Active Node내에서의 QoS향상을 위한 패킷 보안 메커니즘

최정희*, 신미예*, 이동희**, 이상호*
충북대학교 컴퓨터 과학과 네트웍 연구실*, 극동 대학**

A Packet security mechanism for Inproving QoS in Active Node

Jeong-Hee Choi*, Mi-Yea Shinn*, Dong-Hee Lee**, Sang-Ho Lee*
Chungbuk National University*, Kuk-Dong college**

요약

현재 대부분의 액티브 네트워크에서 액티브 노드로 전송된 패킷은 프로그램 코드와 데이터를 동시에 가지고 있으며, 기존 네트웍 방식에서의 저장(store)-전송(forward) 형태와는 달리 저장(store)-연산(compute)-전송(forward) 방식으로 패킷 처리를 한다. 이 과정에서 패킷에 악성 코드가 추가되거나 실제 데이터가 변경되는 경우에는 데이터의 무결성을 보장할 수 없다. 따라서 이 논문에서는 액티브 노드에 들어오는 패킷과 그 노드에서 처리가 이루어진 후 나가는 패킷을 비교하여 악의적이거나 악의적이지는 않지만 실수로 인한 패킷의 오류를 방어함으로서 액티브 노드의 QoS를 향상시켜주는 방식을 제안한다.

1. 서론

인터넷 등과 같은 기존의 네트워크에서는 연결된 시스템들 사이에 패킷을 고속으로 전달하는 것이 주요 기능이었으므로 패킷의 헤더만 처리하면 되었다. 하지만 액티브 네트워크의 경우, 사용자가 원하는 프로그램을 패킷 내에 가지고 있거나 혹은 임의의 중간 노드에서 일부 특별한 관리자가 미리 제공하는 프로그램을 특정 중간 노드에서 실행하여, 고속의 패킷 전달 기능뿐만 아니라 매우 다양하고 유동적인 처리를 패킷에 행할 수 있다.[1][3]

액티브 네트워크를 구현하는 방법은 크게 두 가지로 분류할 수 있다. 첫째는 미리 어떤 특별한 관리자에 의해 중간 노드에 프로그램을 미리 장착해 놓고 필요에 따라 갱신하는 방법으로 프로그래밍 가능한 스위치 방법 (Programming Switch approach) 또는 분리 방법 (Discrete approach)이라 하며, 이러한 방법은 전화와 같은 전기 통신망에서 볼 수 있는 지능망의 개념과 유사하다고 볼 수 있다. 둘째는 프로그램조차 일반 패킷과 마찬가지로 중간 노드에 전달되어 실행되는 방법으로 캡슐 방법 (Capsule approach) 또는 통합 방법 (integrated approach)이라 한다.[4]

액티브 네트웍에서는 프로그램 코드와 데이터가 동시에 한 패킷에 담아져서 전송되고, 액티브 노드에서 이 패킷에 들어 있는 프로그램 코드가 처리된다. 이 때 액티브 노드에서는 패킷 속에 있는 프로그램 혹은 노드에 이미 장착되어 있는 프로그램을 실행시키게 되는데 이 실행 과정에서 악성 코드는 패킷의 무결성을 침해하거나 노드에 존재하는 자원을 과도하게 요구하여 액티브 노드의 성능을 저하시키게 된다. 따라서 액티브 노드에 들어와 실행되는 패킷의 무결성은 물론 노드의 자원을 보호하는 안전하고 신뢰성 있는 보안 메커니즘이 필요하다. 이 논문에서는 패킷의 데이터 무결성을 침해하는 요소들을 살펴보고 그에 따른 대응 방안으로서 액티브 노드에 들어오는 패킷과 나가는 패킷의 해쉬값을 비교하여 악의적이거나 또는 악의적이지는 않지만 실수로 인한 오류를 감지하고 방어하기 위한 모델을 설계하여, 액티브 노드에서 프로그램 코드 처리가 이루어진 후 조작되어진 데이터의 무결성 여부를 체크함으로서 액티브 네트웍 노드의 QoS를 향상시켜주는 방식을 제안한다.

2. 액티브 네트웍 보안

악의적인 코드에 의해서 순수 데이터가 오염되어 질 수 있거나 노드의 자원을 무한히 쓰는 경우에 발생하는 노드의 취약점을 다음과 같이 분류한다.

-*Memory Protection*

메모리 보호의 위협은 서비스 코드가 정확한 행동을 하는지에 대한 보장이 없기 때문에 액티브 네트웍 안에서 서비스 코드의 실행에 직접적 영향을 미친다.

- Node OS, EE(Execution Environment) 혹은 다른 서비스 코드는 액티브 서비스 코드의 비정상적인 행동으로 인해 오염되어 질 수 있다.
- EE에 분포된 서비스 코드는 의도된 침입자 혹은, 의도되지 않은 비트 에러에 의해 오염되어 질 수 있다.
- 액티브 노드안에 저장된 것은 또 다른 액티브 서비스에 의해서 의도적이든 혹은 의도적이지 않든 간에 조작되어 질 수 있다.

