IPv6 환경에서 해쉬 함수를 이용한 안전한 주소 자동 생성 기법 구현*

주승연, 경계현, 고광선, 엄영익
성균관대학교 정보통신공학부
e-mail : {comdoc,gyehyeon,rilla91,yieom}@ece.skku.ac.kr

Implementation of a Secure Address Auto-Generation Scheme using a Hash Function in the IPv6 Environments*

Seungyoun Ju, Gyehyeon Gyeong, Kwang Sun Ko, and Young Ik Eom
School of Info. and Comm. Eng., Sungkyunkwan University

요 약

IPv6 환경에서는 NDP(Neighbor Discovery Protocol)를 이용한 주소 자동 설정 메커니즘을 지원한다. 그러나, NDP 는 메시지 내 중요 정보가 네트워크 상에 그대로 노출됨으로 인해 각종 공격에 취약하다. 이러한 취약성을 극복하기 위해, CGA(Cryptographically Generated Address)를 사용하여 주소의 소유권 증명이 가능한 SEND(SEcure Neighbor Discovery)가 도입되었다. 그러나 SEND 는 높은 비용 연산으로 인해 모바일 기기 등에 적용하는데 한계점을 가진다. SEND 의 한계점을 보완하고자 해쉬 함수를 이용해 주소 자동 설정에 사용되는 임시 주소를 감추는 기법이 제안되었다. 이 기법은 DAD(Duplicate Address Detection) 과정 중 SEND 수준의 보안을 제공하면서도 빠르게 동작할 수 있는 장점을 갖는다. 본 논문에서는 리눅스 환경에서 제안 기법을 구현해 보고, 주소 생성 시간 측정 및 DAD 과정에서 드러난 서비스 거부 공격에 대한 안전성을 검증한다.

1. 서론

IPv6 는 IPv4 의 주소 공간을 128 비트로 확장한 것으로써 헤더의 단순화를 통해 처리의 효율성을 제공하며 다양한 추가 옵션 지원, 보안 강화, QoS(Quality of Service) 처리 지원의 이점을 갖는다[1]. 또한, DHCP 서버나 관리자의 참여 없이 장치 스스로 주소를 생성하고 할당할 수 있는 비상태형 주소 자동 설정 기능을 제공한다. 비상태형 주소 자동 설정에서는 생성된 주소를 할당하기 앞서, 네트워크 상의 주소 유일성 검증을 위해 DAD 과정을 거쳐야 한다[2]. 그러나, 이 과정에 사용되는 메시지에 대한 보호가 없다면, 악의적 노드가 호스트의 중요 정보를 얻어 서비스 거부 공격을 시도할 위험이 존재한다[3].

이러한 공격에 대응하기 위해 CGA 를 사용하는 SEND 프로토콜이 도입되었으나, 주소 소유권 증명 방식을 사용하는 특성상 주소 생성 및 검증에 높은 비용의 연산이 요구된다[4]. 따라서 성능이 낮고 빈번한 네트워크 이동이 있을 수 있는 모바일 기기 적용에는 한계점을 가진다. 이를 보완하고자 DAD 과정에 적은 연산으로 수행될 수 있는 해쉬 함수를 사용하여, 모바일 기기에도 적용이 가능하며 SEND 수준의 보안을 제공할 수 있는 해쉬 함수 기반 주소 생성 및 검증 기법이 제안되었다[5]. 본 논문에서는 리눅스 환경에서 제안 기법을 구현한 후, 주소 생성 속도를 측정하고 DAD 과정에서의 안전성을 검증한다

*본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT 연구센터 육성 · 지원사업의 연구결과로 수행 되었음
(IITA-2007-C1090-0701-0027)

2. 배경 지식 및 관련 연구

본 절에서는 DAD 과정 중 발생할 수 있는 서비스 거부 공격과 NDP 전반에 걸쳐 보안을 제공하는 SEND 메커니즘에 대해 살펴본다. 또한 제안 기법에서 사용하는 난수를 이용한 주소 생성에 대해 알아보고 제안 기법을 설명한다.

