Step by Step

스텝 바이 스텝 3

네트워크 보안 기술의 원리와 응용

네트워크 보안 관리, SNMP

오래전부터 브리지에서 개선된 라우터가 이기종(heterogenous) 망간 연결(인터-네트워크)을 지원하면서 더욱 다양하게 네트워크 리소스가 분산되었다. 이는 기업을 넘어 가정에서도 일반적인 환경이 되었다. 예를 들어, 프린터를 생각해보자. USB나 병렬포트로 연결된 프린터는 분산 객체라고 말할 수 없지만, IP를 보유한 네트워크 프린터는 분산된 네트워크의 개별적인 리소스이다. 이때, 프린터 상태를 어떻게 판단할까? SNMP 프로토콜은 UDP나 TCP(SNMPv2 이후) 기반의 통신으로 이를 가능하게 한다. 대부분의 운영체제(윈도우, 리눅스, 유닉스 등)는 SNMP 프로토콜 해석기(에이전트)가 존재한다. SNMP를 응용하면 다양한 리소스(PC, 프린터, 모바일, 임베디드 장치 등)를 네트워크상에서 쉽게 관리하고 이용할 수 있다.

연 재 순 서
1회 | 2007. 11 | 메시지 인증과 공개키 암호화
2회 | 2007. 12 | 보안 애플리케이션 Kerberos와 PGP
3회 | 2008. 1 | 인터넷 보안, IPSec
4회 | 2008. 2 | 네트워크 보안 관리, SNMP

연 재 가 이 드
운영체제 | 윈도우, 리눅스, 유닉스, 매킨토시 등 모든 플랫폼
기초지식 | 네트워크 개론
응용분야 | 네트워크 보안 애플리케이션, 프로토콜 구현

서상원 smiler@ssm.samsung.co.kr | 필자는 2005년부터 삼성소프트웨어멤버십 15기로 활동 중이다. 최적의 개발을 위한 플랫폼솔루션과 개발 프레임워크에 큰 관심을 두고 있으며, 이에 기반 되는 아키텍처 설계에 고심을 하고 있다. 현재는 무선네트워크관련 표준화에 관심을 갖고 아이디어를 제안하기 위해 고민하고 있다.

지난 3회에 걸쳐 네트워크 보안 솔루션 개발에 필수적으로 알아야 할 내용을 중심으로 전반적인 원리와 응용 사례를 알아봤다. 네트워크를 관리하는 솔루션인 SNMP를 소개하며 아쉽지만 네트워크보안 응용에 관한 연재를 마무리 하고자 한다. 이미 많은 개발자가 SNMP에 많은 관심이 있는 걸로 알고 있다. 지금부터 응용사례를 시작으로 SNMP에 대해 확실히 이해해보자.

SNMP의 응용사례

일상에서 흔히 접하는 네트워크 프린터를 예로 들어보자. 당신은 네트워크 프린터를 제어하는 소프트웨어를 제작해 달라는 고객의 의뢰를 받았다. SNMP에 대한 이해가 없는 개발자라면 〈화면 1〉의 윈도우 프린터 큐(Queue)를 이용해 개발하려고 시도할 것이 분명하다. 윈도우 API를 이용하면 프린터 큐에 접근해 〈화면 1〉과 같이 문서이름, 상태, 소유자, 크기 등을 쉽게 얻을 수 있고 프린트 요청 및 최소까지도 가능하다. 실제로 프로그래밍 지식이 거의 없는 일반인일지라도 프린트가 제대로 안될 경우, 제어판을 열고 프린터 큐 패널의 상태를 확인한다. 단 윈도우가 제공하는 프린터 큐는 로컬 프린터의 경우에 한한다. 즉, USB나 프린터 포트(병렬)에 직접 연결된 경우만 올바른 프린터 상태를 얻을 수 있다. 그렇다고 프린터 큐가 오동작을 한다는 말은 아니다. 네트워크로 연결된 프린터는 피드백을 받는 루틴이 없어 성공했는지, 실패했는지 확인이 어렵다. 좀더 상세하게는 프린트할 용지, 토너 잔여량 같은 상태를 얻지 못한다는 의미이다. 피드백이 필요 없는 업데이트 요청(프린트를 요청하고 취소하는 등의)

〈화면 1〉 프린터 큐 관리자(윈도우 메시지 큐)