-*Resource Management*

자원 관리에 있어서의 위협은 액티브 서비스의 정확성에 더 많은 위협이 있다.

- 단일 혹은 다수의 패킷들은 다중 노드에서 다량의 자원들을 소모 할 수 있다. 예를 들어, 하나의 액티브 패킷은 끝나지 않는 시간동안 두 노드 사이에서 배회 할 수 있다.
- 어플리케이션이나 액티브 서비스로부터 발생되는 서비스는 네트웍에서 또는 네트웍의 일부분에서 다량의 패킷들을 요구하게 된다. 즉 액티브 서비스는 큰 패킷을 이용하여 서비스하게 되는 것이다. [1][2]

3. 액티브 노드 내에서의 패킷 무결성 지원 모델

액티브 네트웍에서의 패킷을 전송하는 방법에는 두 가지가 있는데 그 중 하나는 out-of-band이며 다른 하나는 in-band이다. 전자인 out-of-band 방식에서의 액티브 네트웍 노드는 다수의 미리 정의된 프로토콜을 포함하고 있는 스위치이며, 후자의 in-band방식 패킷은 노드에서의 프로토콜에 의해 처리가 요구되는 것이 아니라, 액티브 패킷이 코드 형식으로 프로토콜을 운반한다. 즉 액티브 네트웍 노드는 운반되는 코드를 실행함에 의해서 패킷을 처리한다.[5]

다음 그림 1에서의 EE(Execution Environment)는 가상 머신(virtual machine), 가상 포트(virtual ports), 그리고 programming interface를 포함한 virtual network node를 정의하며, EE안에서 실행되는 액티브 패킷의 기능은 EE's 가상 머신(virtual machine)에 의해 정의된 명령어에 의해 실행되어지고 service programming interface(SPI)를 통해 EE를 엑세스한다. 노드 OS는 실행 환경과 전송 대역이나 프로세서의 time cycle, 또는 메모리와 같은 물리적 하드웨어 자원 사이의 작동 계층이다. 노드 OS의 존속은 동시 다중 지원 EE를 위한 필요성으로부터 기인한다.[1]

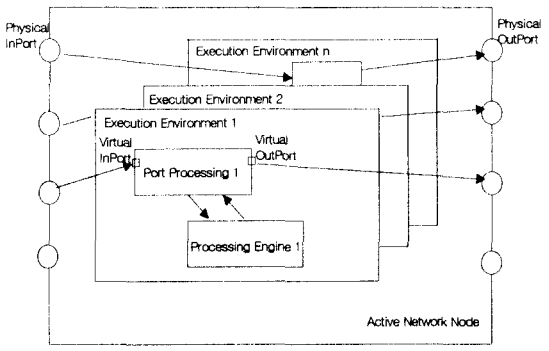

그림 1. Active Network Node Architecture

그림 1에서 액티브 노드 OS는 다중 실행 환경을 지원하고 모든 액티브 노드에 공통된 기본적인 수준의 기능성을 제공하며 액티브 패킷은 액티브 노드에서 프로그램 또는 기능성들에 대한 포인터와 함께 데이터를 추가하여 운반하게 되며, 액티브 노드에서는 이러한 프로그램을 실행한다. 즉, 현재의 라우터와 스위치들은 권한이 부여된 사용자만이 소프트웨어를 인스톨 할 수 있게 되어 있으나, 액티브 네트웍에서는 어떤 사용자든지 프로그램 작성과 인스톨이 가능하다. 이러한 것은 액티브 노드의 상태를 변형시킬 수 있고 또 다른 오염된 패킷을 만들어 낼 수 있다.[6]

아래 그림 2에서는 액티브 네트웍의 노드에 패킷이 들어오면 헤더 부분의 정보를 *packet classification and queuing chip*(PCQ)에서 *filter memory*(FM)에 보관하고 난 후 패킷의 전체를 *queue controller*(QCTL)로 보내진다.

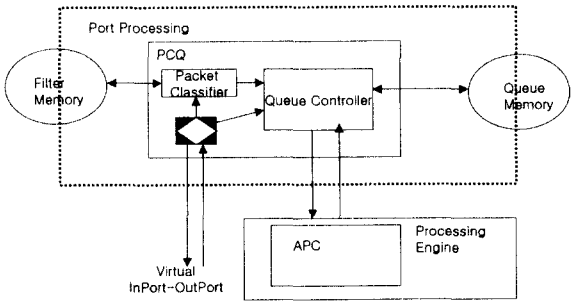

그림 2.액티브 노드의 *Port Processing*과 *Processing Engine*

QCTL에서는 패킷을 *Queue Memory*에서 대기하고 있는 대기 순서에 따라 *Processing Engine*에 보내지며 그 패킷이 가지고 있는 서비스 코드에 의한 실행이 *processing engine*의 *active processor chip*(APC)안에서 이루어진다. Processing Engine에서 실행이 이루어진 패킷은 다시 *packet classification and queuing chip*(PCQ)로 보내어진다.[7]

Processing이 이루어 진 후 패킷은 악성 코드에 의해 변질되어 진 패킷에 대한 아무런 체크도 하지 않은 상태에서 Out-Port로 보내지게 된다. 이 논문에서는 패킷 Processing이 이루어 진 후 패킷이 Out-Port로 나가기 전에 무결성 여부를 체크하는 모델을 아래 그림 3과 같이 제안한다.