2.1. DAD 과정에서의 서비스 거부 공격

DAD 과정에서의 가능한 서비스 거부 공격은 두 가지로 생각할 수 있다[6]. 첫 번째, NS(Neighbor Solicitation) 메시지를 받은 악의적인 노드가 NS 메시지의 target address 필드를 복제하여 NA(Neighbor Advertisement) 메시지를 생성, 전송한다. NS 메시지를 보낸 노드가 이 복제 NA 메시지를 받으면 DAD 과정을 실패하게 된다.

두 번째, NS 메시지를 받은 공격자가 NS 메시지의 target address 필드를 복제하여 NS 메시지를 생성, 전송한다. 이렇게 함으로써 링크 상에 같은 임시 주소로 DAD 과정을 수행하고 있는 노드가 있는 것으로 가장한다. DAD 과정 중에 있는 노드가 자신의 임시

주소와 동일한 주소로 DAD 과정을 수행하는 노드가 있음을 알게 되면 DAD 과정은 실패한다[6].

이러한 공격은 송신자의 주소에 대한 소유권을 검증할 수 없다는 점, NS 메시지의 target address 필드에 임시 주소가 평문으로 포함되어 공격자에 의한 위조가 가능하다는 점에 기인한다. SEND 는 주소에 대한 소유권 검증을 통해 공격을 방지하는 반면, 제안 기법은 해쉬 함수를 사용하여 임시 주소를 감추는 방식으로 공격을 무효화 시킨다.

2. 2. SEND(Secure Neighbor Discovery)

SEND 프로토콜은, 공개키를 이용하여 주소를 생성하는 CGA 를 사용함으로써, 메시지를 수신한 노드가 해당 메시지를 전송한 노드의 주소에 대한 소유권을 검증 할 수 있다[7]. 그렇기 때문에 DAD 과정 중 서비스 거부 공격을 방지 할 수 있다. 하지만 CGA 의 생성에 P3-696MHz 시스템에서 평균 1.7964 초라는 높은 비용의 연산 처리가 요구되므로 휴대형 단말기에 적용하는데 한계점을 가진다. 또한 소유권 검증을 위해서 RSA 알고리즘이 사용되므로 이 역시 많은 부하가 걸리게 된다[8].

2.3. RGA(Randomly Generated Address)

RGA 는 IPv6 주소의 하위 64 비트 interface identifier 를 난수로 대체하여 전체 주소를 구성한다[9]. 이 경우, 난수는 interface identifier 에서와 같이 u 비트는 'local'로 설정이 되어야 하고 g 비트는 'individual'로 설정이 되어야 한다. 그러므로 실제 난수로 생성되는 부분은 62 비트가 된다[10]. 이렇게 생성 된 RGA 의 충돌 가능성은 생일 문제(Birthday problem)를 통해 확인할 수 있다. n 개의 주소에서 k 개를 생성할 때, 한 개의 주소라도 중복이 생길 확률의 상한은 다음 식에 의해서 구해진다.

$$P(n,k) = 1 - (\frac{n-k+1}{n})^{k-1}$$

n = 2^{62}, k = 500 이라 할 경우, 충돌 쌍이 하나라도 존재할 확률은 5.4·e-14 이 된다[9]. 이 확률을 고려할 때 충돌 가능성은 극히 희박함을 알 수 있다. 본 논문에서 구현하는 기법은 임시 주소로 이 RGA 를 사용한다.

2. 4. 해쉬 함수 기반 주소 생성 및 검증 기법

본 기법은 표 1 과 같은 NS 메시지 수정을 필요로 한다[5].

