메시지가 올바른 응답인지를 판단하기 위하여 질의 요청 시에 생성하는 TransactionID를 의미하며, QueryName은 질의를 요청한 도메인 이름이다.

DNSCache는 Recursive DNS에게 전달한 올바른 DNS 응답 정보를 의미한다. DNSCache가 관리하는 데이터 스키마 구조는 DNSCache = <TimeStamp, Priority, IP, Port, TID, QueryName, ResponseIP>로 구성된다. 여기서 TimeStamp는 DNS 응답 정보가 수신된 시간이며, Priority는 향후 캐쉬에 저장된 DNS 정보에 대한 사후 검증 시에 사용하는 우선 순위를 의미하고 Birthday 공격 탐지기의 캐쉬오염 공격 검출 과정에서 결정된다. IP, Port, TID, QueryName은 QueryDNS 정보와 동일하며, ResponseIP는 DNS 응답으로 전송된 도메인 이름에 대응되는 IP 주소를 의미한다.

WrongResponse는 반복적 질의에 대한 응답 정보가 QueryDNS에 존재하지 않는 정보, 즉 잘못된 IP 주소, Port 번호 오류, 요청하지 않은 Transaction ID 등으로 인하여 Recursive DNS가 요청하지 않는 DNS 정보가 수신되었을 경우 이를 저장하여 Birthday 공격을 감지하는데 사용된다. WrongResponse에 저장되는 정보는 WrongResponse = <IP, Port, QueryName, Response#, TimeStamp>로 구성되며, IP, Port, QueryName은 QueryDNS와 동일하다. Response#는 수신된 DNS 응답 정보에 잘못된 TID 정보를 포함하고 있는 메시지의 개수를 의미한다. 그리고 TimeStamp는 잘못된 DNS 응답 정보를 마지막으로 저장한 시간을 의미하며, 이 정보는 WrongResponse에 저장된 DNS 응답 정보를 초기화하기 위한 목적으로 사용한다.

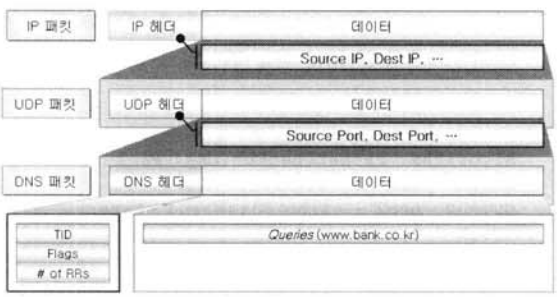
일반적으로 Birthday 공격의 경우 충분히 많은 수의 임의의 TID를 생성(보통 750개 이상)하여 Recursive DNS에게 전송하기 때문에 잘못된 TID를 포함하는 응답 메시지의 수를 이용하여 Birthday 공격 여부를 탐지한다.

다음은 제안된 시스템의 주요 모듈들에 대한 기능을 상세히 설명한다.

3.3 반복적 질의 감시기

반복적 질의 감시기의 역할은 캐쉬 오염 공격 탐지를 위한 사전 작업으로 Recursive DNS가 Authoritative DNS에게 요청한 반복적 질의에 대한 기본적인 정보를 저장한다. Recursive DNS는 (그림 6)과 같이 Authoritative DNS에게 도메인 정보를 요청할 때 질의하는 도메인 이름과 더불어 자신의 IP 주소와 데이터를 전송받을 포트 번호, 그리고 올바른 응답을 판별하기 위한 Transaction ID 정보를 함께 전송한다. 따라서 향후 질의에 대한 공격 탐지를 위하여 Recursive DNS가 요청한 질의에 대한 기본 정보를 저장하여야 한다.

(그림 7)은 전체 구조 상에서 반복적 질의감시기의 처리 알고리즘을 보여준다.



(그림 6) DNS 질의 패킷의 구조

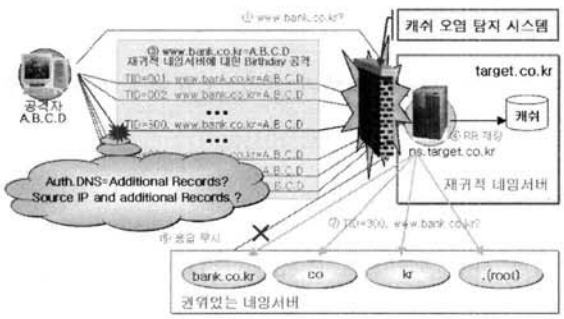
```
Algorithm. RecursiveQueryMonitor(dns: Packet)
INPUT   dns  : DNS Packet 정보
BEGIN
   // Step1. packet으로부터 각각의 field 정보 추출
   ip_no = ExtractIP(dns)
   port_no = ExtractPort(dns)
   transID = ExtractTransactionID(dns)
   queryName = ExtractQuery(dns)

   // Step2. QueryDNS DB에 저장
   SaveToQueryDNSDB(ip_no, port_no, transID, queryName)
END
```