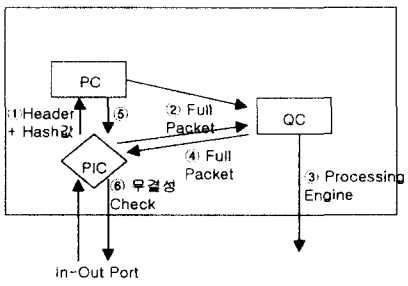

그림3. PIC(Packet Innocent Check)에서의 패킷 무결성 체크

위 그림 3에서 보여주는 PIC에서의 패킷 무결성 체크는 다음과 같이 이루어진다.

1. 액티브 패킷이 노드에 InPort를 통해 들어오면 PIC에서 패킷의 Header 와 데이터 부분을 압축한 해쉬값을 FM에 저장하고, 패킷의 전체를 QC로 보낸다.
2. QC는 대기 순서에 따라 패킷을 *Processing Engine*에 보내고, *Processing Engine*에서는 QC에서 보내주는 순서에 따라 Processing이 된다.
3. *Processing Engine*에서 Processing이 끝난 패킷을 다시 QC에 보내고 QC는 순서에 따라 PCI로 패킷 전체를 보낸다.
4. PIC에서는 QC에서 받은 패킷의 데이터 부분을 해쉬 하고 , 그 해쉬값을 FM에 저장해 놓은 값과 비교한다.
5. 그 비교값이 일치한다면 그 패킷을 OutPort로 보내고, 일치하지 않는다면 FM에 저장되어 있던 Source address에게 패킷의 재전송을 요구하거나 패킷을 폐기 처분한다.

따라서 하나의 패킷이 APC에서 실행이 이루어 질 때 악성 서비스 코드로 인해 이 데이터가 변형이 되었다고 한다면 이 경우 변형된 데이터가 QCTL을 통과하여 *Packet Innocent Check*으로 갔을 때 이미 보관하고 있는 헤더와 해쉬값을 FM(*filter Memory*)에서 찾아 *Packet Innocent Check*에 도착한 패킷의 헤더와 그 패킷 안에 있는 데이터 부분을 다시 해쉬하여 얻은 값을 서로 비교한다. 이때 그 비교되는 해쉬값이 다르다면 실행 후에 그 데이터가 변형 혹은 조작되었을 가능성이 크므로 *Packet Innocent Check*에서는 변형된 패킷을 폐기시키거나 혹은 폐기시킨 후 FM에 저장되어 있었던 헤더의 source address에게 재전송을 요구한다. 그러나 비교한 해쉬값이 같다면 이 패킷은 무결성이 이루어 진 것이라 여겨져 정상적으로 OutPort에 보내어 진다. 이러한 방식으로 이 논문에서는 무결성을 지원하는 *Packet Innocent Check* 방식을 제안했다.

4. 결론 및 향후 연구과제

이 논문에서 제시하는 *Packet Innocent Check* 에서는 액티브 패킷이 액티브 노드에 들어와서 어떠한 실행을 한 후에 고의적인 악성 코드에 의한 데이터 변형이거나 비트 에러와 같은 비 고의적이지만은 다른 서비스에 해를 입힐 수 있는 변형된 패킷을 감지함으로서 전체적인 액티브 네트웍에 대한 QoS를 향상시킨다.

후속 연구로는 다중 노드에서 패킷의 무결성 체크를 할 때 *Packet Innocent Check*에서의 데이터 해쉬값을 비교함에 있어서의 속도 문제와 동시에 다량의 패킷이 액티브 노드에 유입이 되었을 경우의 제어 관리에 대한 설계 및 성능 평가가 이루어져야 한다.

5. 참고문헌

[1] Marcus Brunner, "Active Networks and its Management" IEEE 2000 pp.414-424

[2] D. Scott Alexander, William A. Arbaugh, Angelos D. Keromytis, Jonathan M. Smith, "Security in Active Networks", http://heavenly.nj.nec.com/alexander99sec-urity.html

[3] A Survey of Active Network Research, IEEE Communications, Jan. 1997

[4] D. Tennenhouse and D. Wetherall, Toward an Active Network Architecture, com. commun. Rev. vol 26, no 2 Apr. 1996

[5] Zhaoyu Liu, Roy H. Campbell, M.Dennis Mickunas, "Securing the Node of an Active Network" DARPA

[6] David M. Murphy "Building an Active Node on the Internet", MIT Master's thesis, May. 1997

[7] Tilman Wolf and Jonathan S.Turner, "Design Issues for High-Performance Active Router" page404-409. IEEE Journal on selected areas in COMMUNICATIONS, VOL.19, NO.3, March 2001.