<표 1> 제안 기법의 NS 메시지 수정사항

수정사항	표 준	제안 기법
IPv6 링크-로컬 주소 생성	MAC 주소 이용	난수 사용
Target address 필드	생성된 IPv6 주소	생성된 IPv6 주소의 해쉬 값
IPv6 목적지 주소	Solicited-Node 멀티캐스트 주소	All-Node 멀티캐스트 주소

수정 된 NS 메시지는 임시 링크-로컬 주소를 MAC 주소 대신 난수를 사용하여 구성한다. 또한 공격자에게 NS 메시지가 노출되더라도 정보를 알 수 없도록 하기 위해 NS 메시지의 target address 필드에 임시 주소에 대한 128 비트 해쉬 값을 넣는다. 목적지 주소는 all-node 멀티캐스트 주소(ff02::1)를 사용한다. 이로써 solicited-node 멀티캐스트 주소와 달리 하위 24 비트의 예측이 불가능하게 된다. 이러한 수정으로 임시 주소를 감추고 공격자의 복제를 무의미하게 한다.

임시 주소의 노출 없이 해쉬 값 충돌 확인을 수행하기 위해선 Challenge-Response 검증 과정을 사용한다[5].

3. 구현

본 절에서는 리눅스 환경에서 주소 생성 기법을 구현하기 위해 추가 해야 할 ICMPv6 메시지들의 구조를 알아보며 적당한 보안을 제공할 수 있는 보안 알고리즘을 선정한다. 또한, 전체 구현을 주소 생성 및 할당 모듈과 메시지 핸들러 모듈로 나눠 각 모듈 별 구현을 상세하게 알아본다.

3.1. 구현 환경

본 논문에서는 주소를 생성하여 DAD 를 수행하는 호스트와, DAD 과정 중의 호스트에게 공격을 시도하는 공격자로 망을 구성하여 안전성을 평가한다.

3.2. 보안 알고리즘

본 논문에서 난수 알고리즘으로, 구현 환경이 될 리눅스에서 기본적으로 제공하는 LRNG(Linux pseudo-Random Number Generator)를 사용한다. LRNG 는 시스템 이벤트를 이용해 엔트로피를 축적한다. 난수 생성은 시프트 레지스터, SHA-1(Secure Hash Algorithm-1) 연산들을 사용한다[11].

다음으로 임시 주소를 감추기 위해 사용 될 해쉬 알고리즘을 선정해야 한다. 해쉬 출력이 IPv6 주소 길이와 같은 128 비트를 가지며, 그 안전성이 검증되어 유용하게 이용되고 있는 알고리즘으로 MD5(Message Digest 5)를 생각 할 수 있다. MD5 는 리눅스에서 파일의 무결성 검증을 위해 사용되고 있다[12]. 해쉬 충돌을 찾아낼 수 있는 방법이 알려졌으나 한 시간여의 긴 시간을 요구하므로 본 적용과 같이 짧은 시간(NS 전송 후 1000ms 이내 NA 가 없을 시 주소 할당)의 은폐를 요구하는 환경에서는 적당한 안전성을 제공한다[13]. 무엇보다 빠르게 해쉬 값을 계산, 바로 적용이 가능하므로 본 구현에 적합하다.

3.3. ICMPv6 메시지 타입 추가

해쉬 함수 기반 주소 생성 기법을 적용하게 되면 기존 NS / NA 메시지와 혼재 된 상황에서 NS / NA target address 필드의 값이 해쉬 값인지 아닌지 여부의 판단이 불가능하다. 따라서 표 2 와 같은 총 4 개의 새로운 ICMPv6 메시지 타입의 지정이 필요하다.

추가 메시지의 구성은 NS 메시지 구조를 기본으로 한다. Challenge-Response 프로토콜을 위한 CH / RE 메

시지는 난수를 넣을 공간 128 비트를 추가하여야 하며 target address 필드에는 충돌한 해쉬 값을 넣는다. 호스트는 자신의 해쉬 값과 CH / RE 메시지 내 target address 필드를 비교하여 처리해야 하는 메시지인지 여부를 판단하게 된다.