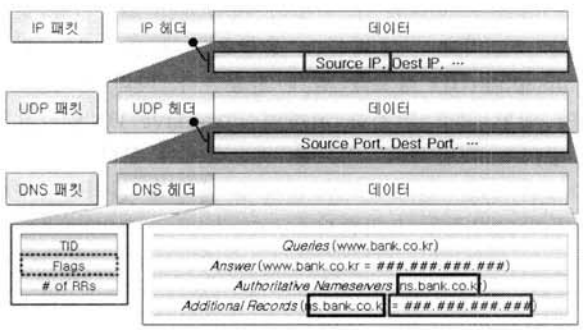
(그림 7) 반복적 질의감시기 처리 알고리즘

3.4 패킷 검사기

Authoritative DNS로부터 전송된 도메인 정보를 포함하는 패킷은 질의 대상 도메인 및 응답한 DNS의 정보를 포함한다. 따라서 (그림 8)과 같이 Recursive DNS가 수신하는 도메인 응답 메시지의 신뢰성을 판단하는 가장 기초적이고, 단순한 방법이 Authoritative DNS가 응답한 패킷내의 정보들의 일치성 검사이다.



(그림 8) DNS 응답 패킷의 도메인 정보



(그림 9) DNS 응답 패킷의 구조

(그림 9)는 Recursive DNS의 도메인 질의에 대하여 Authoritative DNS가 응답하는 패킷의 구조를 보여준다. 그림에서와 같이 전송받은 도메인 정보가 Authoritative DNS로부터의 올바른 응답 전송이라면 패킷 내에 포함된 네임 서버의 도메인 정보와 IP 주소 정보들은 모두 일치하는 값을 가져야 한다. 따라서 패킷검사기는 (그림 10)과 같이 Recursive DNS가 수신하는 모든 응답 패킷에 대하여 IP Header와 Authoritative Nameservers 필드에 기술된 응답 시스템의 IP 주소, Authoritative Nameservers와 Additional Records 필드에 기술된 네임 서버의 도메인 정보를 비교하여 응답 메시지의 신뢰성을 검사한다. 이러한 메커니즘은 현재 전세계적으로 가장 많이 사용되고 있는 BIND 시스템에서도 DNS 시스템의 보안 문제로 버전 8의 중간 릴리스 버전부터 적용되고 있다.

3.5 Birthday 공격 탐지기

Birthday 공격 탐지기는 Recursive DNS에 대한 공격자의 캐쉬 오염 공격을 탐지하는데 먼저 Recursive DNS로 수신된 DNS 응답이 올바른지를 판별하기 위하여 수신된 DNS 응답 메시지 정보가 QueryDNS 데이터베이스에 존재하는지를 검사한다. QueryDNS 데이터베이스는 Recursive DNS가 Authoritative DNS에게 요청한 모든 반복적 질의 정보를 저장하기 때문에 올바른 DNS 응답일 경우 해당 정보는 반드시 QueryDNS에 존재하여야 한다. 따라서 Recursive DNS가 수신한 DNS 응답 메시지가 QueryDNS에 존재하는 경우, 즉 응답 메시지와 동일한 IP 주소, 포트 번호, Transaction

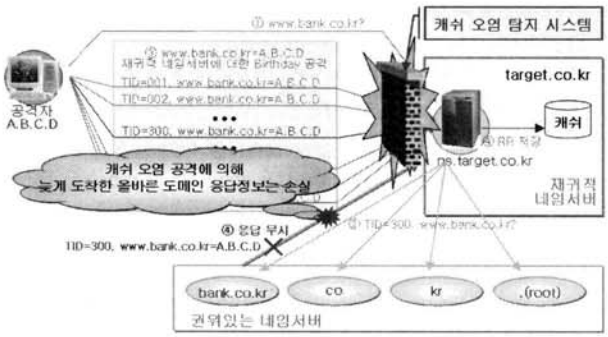
```
Algorithm. PacketChecker(dns: Packet)
INPUT   dns  : DNS Packet 정보
OUTPUT  bool : cache poisoning state
BEGIN
   // Step1. 패킷의 Flag field를 검사
   IF (Flag is not correct) THEN
      RETURN false
   ENDIF

   // Step2. packet으로부터 각각의 field 정보 추출
   ip_header  = ExtractIP(dns, IPHeader)
   ip_authRR  = ExtractIP(dns, AuthoritativeNameServer)
   dom_authRR = ExtractDomain(dns, AuthoritativeNameServer)
   dom_addRR  = ExtractDomain(dns, AdditionalRecords)

   // Step3. Answers field와 Authoritative DNS 정보 비교
   IF ((ip_header .NOTEQUAL. ip_authRR) OR
       (dom_ansRR .NOTEQUAL. dom_addRR)) THEN
      CALL PoisonAlerter
      RETURN false                  // AlertCachePoisoningAttack
   ENDIF

   // Step4. Packet 내의 모든 도메인 정보는 일치함
   RETURN true
END
```