필자 메모

IPv6시대에는 모든 디바이스에 IP가 부여된다. 이는 SNMP와 같은 관리프로토콜의 중요성을 한층 부각시킨다. 그때를 위해서라도 이번 기회에 한번 정리해보는 것은 어떨까? 기존 SNMP는 기능적인 측면에서는 여러 가지로 부족하다는 게 대부분 전문가의 의견이다. 다만 프로토콜 자체가 단순해 구현이 쉽다보니 다른 프로토콜에 비해 빠르게 전파돼 표준화까지 이른 것뿐이다. 특히 보안적인 측면은 최근에 보완되는 상황이며, 보안 분야에 관심 있다면, 한번쯤 자신의 아이디어를 제안해볼 가치가 있을 것이다.

은 별다른 문제없이 동작한다.

이런 경우에 사용가능한 솔루션이 바로 SNMP이다. 네트워크로 연결 된 다양한 리소스간의 상호작용을 하는 UDP(디폴트) 기반의 프로토콜을 제공한다. 네트워크 프린터도 하나의 리소스 객체로 분류하며, 이들은 SNMP 프로토콜 스택을 포함한다. 결과적으로 벤더 의존적이지 않은 표준 프로토콜을 사용함으로써 호환성을 높이는데 크게 기여한다. 또 다른 예로 NMS나 WhatsUp Gold와 같은 서버 모니터링 시스템이다. 서버-클라이언트 솔루션을 보유한 대부분 업체는 이와 같은 모니터링 소프트웨어를 사용한다(대표적인 SNMP 응용 프로그램이라고 할 수 있다).

〈화면 2〉는 필자가 구축했던 서버를 WhatsUp Gold로 설정한 화면이다. 녹색박스는 이상 없는 서버임을 나타내고 분홍색 박스는 서비스의 문제를 알려준다. 문제가 있는 서버를 클릭하면 해당 서버의 프로세스별 CPU 점유율, 메모리 사용량, 서비스 포트 동작 여부 등에 관한 자세한 정보를 확인할 수 있다. 이런 종류의 소프트웨어를 흔히 접해봤겠지만 내부 동작원리가 궁금했던 독자라면 이번 기회에 궁금증을 해결하기 바란다.

〈화면 2〉 WhatsUp Gold 서버 모니터링 화면

SNMP의 역사

SNMP는 1987년도 SGMP(Simple Monitoring Protocol)에서 발전한 프로토콜이다. 이때부터 네트워크 관리의 필요성이 부각되기 시작했고 SGMP와 더불어 CMOT(CMIP over TCP/IP) 프로토콜 솔루션이 개발되고 있었다. CMOT는 TCP 기반으로만 동작하다보니 UDP 기반 SGMP에 비해 복잡한 형태를 가졌지만 기능과 성능면에서 우위를 점하고 있었다. 당시에는 CMOT가 표준이 될 것이라고 예상했다고 한다. 하지만, 예상을 뒤엎고 SGMP가 쉬운 구현과 이식(포팅)의 이점을 내세워 SNMP라는 이름으로 표준 프로토콜로 자리매김했다. CMOT도 완전히 없어진 게 아니라 SNMP를 보완하는 측면에서 계속해서 발전해왔다. 시기적으로 SNMP는 인터넷의 확산과 함께한 프로토콜이다. SNMP 버전 1부터 보안을 고려한 워킹 그룹과 그렇지 않은 그룹으로 나누어 개발을 진행했고 버전3에서는 어느 정도 보안 기능을 광범위하게 적용할 수 있게 됐다. 현재 버전3이 가장 많이 쓰인다.

기본 개념

SNMP는 네트워크 관리 시스템을 의미하는 데, 네트워크 모니터링과 제어를 위한 도구 집합체라고 말할 수 있다. SNMP에서 사용하는 네트워크 관리모델에는 관리스테이션(Management Station), 관리에이전트(Management Agent), MIB(Management Informaion Base) 프로토콜이 있다. 관리스테이션은 관리자가 네트워크를 모니터링하고 제어하는 인터페이스를 제공한다. 데이터분석이나 장애복구가 가능한 중앙관리 시스템으로, MIB라고 불리는 집합체의 정보를 담고 있는 데이터베이스를 포함한다. 네트워크에 연결 된 리소스(프린터, 팩스, 서버 등)는 개별적인 객체(object)라고 표현을 하는데 이 객체에 대한 정보 집합을 MIB라고 한다. MIB에서 원하는 리소스의 특정 데이터를 질의(Query)할 수 있는데, 이를 담당하는 것이 에이전트이다.