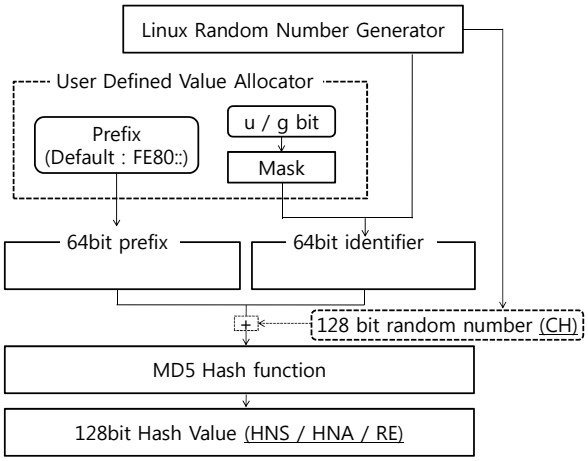
<표 2> 추가 된 메시지

Message Type	Target Address 이하 필드
HNS* (148)	MD5 에 의해 해쉬 된 임시 주소
HNA* (149)	HNS 에 의해 수신한 해쉬 값
CH** (150)	검증을 위한 난수 (r)
	MD5 에 의해 해쉬 된 임시 주소
RE** (151)	검증 정보 [MD5(임시 주소 ‖ r)]
	CH 에 의해 수신한 해쉬 값

* target address 필드에 해쉬 값을 갖는 NS / NA 메시지
** CHallenge - REsponse 과정을 수행하기 위한 메시지

3.4. 주소 생성 및 할당 모듈

본 모듈은 주소를 생성하고 이에 대한 해쉬 값을 계산하며, DAD 과정을 정상적으로 마치게 되면 주소를 인터페이스에 할당한다. 또한 CH 메시지를 위한 난수 생성, RE 메시지를 위한 해쉬 값 계산 기능을 갖는다. 그림 1 은 LRNG 로 부터 주소 생성, 해쉬 값을 출력하는 주소 생성 부분을 보여준다.



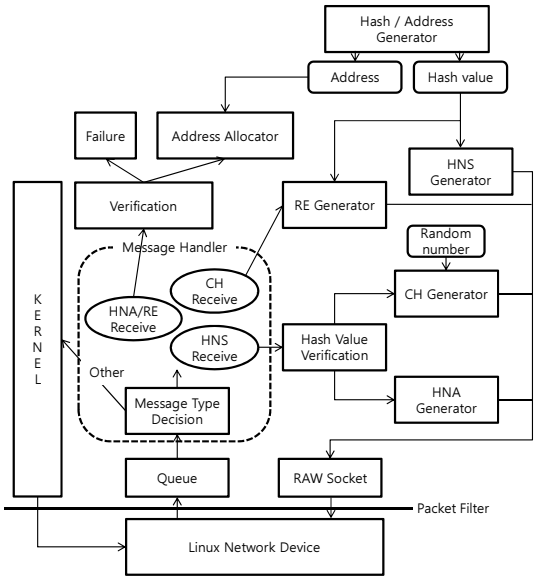
(그림 1) 주소 생성

64 비트 prefix 할당, u / g 비트 설정, MD5 해쉬 함수 수행의 과정을 거치며, 생성 해쉬 값은 HNS / HNA 메시지 내 target address 필드에 사용된다. CH 메시지를 받을 경우는 CH 메시지 내 난수와 임시 주소의 OR 연산 후 MD5 해쉬 값을 계산함으로 RE 메시지에 사용 될 검증 정보를 생성한다.

3.5. 메시지 핸들러 모듈 구현

메시지 핸들러 모듈의 전체 구성은 그림 2 와 같다. 메시지 핸들러 모듈은 응용 레벨에서 실행되지만, 패킷은 netfilter 를 사용하여 커널 처리 전의 장치 레벨에서 가져온다. ip6tables 를 이용하여 ICMPv6 패킷은 queue 에 쌓아두도록 하고, ipq library 로 queue 에서 패킷을 가져와 타입에 맞는 처리를 하도록 구현하였다. 반면, 생성된 메시지는 ICMPv6 레벨의 RAW 소켓을 사용하여 send_msg() 함수를 이용해 전송한다.