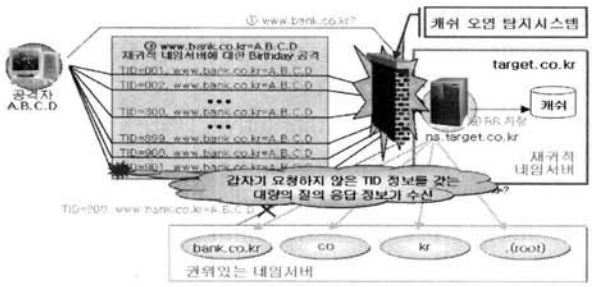
(그림 10) 도메인 일치성 검사를 위한 알고리즘



(그림 11) 캐쉬 오염 공격으로 인한 DNS 응답 유실

ID를 갖는 정보가 QueryDNS에 존재하는 경우 DNS 응답 메시지는 올바른 응답 메시지로 간주한다. 올바른 응답 메시지가 도착하면 해당 정보를 QueryDNS로부터 삭제하고, 향후 사후 검증을 위하여 수신된 DNS 응답 정보를 DNSCache 데이터베이스에 저장한다. 그리고 DNS 응답 정보는 캐쉬 갱신 관리자를 통하여 Recursive DNS에 전달한다.

Recursive DNS가 수신한 DNS 응답 메시지가 QueryDNS에 존재하지 않는다면 해당 도메인에 대한



(그림 12) 잘못된 DNS 응답의 급증

응답 메시지는 캐쉬 오염 공격이 성공한 상태이거나 Birthday 공격 중인 상태를 의미한다. 만일 캐쉬 오염 공격이 성공적으로 이루어진다면 (그림 11)과 같이 올바른 DNS 응답 정보는 두 번 도착하게 되며, 나중에 도착한 DNS 응답 정보는 무시된다.

제안하는 기법에서는 이러한 상태를 감지하기 위하여 Recursive DNS에 전달되는 DNS 정보를 DNSCache에 저장한다. 캐쉬 오염 공격이 성공한 경우, QueryDNS로부터는 DNS 응답 정보가 삭제되고 DNSCache에 DNS 응답 정보가 저장된다. 따라서 Authoritative DNS로부터 전송된 올바른 DNS 응답의

```
Algorithm. CheckFloodingAttack(dns: Packet)
INPUT   dns    : DNS Packet 정보
OUTPUT  bool   : cache poisoning state
BEGIN
   // Step1. 전송 받은 packet으로부터 query 정보 추출
   ip   = ExtractData(dns, QueryIP)
   port = ExtractData(dns, QueryPort)
   tid  = ExtractData(dns, QueryTID)
   name = ExtractData(dns, QueryDomain)

   // Step2. 질의한 내용에 대한 응답인지를 검사
   IF (FindQuery(QueryDNS, ip, port, tid, name)) THEN
      remove query data from QueryDNS
      save response data into DNSCache
      RETURN false
   ENDIF

    // Step3. 이미 수신된 질의 결과인지를 검사
    IF (FindCache(DNSCache, ip, port, tid, name)) THEN
      CALL PoisonAlerter
       RETURN true            // AlertCachePoisoningAttack
   ENDIF

    // Step4. 해당 도메인에 대해 기존의 잘못된 정보가 수신되었는지를 검사
   IF (.NOT. FindWrongResponse(ip, port, name)) THEN
      Save response into WrongResponse
   ENDIF

   Increase response # in WrongResponse

   // Step5. 잘못된 정보를 포함한 도메인 정보의 수에 따fms 공격 탐지
   IF (response # > threshold) THEN
      CALL PoisonAlerter
      RETURN true            // AlertCachePoisoningAttack
   ELSE
      IF (FindCache(DNSCache, ip, port, -, name)) THEN
         Increase priority of DNS Cache information
      ENDIF
      RETURN false
   ENDIF
END
```

(그림 13) 캐쉬 오염 공격 탐지를 위한 알고리즘

경우 해당 정보가 QueryDNS에서 검색되지 않았다 하더라도 IP 주소 정보만이 상이한 동일한 DNS 응답 정보가 DNSCache에서 검색된다.

Recursive DNS는 Authoritative DNS에 DNS 질의를 요청할 때, 자신의 질의에 대한 올바른 응답인지를 검사하기 위하여 TID 값을 함께 전송한다. 이때 TID 값은 Recursive DNS가 Authoritative DNS에 질의를 요청하는 시점에서 생성된 임의의 ID이기 때문에 공격자가 사전에 이 값을 알아내기는 어렵다. 그러나 Birthday 공격에 의한 캐쉬 오염 공격에서는 충분히 많은 양(약 750개 이상)의 임의의 TID 값을 생성하여 전송할 경우 공격 성공률은 매우 높아진다.