에이전트는 관리스테이션의 명령을 받는다고 이해하면 된다. 예를 들어, 중앙 네트워크에 A부터 순서대로 10개의 서버가 연결됐다고 하자. 서버는 모두 리소스이고 SNMP에서 객체로 인식된다. 이때 B번 서버의 CPU 점유율을 얻고 싶다면, 어떻게 하면 될까? 간단하다. 관리자는 관리스테이션을 이용해 B번 서버와 연결 된 에이전트에 명령을 내린다. 'Get B.CPU'와 같은 식으로 명령을 내리면 에이전트는 MIB에서 검색해 CPU 점유율을 얻어오고 이 정보를 관리스테이션에 전달한다. 실제 프로토콜 명령과는 조금 다르지만 'Get B.CPU'란 명령어가 Get이라는 프로토콜을 이용해 B.CPU를 조회한다. 관리스테이션이 에이전트에게 명령했듯이 프로토콜은 관리스테이션과 에이전트를 연결하는 통신절차라고 볼 수 있다. 하나의 관리스테이션에 여러 개의 에이전트가 연결된다. 여기까지 이해했다면 SNMP가 어려운 게 아니라는 것쯤은 눈치 챘을 것이다. 뒷부분에서는 실제 프로토콜 메시지를 이용해 좀더 자세히 동작하는 과정을 알아볼 것이다. SNMP 프로토콜에서 사용하는 기본적인 함수(프로토콜 명

GET	관리스테이션이 에이전트로 원하는 객체의 특정 정보를 요청한다.
GET NEXT	GET과 동일하다. 다음번 정보를 얻고자 할 때 사용한다. SNMP의 각 정보들은 계층적인 구조를 가진다.
SET	관리스테이션이 에이전트로 특정한 값을 설정하기 위해 사용한다.
TRAP	에이전트가 관리스테이션에 어떤 정보를 비동기적(Asynchronous)으로 알리기 위해 사용한다. Notify라고 하며, 콜백(Callback) 함수와 같은 역할을 한다. TRAP을 제외한 나머지 함수들은 모두 동기적으로 동작한다.

〈표 1〉 기본적인 함수 원형

Version	Community	SNMP PDU

(a) SNMP 메시지

PDU type	Request-id	0	0	Variablebindings

(b) GetRequest PDU, GetNextRequest PDU, and SetRequest PDU

PDU type	Request-id	Error status	Error index	Variablebindings

(c) GetResponsePDU

PDU type	enterprise	Agent addr	Generic-trap	Specific-trap	Time stamp	Variablebindings

(d) Trap PDU

name1	value1	name2	value2	…	namen	valuen

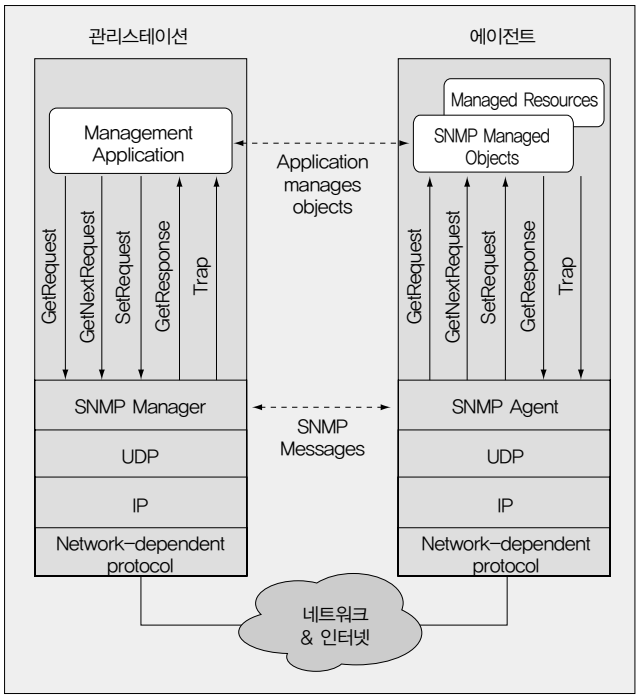
(e) Variablebindings

〈그림 1〉 함수에 따른 PDU 구성

령어)는 〈표 1〉과 같다.