본 구현에서 관심을 갖는 메시지 타입은 추가로 정의한 네 가지 타입뿐이므로 나머지 ICMPv6 패킷은 커널에 재주입하여 커널에서 처리토록 했다.



(그림 2) Message Handler Module

받아들인 메시지의 처리는 현재 네트워크 장치의 상태에 따라서 달라진다. 구현에서 정의한 장치 상태는 Auto-configuration state, Response waiting state, Listen state 의 3 가지이다. 각 상태에서의 4 가지 메시지에 대한 처리는 표 3, 표 4, 표 5 에 나타냈다.

먼저, Auto-configuration state 는 주소를 생성하고 DAD 과정 수행 중에 있는 상태이다.

<표 3> Auto-configuration state 메시지 처리

Received Message Type	처 리
HNS	해쉬 값이 같을 경우 CH 전송
HNA	송신 주소와 임시 주소를 비교
CH	RE 메시지 생성 / 전송
RE	Don't care

다음으로, Response waiting state 는 CH 메시지 전송 후 RE 메시지를 대기하고 있는 상태이다.

<표 4> Response waiting state 메시지 처리

Received Message Type	처 리
HNS	해쉬 값이 같을 경우 CH 전송
HNA	송신 주소와 임시 주소를 비교
CH	RE 메시지 생성 / 전송
RE	새로 계산 된 해쉬 값과 비교

마지막으로, Listen state 는 주소 설정을 맞추고 설정 주소로 패킷을 받아들일 수 있는 상태이다.

<표 5> Listen state 메시지 처리

Received Message Type	처 리
HNS	해쉬 값 같을 경우 HNA 전송
HNA / CH / RE	Don't care

4. 실험 결과 및 고찰

4.1. 성능 평가

10000 개 주소 생성 시간의 평균은 표 6 과 같다.

<표 6> 구현 기법과 CGA 의 보안 강도에 따른 주소 생성 시간 비교

구 분	P3-550 MHz	
구현 기법	0.00012 초	
CGA	0.00015 초 (sec* = 0)	1.8771 초 (sec* = 1)

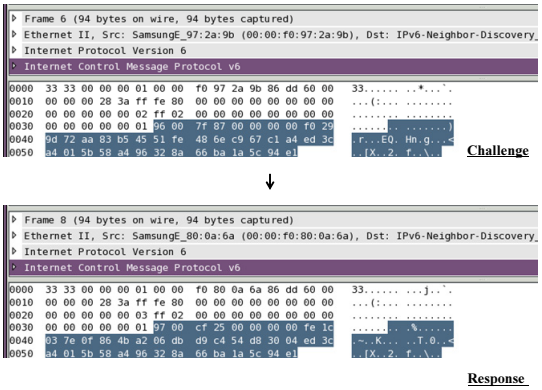
* 보안 강도

구현 기법은 CGA 를 최하 보안 강도인 0 으로 설정할 경우에는 CGA 와 성능이 비슷하다. 하지만 CGA 의 보안 강도를 1 로 할 경우에는 구현 기법이 CGA 보다 만 배 가량 빠르게 동작함을 알 수 있다. 이는 CGA 가 두 번의 160 비트 출력을 갖는 SHA-1 알고리즘을 수행하지만, 구현 기법은 한 번의 빠른 MD5 해쉬 함수를 수행하기 때문이다.

4.2. 안전성 평가

공격자를 2 절에 언급 된 서비스 거부 공격을 수행하도록 구현하여 동일 링크 상에 둔다. NA 복제의 경우, 주소 자동 설정을 시도하는 노드 A 는 Auto-configuration state 또는 Response waiting state 에서 동일 해쉬 값을 갖는 NA 메시지를 받게 된다. 그러나 NA 메시지의 송신지 주소가 노드 A 의 생성 주소와 다르므로 DAD 과정에서의 공격은 실패하게 된다.