제안하는 방법은 Birthday 공격의 이러한 특성을 이용하여 캐쉬 오염 공격을 탐지한다. Recursive DNS 관점에서는 (그림 12)와 같이 자신이 생성하지 않은 TID를 가지는 대량의 DNS 정보가 갑자기 수신될 경우, Birthday 공격이 진행 중임을 감지할 수 있다.

이 논문에서 제안하는 방법은 Birthday 공격을 감지하기 위하여 잘못된 DNS 응답 정보를 WrongResponse 데이터베이스에 저장한다. 만일 특정 도메인 정보에 대하여 Recursive DNS가 생성하지 않은 잘못된 응답의 수(Response#)가 임계치 이상 증가할 경우 Birthday 공격이 진행되고 있는 것으로 간주하여 시스템 관리자에게 경고 메시지를 전송한다. 잘못된 응답의 수가 임계치를 넘지 않는 경우라 하더라도 캐쉬 오염 공격 가능성이 있기 때문에 사후 검증 과정에서 우선적으로 검증을 수행할 수 있도록 해당 도메인 정보와 잘못된 응답수를 DNSCache에 저장하여 도메인 정보에 대한 검증과정에서의 우선순위 값으로 활용한다. 검증의 우선순위는 잘못된 응답수가 많을수록 공격의 가능성이 높아지게 되기 때문에 우선순위를 높게 설정한다. (그림 13)은 Birthday 공격을 탐지하기 위한 전체적인 알고리즘을 보여준다.

3.6 캐쉬 검증 관리자

Recursive DNS의 캐쉬에 저장된 정보는 외부로부터의 공격이 아니라도 캐쉬 정보 갱신 주기에 의하여 부정확한 정보를 보유할 수 있다. 또한 앞에서 언급한 것과 같이 Recursive DNS에 대한 외부로부터의 직접적인 공격뿐만 아니라 네임 서버의 응답 조작 및 전송되는 존 정보의 조작을 통한 공격의 경우 사전 탐지 과정에서 오염된 DNS 정보를 검출하지 못하는 경우가 발생한다.

제안하는 기법은 이러한 문제를 해결하기 위하여 Recursive DNS 내의 캐쉬된 정보에 대한 사후 검증 과정을 거쳐 Recursive DNS의 신뢰성을 향상시키도록 한다. 사후 검증 과정은 DNS 정보가 Birthday 공격 탐지기에서 얻어진 잘못된 응답 수 등의 정보를 이용하여 결정된 검증 우선 순위가 높은 DNS 정보를 먼저 검증

```
Algorithm. ValidateCache(dnsInfo)
INPUT   dnsInfo : DNSCache에 저장된 DNS 정보
OUTPUT  bool    : cache poisoning state
BEGIN
   // Step1. 우선 순위가 높은 데이터를 먼저 검사
   Sort cache data by priority and TimeStamp

   // Step2. DNSCache로부터 데이터를 추출
   vData = DNS_Lookup from DNSCache

   // Step3. Authoritative DNS로부터 DNS 정보를 얻어옴
   qData = DNS_Lookup from AuthoritativeDNS

   // Step4. DNS 정보의 오염성 여부 검사
   IF (vData = qData) THEN
     Remove DNS information from DNSCache
     RETURN true
   ELSE
     CALL PoisonAlerter
     RETURN false
   ENDIF
END
```

(그림 14) 캐쉬 정보 사후 검증을 위한 알고리즘

하고, 같은 우선 순위를 갖는 DNS 정보는 캐쉬에 저장된 시간을 의미하는 DNSCache의 TimeStamp 필드 값에 의한 순서로 검증된다.

(그림 5)에서 도메인 검증기가 실질적인 DNS 정보의 검증 작업을 수행한다. 도메인 검증기는 DNSCache에 저장된 DNS 정보들의 검증 우선순위 값과 TimeStamp를 고려하여 검증 순서를 결정하고, 도메인 리졸버를 호출하여 Authoritative DNS로부터 DNS 정보를 수신하여 이를 캐쉬에 저장된 도메인 정보와의 비교·검증 작업을 수행한다. 검증 관리자는 도메인 검증기에게 주기적으로 검증 요청 이벤트를 발생시켜 DNSCache에 저장된 DNS 정보를 검증하도록 한다. 마지막으로 도메인 리졸버는 Authoritative DNS에게 DNS 정보를 요청하고, 이에 대한 응답을 수신하여 도메인 검증기에게 전달하는 기능을 수행한다.

(그림 14)는 이러한 캐쉬검증관리자의 동작에 관한 알고리즘을 기술한 것이다.