실제로 〈표 1〉의 함수를 사용하려면 프로토콜 패킷에 정보를 실어 줘야 한다. 패킷구조는 〈그림 1〉과 같다. 이때 패킷을 PDU(Protocol Data Unit)라고 부른다.

〈그림 1〉의 (a)는 기본적인 메시지 형태이며, SNMP PDU 부분에 (b)~(e) 형태의 PDU를 생성한다. 즉 함수마다 인자의 개수와 형식이 다르므로 함수 별로 PDU의 구조가 다른 것이다. (b)와 (c)는 거의 동일하며, 특정 객체 ID를 이용해 원하는 정보를 얻을 수 있다. 객체 ID를 생성하는 방법은 뒷부분에서 다룰 것이다. (d)는 에이전트로부터 받은 Trap 패킷의 구조로 어떠한 에이전트로부터 정보가 도착했는지(Agent addr), Trap 유형이 무엇이고 도착한 시간은 언제인지에 대한 정보를 포함한다. (e)는 모든 PDU에 공통적으로 있는 자료구조로 인자값을 이름과 값으로 복수 개 전달할 때 이용한다.

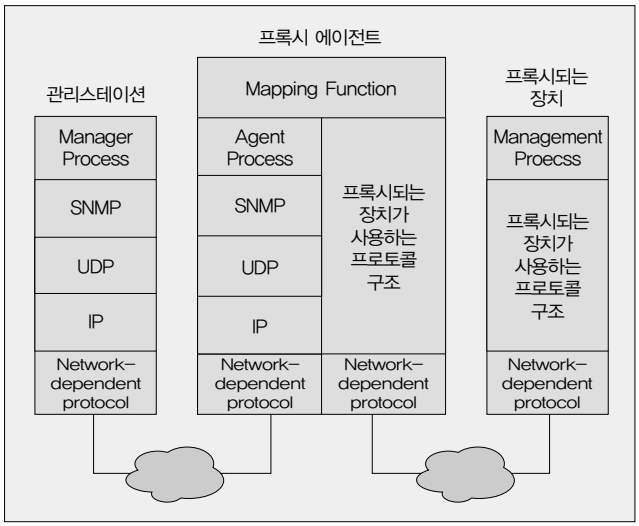


〈그림 2〉 SNMP 프로토콜 콘텍스트(Context)

프로토콜 과정과 구조

〈그림 2〉는 관리스테이션과 에이전트가 SNMP 메시지를 주고받는 프로토콜 콘텍스트를 나타낸다. 기본적으로 IP/UDP 기반으로 동작하며, MIB 정보를 에이전트로부터 얻는다. 앞서 다루었던 프린터 사례로 돌아가면, 프린터는 리소스이자 에이전트라고 할 수 있다. IP와 UDP가 설치됐고, 그 위에 SNMP 프로토콜 스택이 구현됐다. UDP를 기반으로 하면 TCP에 비해 로직이 간단하고 어느 장치든지 이식이 용이하다. 대부분 임베디드시스템에 UDP 스택이 설치 된 것도 같은 이유이다. 임베디드시스템에서 타겟보드의 전송모듈로 가장 많이 쓰이는 유틸리티가 UDP 기반의 TFTP라는 것을 알 것이다.

상황에 따라 SNMP가 구현되지 않은 장치가 있을 수 있다. 네트워크에 연결돼 있더라도 관리가 불가능한 것일까? 그렇지 않다. 이럴 때를 대비해 프록시(Proxy)라는 방법을 제공한다. 〈그림 3〉은 프록시 설정을 보여준다.



〈그림 3〉 프록시 구성

프록시 구성을 사용하는 장치로는 브리지나 모뎀이 있다. 또한

SNMP 모듈을 설치하기에는 규모가 작은 시스템도 이에 포함된다. 하지만 SNMPv2 이후로 UDP 뿐만 아니라 TCP를 비롯한 여러 프로토콜을 지원하기 시작했다. 프록시를 구성하는 경우가 드물기는 하지만 〈그림 3〉에서 프록시되는 장치만 고려한 프로토콜이 설치된 경우는 SNMP 자체에서 대부분 프로토콜을 지원하지 못한다. 〈그림 3〉을 보면 관리스테이션이 장치에 직접 PDU 메시지를 요청하는 것이 아니라 프록시 에이전트에 요청한다. 에이전트는 SNMP 프로토콜 요청받아 매핑함수(Mapping Function)를 이용해 프록시되는 장치가 인식할 수 있는 프로토콜 패킷으로 변환한다.