NS 복제의 경우는 해쉬 값 충돌인지, 동일한 임시 주소로 DAD 과정을 수행 중인지를 확인하기 위해 Challenge-Response 검증 과정을 필요로 한다. 그림 3 은 CH / RE 메시지를 패킷 모니터링 툴을 이용하여 확인한 결과이다.



(그림 3) Challenge-Response 메시지

공격자로부터 동일한 해쉬 값의 NS 메시지를 받은 노드 A 는, 검증에 사용할 난수를 CH 메시지에 포함하여 공격자에게 보낸다. 공격자는 노드 A 로 부터 받은 난수와 자신의 주소를 이용하여 해쉬 값을 생성하고, 노드 A 에게 RE 메시지를 전송한다. 노드 A 는 자신이 생성한 난수와 자신의 주소를 이용하여 새로운 해쉬 값을 구하고 공격자로 부터 받은 RE 메시지 내 해쉬 값과 비교한다. RE 메시지 내 해쉬 값은 공격자의 주소로 생성 된 것이므로, 노드 A 에 대한 DAD 과정에서의 공격은 실패하게 된다[5].

5. 결론

본 논문에서는 해쉬 함수를 사용하여 빠르게 주소를 생성, 검증하는 기법을 구현하여 보았다. 구현 결과 SEND 의 적절한 보안 수준에서의 주소 생성보다 빠른 속도로 주소를 생성함을 확인하였으며, DAD 과정 중 DoS 공격에도 대응함을 알 수 있었다.

하지만, 리눅스 응용 레벨에서의 구현이므로 실제 적용을 위해서는 커널 레벨에서의 구현이 필요하며 DAD 과정 이외의 NS / NA 위장 공격에 노출되어 있으므로 이에 대한 추가적인 연구가 필요하다.

참고문헌

[1] S. Deering and R. Hinden, RFC2460, Internet Protocol Version 6 (IPv6) Specification, Dec. 1998.

[2] S. Thomson and T. Narten, RFC2462, IPv6 Stateless Address Autoconfiguration, Dec. 1998.

[3] J. Arkko, T. Aura, J. Kempf, V. m. Mänylä, P. Nikander, and M. Roe, "Securing IPv6 neighbor and router discovery," Proc. of the 3rd ACM workshop on Wireless Security'02, pp. 77-86, Sep. 2002.

[4] J. Arkko, J. Kempf, B. Zill and P. Nikander, RFC3971, SEcure Neighbor Discovery (SEND), Mar. 2005

[5] 경계현, 고광선, 엄영익, "IPv6 환경에서 해쉬 함수 기반 강건한 주소 생성 및 검증 기법," 한국정보보호학회논문지 Vol. 17, No. 1, pp. 115-119, Feb. 2007.

[6] P. Nikander, J. Kempf and E. Nordmark, RFC3756, IPv6 Neighbor Discovery (ND) Trust Models and Threats, May. 2004.

[7] T. Aura, RFC3972, Cryptographically Generated Addresses (CGA), Mar. 2005.

[8] 박기태, 김중민, 복혁구, "IPv6 의 보안기능을 강화하는 Secure ND Protocol 의 구현," 정보과학회논문지, Vol. 2, No. 1, Dec. 2005.

[9] M. Bagnulo, I. Soto, A. Garcia-Martinez and A. Azcorra, Internet-Draft, Random generation of interface identifiers, Jan. 2002.

[10] R. Hinden and S. Deering, RFC2373, IP Version 6 Addressing Architecture, Jul. 1998.

[11] Z. Gutterman, B. Pinkas and T. Reinman, "Analysis of the Linux Random Number Generator," Proc. of the 2006 IEEE Symposium on Security and Privacy, May. 2006.

[12] R. Rivest, RFC1321, The MD5 Message-Digest Algorithm, Apr. 1992.

[13] X. Wang, D. Feng, X. Lai, H. Yu, "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD," Proc. of the CRYPTO 2004, Aug. 2004.