4. 제안 기법의 평가

제안된 기법의 성능을 검증하기 위하여 기존의 전통적인 DNS와 DNSSEC을 적용한 DNS와의 비교 분석을 수행하였다. 먼저 DNS서비스를 위해 요구되는 시간을 분석하면 다음과 같다.

일반적인 DNS에서 사용자에게 서비스를 제공하는데 소요되는 시간 T_{DNS}는 다음과 같이 계산된다.

$$T_{DNS} = T_{RDT} + T_{RDP} + n \times (T_{ADT} + T_{ADP}) \times C_{Auth} \quad \text{(식 1)}$$

여기서 T_{RDT}는 이용자와 Recursive DNS 간의 데이터 전송 시간을 의미하며, T_{RDP}는 DNS 정보를 얻기 위하여 Recursive DNS 내에서 처리되는 시간을 의미한다. 그리고 T_{ADT}와 T_{ADP}도 각각 Recursive DNS와 Authoritative DNS와의 데이터 전송 시간 및 Authoritative DNS에서의 데이터 처리 시간을 의미한다. n은 Authoritative DNS의 계층적 단계를 표시하고 C_{Auth}는 각각의 Authoritative DNS에서의 지연 시간을 의미한다.

DNSSEC이 적용된 DNS 서비스 체계에서는 DNS간에 전송되는 리소스 레코드 정보에 암호화 기법을 적용하기 때문에 DNS 서비스를 제공하기 위하여 제공되는 시간 T_{DNSSEC}은 다음과 같다.

$$T_{DNSSEC} = T_{DNS} + T_{RSEC} + n \times T_{ASEC} \times C_{ASEC} \quad \text{(식 2)}$$

위의 T_{RSEC}과 T_{ASEC}은 각각 Recursive DNS와 Authoritative DNS에서 리소스 레코드에 대한 암호화 기법 적용을 위하여 소요되는 시간을 의미하며, C_{ASEC}는 여러 Authoritative DNS 간의 시스템 성능에 의한 지연 시간을 의미한다. DNSSEC 기반의 방법들은 위의 식과 같이 Recursive DNS와 Authoritative DNS에서 전송되는 데이터의 암호화를 위하여 추가적인 시간 비용($T_{RSEC} + n \times T_{ASEC} \times C_{ASEC}$)을 요구한다. 그러나 이러한 암호화를 위한 비용은 루트 DNS와 같이 DNS 서비스 체계상에서의 상위 레벨에 위치한 Authoritative DNS 일수록 DNS 정보를 요청하는 Recursive DNS의 수가 급속하게 증가하게 되며, Authoritative DNS가 처리하여야 할 T_{ASEC} 비용도 크게 증가하는 문제가 발생한다.

이러한 문제를 해결하기 위하여 Lihua Yuan 등이 제안한 DoX 시스템에서 DNS 서비스를 처리하는데 소요되는 시간 T_{DoX}는 아래와 같이 계산되는데,

$$T_{DoX} = T_{DNS} + m \times (T_{vDT} + T_{vDP}) \times C_{DoX} \quad \text{(식 3)}$$

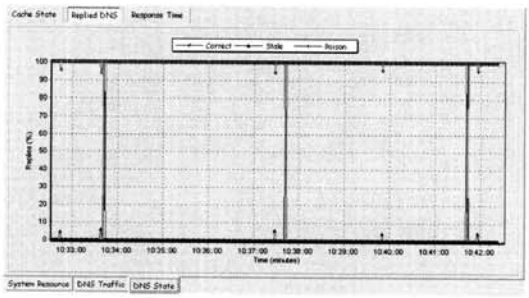
여기서, m은 협업하는 Recursive DNS 시스템의 수를 의미하며, T_{vDT}와 T_{vDP}는 협업하는 Recursive DNS에 요청한 DNS 정보를 처리하는 소요되는 데이터 전송 시간과 처리 시간을 의미한다. C_{DoX}는 협업하는 Recursive DNS간의 성능 차이로 인하여 발생하는 지연 시간을 의미한다. 위의 식과 같이 DoX 시스템의 경우 협업하는 Recursive DNS들의 수에 따라 추가적인 시간 비용(T_{vDT} + T_{vDP})은 크게 증가하며, 네트워크 상의 부하도 커지는 문제를 갖는다. 일반적인 DNS 시스템과 비교하여 DoX 시스템은 $m \times (T_{vDT} + T_{vDP}) \times C_{DoX}$ 만큼의 추가적인 시간 비용이 소요된다.

제안하는 시스템에서의 데이터 처리 비용은 아래와 같고 여기서 C_{CPDS}는 제안하는 시스템으로 인한 Recursive DNS에서의 데이터 처리 지연 시간을 의미한다.