변환된 패킷을 프록시되는 장치에 전달하면 결국 관리스테이션이 장치와 통신하는 것과 같은 결과가 발생한다. 프록시 에이전트는 일종의 번역기라고 볼 수 있다. 이런 번역기만 있으면 자체 프로토콜을 가진 어떠한 장치라도 네트워크 관리 대상에 추가할 수 있다는 점이 프록시의 가장 큰 장점이다. SNMP에 관심 있는 독자라면 임베디드 리눅스에 SNMP를 설치하고 프록시 시스템을 구축해 볼 것을 권장한다. 자신이 임의로 프로토콜을 설계하고 SNMP와 호환할 수 있는 프록시를 구성해 WhatsUP Gold에 추가해보길 바란다.

하나의 관리스테이션은 여러 개의 호스트를 관리하며, SNMP 프로토콜은 FTP나 HTTP 등과 같은 애플리케이션 수준(Level)의 프로토콜이라고 정리할 수 있다. 〈표 1〉에서 기본적인 함수 원형을 알아봤다. SNMPv1과 SNMPv2에서는 어떠한 차이점이 있을까? SNMPv3는 뒷부분에서 다루기 때문에 생략할 것이다. 〈표 2〉에서 PDU를 비교 정리하였다. SNMPv2의 Bulk 명령은 관리자가 한꺼번에 커다란 블록의 데이터를 가져온다. 특히 GetBulk 명령은 한 번의 명령으로 테이블 전체를 전송할 목적으로 설계됐다. GetNext 명령은 〈표 1〉에서도 설명했지만, MIB안의 객체가 트리구조로 정리 된 특성을 활용한다. 한 객체가 GetNext 명령 안에서 호출되면 에이전트는 트리에서 다음번 객체를 찾고 그 객체 값을 반환한다. 즉, 찾으려는 객체가 명확하지 않을 때, GetNext를 이용해 트리를 순차적으로 검색하며 원하는 객체를 찾을 수 있다.

SNMPv1 PDU	SNMPv2 PDU	방향	설명
GetRequest	GetRequest	관리자 -> 에이전트	각 객체 목록 값 요청
GetNext Request	GetNext Request	관리자 -> 에이전트	각 객체 목록의 다음 값 요청
-	GetBulk Request	관리자 -> 에이전트	다중(Bulk) 값 요청
SetRequest	SetRequest	관리자 -> 에이전트	각 객체 목록 값 설정
-	InformRequest	관리자 -> 에이전트	요구되지 않은 정보 전송
GetResponse	Response	에이전트 -> 관리자 관리자 -> 에이전트	관리자 요청에 대한 응답
Trap	SNMPv2-Trap	에이전트 -> 관리자	요구되지 않은 정보 전송

〈표 2〉 SNMPv1과 SNMPv2의 PDU 비교

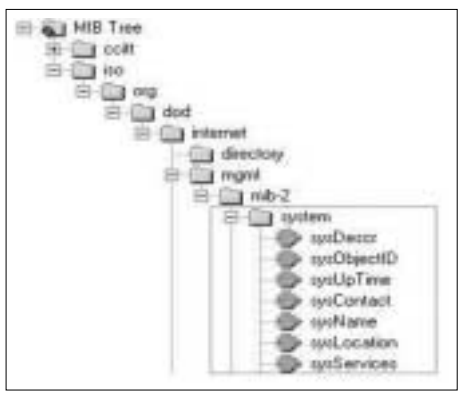
〈그림 4〉는 여러 개의 Topology를 연결한 분산 네트워크 구조를 보인다. FDDI와 토큰링 LAN, Ethernet(CSMA/CD)을 라우터를 이용해 인터-네트워크(inter-net)를 구축했다. 라우터는 여러 연결점을 갖는 브릿지(Bridge) 역할을 하고 인터-네트워크 간의 물리계층 프로토콜을 변환하는 복잡한 기능을 한다. 분산 네트워크일지라도 중앙 서버가 존재하며, 각 네트워크 마다 중간 관리자를 둔다. 라우터는 네트워크를 연결하는 에이전트 역할을 하는데, 앞서 설명한 프록시 에이전트의 개념이다. 중앙관리자는 관리스테이션으로서의 역할을 하지만 중앙관리자 입장에서는 그저 에이전트에 불과하다(〈그림 4〉는 이달의 디스켓 참조).