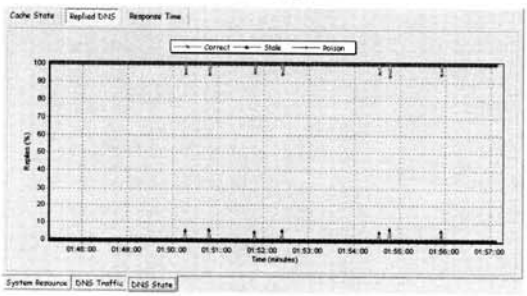
$$T_{CPDS} = T_{DNS} + C_{CPDS} \quad \text{(식 4)}$$

제안하는 기법은 Authoritative DNS와의 통신 시에 주고받는 메시지를 모니터링하여 Recursive DNS의 오염 여부를 판단하고, 개별 Recursive DNS에 대해서만 독립적으로 동작하기 때문에 전체 네트워크 상에서의 추가적인 부하가 발생하지 않는다. 따라서 제안하는 기법의 DNS 처리 비용은 (식 4)에서와 같이 DNS 데이터 모니터링을 위한 약간의 지연 시간 C_{CPDS}만이 추가로 요구되어 DNS 성능에 대한 영향이 미약한 것을 확인할 수 있다.

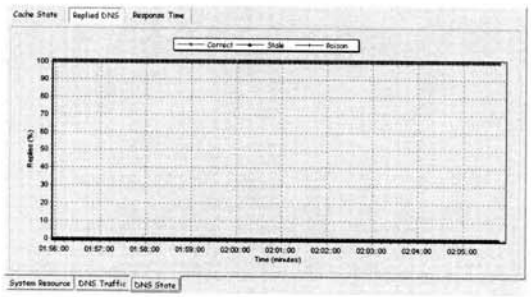
이 논문에서 제안하는 기법은 기존의 DNS 체계에 어떠한 수정을 가하지 않고, 다른 DNS와는 독립적으로 동작하면서 캐쉬 오염 공격에 효과적으로 대응할 수 있는 캐쉬 오염 탐지 프레임워크로 캐쉬 오염 공격자의 공격 유형을 분석하여 공격을 사전 탐지할 수 있는 기법과 사전 탐지 과정에서 검출하지 못한 오염된 정보에 대한 사후 검증 과정을 거쳐 캐쉬에 저장된 DNS 정보의 신뢰성을 강화하였다. (그림 15)는 일반적인 DNS, DNSSEC을 적용한 DNS 및 이 논문에서 제안하는 기



(a) 일반적인 DNS



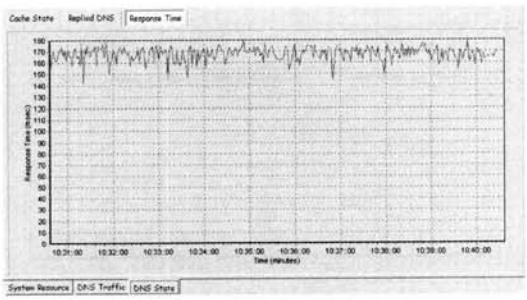
(b) DNSSEC



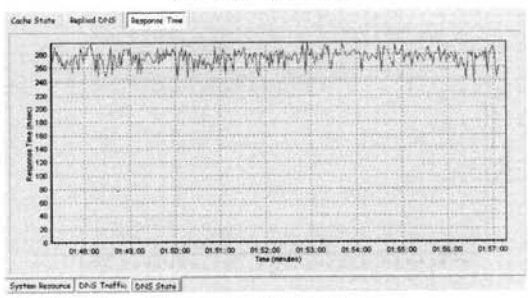
(c) 제안된 방법

(그림 15) DNS 캐쉬 정보의 신뢰성 비교

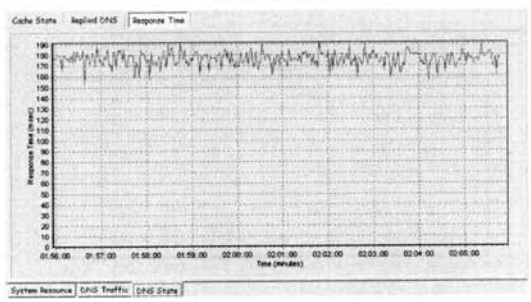
법을 적용한 DNS 시스템을 각각 구현하여 DNS 서버의 캐쉬에 대한 신뢰성을 비교한 것이다. 시스템간 성능 비교를 위하여 Recursive DNS에 요청하는 정보는 국내 .kr 도메인으로 등록된 702,781개의 도메인 중에서 중복을 허용하며 무작위로 5만개의 정보를 선택하여 질의 및 성능평가를 수행하였다. 실험은 10개의 프로세스를 생성하였으며 각각 프로세스는 매번 0~0.5초 사이의 시간 간격으로 독립적으로 질의를 요청하였다. 또한 초기의 캐쉬정보 생성을 위하여 프로그램 실행 후 처음 30분은 캐쉬 오염공격을 수행하지 않았으며, 이후 0~5분 사이의 임의의 시간에 캐쉬 오염공격이 이루어지도록 하였다. 일반적인 DNS 체계와 비교하여 DNSSEC이 적용된 DNS의 경우 캐쉬 오염은 발생하지 않았지만, 정상적인 상태에서도 발생할 수 있는 잘못된 캐쉬 정보는 지속적으로 유지되는 것을 볼 수 있다. 그러나 제안하는 방법은 외부 공격자로부터의 캐쉬 오염 공격뿐만 아니라 정상적인 상태에서 발생되는 잘못된 정보를 최소화하여 DNS 서버의 캐쉬 정보에 대한 신뢰성이 향상된 것을 확인할 수 있다.



(a) 일반적인 DNS



(b) DNSSEC



(c) 제안된 방법

(그림 16) 도메인네임 질의에 대한 응답시간 비교

〈표 1〉 기존 연구 대비 제안 기법의 평가

구분	기존 연구		제안 기법
	DNSSEC	DoX	
사용 프로토콜	확장된 DNS Protocol	DNS Protocol + P2P Protocol	DNS Protocol
DNS간 의존도	고	중	저
전체 Recursive DNS 성능에 대한 영향	고	중	저
성능 영향도	고	중	저
보안성	고	중	중
적용 단위	DNS 전체	협약된 DNS	개별 DNS
실무 적용 가능성	저	중	고

시스템의 신뢰성 측면에서 외부 캐쉬 오염 공격에 대해서는 기존의 DNSSEC과 거의 유사한 신뢰성을 보였으며, 정상적인 상태에서 발생하는 DNS 정보의 갱신 전파 지연으로 발생하는 DNS 캐쉬의 신뢰성은 향상된 것을 확인할 수 있었다.

(그림 16)은 도메인네임 질의에 대한 비교한 것이다. 일반적인 DNS 서버의 질의 응답시간은 평균 171msec(0.17)초로 가장 빨랐으며, DNSSEC이 적용된 서버는 평균 281msec(0.28)의 응답시간이 소요되었다. DNSSEC의 경우 DNS 서버간의 전송되는 정보에 대한 추가적인 서명인증 과정이 필요하며, 서명인증 알고리즘이 복잡하여 응답시간이 지연된 것으로 분석되었다. 제안된 방법의 경우 평균 179msec(0.179초) 정도의 응답시간이 소요되었으며, 일반적인 DNS와 비교하여 약 4% 정도의 추가적인 응답시간을 요구하였다.

<표 1>은 제안된 기법의 성능을 기존 연구들과 비교하여 정리한 것이다. DNSSEC과 DoX 시스템은 기본적으로 다른 시스템들과의 협업에 의해 구동 되기 때문에 DNS간 의존도가 높으며, DNS 보안성 향상을 위한 알고리즘의 데이터 처리 성능이 전체 시스템에 주는 영향이 커진다. 또한 DNS 보안 메카니즘의 적용 범위가 넓어짐에 따라 실무 적용이 어려워지는 문제가 발생한다.

보안성 측면에서는 <표 1>과 같이 DNSSEC와 비교하여 동일한 성능을 얻을 수는 없지만 캐쉬 정보의 사후 검증 과정을 거쳐 이러한 단점은 보완될 수 있다. 또한 DNS간 의존도를 낮추고 기존의 DNS 체계를 수정하지 않고도 제안 기법을 실세계 환경에 적용할 수 있는 장점이 있다.

5. 결 론

전 세계적으로 인터넷 이용이 급격히 증가하면서 해

킹 등과 같은 인터넷 서비스에 대한 공격이 증가하고 있으며, 이에 따라 국가 인터넷 기반 시설에 대한 안정성 제고가 매우 중요한 문제로 대두되었다. 특히 2003년 1월 25일 발생하였던 인터넷 대란과 같이 인터넷 접속 관문이라 할 수 있는 도메인 접속에 장애가 발생할 경우 사회·경제적으로 엄청난 피해를 양산할 수 있다.

DNS 정보 위·변조 공격을 막기 위하여 지금까지 제안되어온 연구들은 대부분 암호화 기법을 기반으로 하고 있으나, 이 방법은 루트 DNS로부터 클라이언트까지의 모든 시스템에 대하여 DNSSEC이 적용된 레코드를 처리할 수 있도록 하여야 한다. 그리고 암호화 처리를 위한 추가적인 연산 작업은 전체 시스템의 성능을 저하시키는 문제를 발생시킨다.

이 논문에서는 현재의 DNS 서비스 환경에 즉시 적용이 가능한 DNS 정보의 오염 여부를 실시간으로 탐지할 수 있는 방법을 제안하였다. 또한 외부 공격에 의한 오염된 DNS 정보뿐만 아니라 기간 만료로 인하여 유효하지 않은 DNS 정보까지도 검출하여 Recursive DNS의 신뢰성을 향상시키는 방법을 제안하였다.