MIB

MIB(Management Information Base)는 네트워크에 연결된 리소스의 데이터 저장소다. MIB-I, MIB-II와 같은 분류를 갖고 있으며, 의미 있는 연결고리를 갖도록 트리구조로 저장된 데이터베이스라고 할 수 있다. 관리스테이션은 에이전트로 MIB를 검색하고 결과를 이용해 모니터링을 수행한다. MIB의 노드는 리소스의 고유 객체 ID로 표현된다. 그럼 객체 ID는 어떻게 구성됐을까? 또한 트리구조라고 했는데 도대체 어떤 구조를 갖고 있을까? 해답은 〈그림 5〉에 있다. MIB는 트리구조를 이루는데 트리의 Leaf 노드 객체가 실제로 관리되는 객체를 의미한다. 이 객체는 리소스의 정보를 모두 가졌다. 객체 ID를 표현하면 다음과 같다.

예시 1) object ID: 1.3.6.1

이는 〈그림 5〉의 왼쪽 트리에 기입한 숫자에 기인해 상위 노드 1은 iso를 의미하고 3은 org, 6은 dod, 1은 dod의 첫 번째 노드인 internet을 의미한다.



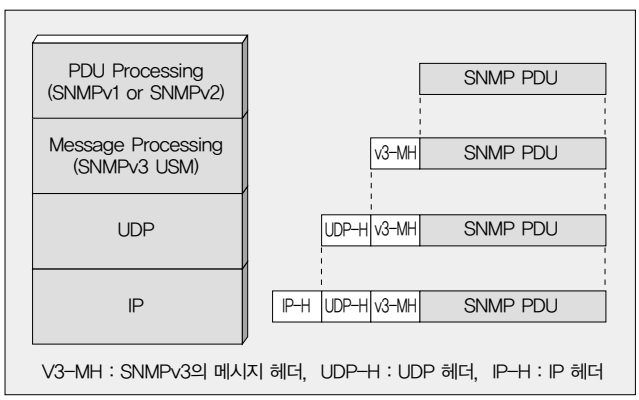
〈그림 5〉 MIB 트리

예시 2) object ID: 1.3.6.1.2.1.6

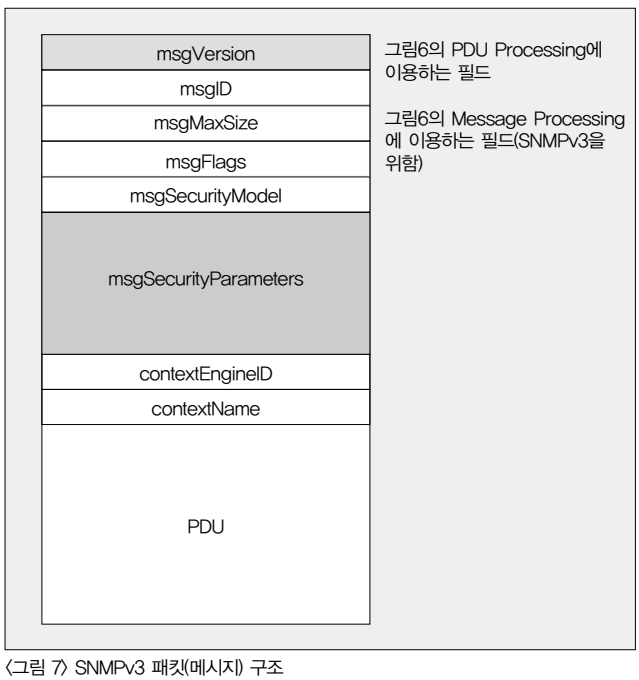
예시 2도 마찬가지로 〈그림 5〉의 트리 노드 번호에 의해 iso org dod internet mgmt mib-2 tcp를 가리키는 것이다. PDU를 생성할 때, 다음과 같은 객체 ID를 알아야 원하는 정보를 얻을 수 있다.

SNMPv3

아직까지도 SNMPv1과 SNMPv2가 쓰이고 있다. 이는 이전 버전을 무시한 채 SNMPv3와 SNMPv4를 사용하는 것은 아니라는 것이다. 물론 SNMPv3가 가장 많이 쓰이기 때문에 SNMPv3를 중심으로 정리할 것이다. 〈그림 6〉은 이전 버전과의 호환성을 제공하는 SNMPv3 모델을 나타낸다.