이 논문에서 제안하는 방법은 기존의 DNS 서비스 체계를 그대로 유지하면서 각각의 Recursive DNS 서버에 독립적으로 구성할 수 있도록 하여 현재 운영되고 있는 DNS 서비스 체계에 즉시 적용이 가능하다. 따라서 제안된 방법을 활용하면 현재의 DNS 서비스 체계에서 각 Recursive DNS가 관리하고 있는 DNS 정보에 대한 신뢰도를 향상시키는데 크게 기여할 수 있을 것이다.

이 논문에서 제안하였던 방법을 구현한 시스템은 IPv4 주소 체계를 기준으로 구현되었다. 그러나 IPv4 주소 체계상에서의 주소 자원 고갈 문제로 인하여 네트워크 주소 체계는 IPv6로 전이되고 있다. 또한 RFID/USN 환경이 발전함에 따라 이를 지원하기 위한 DNS 체계가 필요하다. 따라서 구현된 시스템을 향후 IPv6 주소 체계 및 RFID/USN 환경에 적용하기 위한 연구가 필요하다.

참 고 문 헌

[1] ITU, "World Telecommunication Indicators Database 10th Edition," http://www.itu.int/ITU-D/ict/publications/world/world.html, January 2007.

[2] OECD, "OECD Communications Outlook 2005," http://www.thepublicvoice.org/events/tunis05/oecd_outlook.pdf, OECD Publishing, 2005.

[3] 한국인터넷진흥원, 「2006 한국인터넷백서」. 한국인터넷진흥원, 2006년 5월.

[4] 정보통신부, "정보통신기반보호법," (일부개정: 2005. 3. 31).

[5] L. Yuan, K. Kant, P. Mohapatra, and C. N. Chuah, "DoX: A Peer-to-Peer Antidote for DNS Cache Poisoning Attacks," Proceedings of IEEE International Conference on Communications(ICC2006), Istanbul, Turkey, June 2006.

[6] P. Albitz and C. Liu, DNS & BIND, 4th Edition, O'REILLY Press, 2001.

[7] K. D. Frazer, "NSFNET: A Partnership for High-Speed Networking," Final Technical Report 1987-1995, Merit Network Inc., 1995.

[8] P. Vixie, "DNS and BIND Security Issues," Proceedings of the 5th USENIX UNIX Security Symposium, Salt Lake City, Utah, USA, pp.209-216, June 1995.

[9] D. Eastlake, "Domain Name System Security Extensions," RFC 2535. http://www.ietf.org/rfc/rfc2535.txt, 1999.

[10] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS Security Introduction and Requirements," RFC 4033. http://www.ietf.org/rfc/rfc4033.txt, 2005.

[11] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "Resource Records for DNS Security Extensions," RFC 4034, http://www.ietf.org/rfc/rfc4034.txt, 2005.

[12] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "Protocol Modifications for the DNS Security Extensions," RFC 4035, http://www.ietf.org/rfc/rfc4035.txt, 2005.

[13] C. Liu, "Transactional Security in BIND 9: Securing DNS," Linux Magazine, November 2001.

주 용 완

e-mail : ywju@nida.or.kr

1997년 한국외국어대학교 경영학(학사)

2002년 한국외국어대학교 경영학(석사)

2007년 숭실대학교 공학박사

1997년~1999년 한국정보사회진흥원(구 한국전산원) 주임

2001년~2003년 아시아태평양인터넷협회 이사

2002년~2004년 아시아태평양인터넷정보센터 이사

2003년~2005년 IPv6 포럼 주소워킹그룹 의장

2005년~2005년 정보통신부 국제표준화 전문가

2000년~현 재 한국인터넷진흥원 정책기획단

관심분야 : 인터넷 기술정책, DNS 시스템, 인터넷 보안, 네트워크 성능관리, IPv6 등

이 응 재

e-mail : ejlee@nida.or.kr

1994년 충북대학교 컴퓨터과학과(학사)

1996년 충북대학교 전자계산학과(석사)

2005년 충북대학교 대학원 전자계산학과 (박사)

2006년~현 재 한국인터넷진흥원 정책기획단

관심분야 : 이동객체 데이터베이스, 위치기반서비스, 지리정보 시스템, 센서네트워크, 네트워크 성능관리, 인터넷 기술정책, DNS 시스템, 인터넷 보안 등

남 광 우

e-mail : kwnam@kunsan.ac.kr

1995년 충북대학교 전자계산학과(학사)

1997년 충북대학교 전자계산학과(석사)

2001년 충북대학교 전자계산학과(박사)

2001년~2004년 ETRI 텔레매틱스·USN연구단 선임연구원

2004년~현 재 군산대학교 컴퓨터정보공학과 조교수

관심분야 : LBS, geoSensor Network, 이동체 데이터베이스, GIS