〈그림 6〉 SNMPv1과 SNMPv2와의 호환성을 제공하는 SNMPv3 USM(User Security Model)



〈그림 7〉 SNMPv3 패킷(메시지) 구조

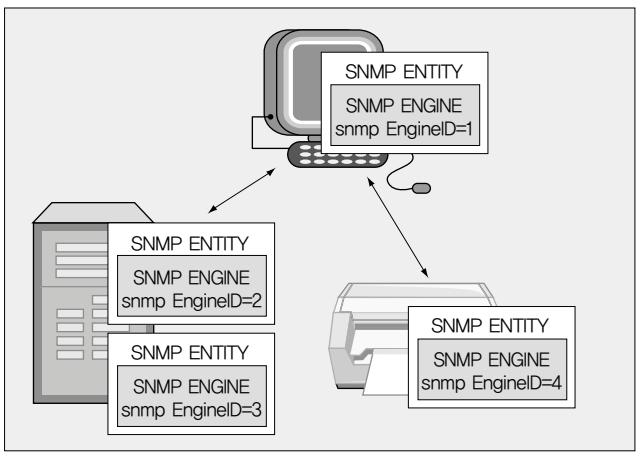
〈그림 6〉은 메시지 처리와 PDU 처리를 구분해 SNMPv3의 메시지를 이전 버전 PDU로 변환시켜 호환성을 제공한다. 그렇기 때문에 이전 버전의 SNMP 시스템을 별다른 수정 없이 계속 사용할 수 있다. 〈그림 7〉은 SNMPv3의 패킷 구조를 나타낸다. msgID부터 msgSecurityModel까지는 〈그림 6〉의 UDP 계층 위에 있는 메시지 처리 모듈(Message Processing)에서 사용하는 필드이다. SNMPv3만을 위한 필드이다. msgID는 요청과 응답 메시지를 연결하기 위한 두 객체 사이에 사용되는 유일한 식별자이다. msgMaxSize는 메시지를 송신할 때 최대 메시지크기를 나타낸다. 바이트 단위를 사용하며, 484부터 231-1까지의 범위를 갖는다. msgFlags는 reportableFlag. privFlag, authFlag 등의 PDU관련 플래그를 설정할 수 있다. reportableFlag가 1이면 레포트 PDU(응답)를 수신 받겠다는 의미이고, 0이면 레포트 PDU를 받지 않겠다는 것이다.

즉, Set, Get, Inform 같은 메시지는 reportFlag를 1로 설정하고, Response, Trap은 0으로 설정한다. 다음으로 privFlag는 암호화와 인증여부를 결정하는데, 1이면 암호화를 적용하고 0이면 인증을 적용한다. authFlag는 보안수준을 결정한다. msgSecurityModel은 어떠한 버전의 보안 모델을 이용할 것인지를 결정한다. 기본 값으로 1은 SNMPv1을 나타내고 2와 3은 각각 버전2와 버전3을 의미한다. 버전3은 USM라는 사용자 보안 모델로써 이를 위해 msgSecurityParameters 필드를 사용한다. 뒷부분에서 〈표 4〉와 함께 자세히 다룬다.

SNMPv3의 아키텍처는 이전 버전과는 다소 차이가 있다. 뷰-기반 접근 제어(View-Based Access Control Model, VACM) 방식의 보안 메커니즘을 이용한다(SNMPv3 RFC 문서에서 VACM 정책 부분 참조). VCAM 자체는 에이전트에 대한 접근 제어정책을 정의하고 원격 구성이 가능하도록 MIB를 사용한다. 구성하는 요소는 크게 5가지로 나눈다. 그룹, 보안 수준, 콘텍스트, MIB 뷰, 접근정책(Access Policy)이다. 그룹은 〈Security Model, SecutiryName〉의 파라미터로 구성할 수도 있고 단순한 groupName으로 구분할 수도 있는데, 그룹별 관리를 위한 체계이다.

보안 수준은 리소스에 대해 읽기만 허락할지 쓰기 접근까지 허락할 것인지를 결정한다. 콘텍스트는 MIB에 속하는 객체의 일부 부분집합 개념이다. 즉, contextEngineID에 의해 유일하게 식별되는 SNMP 객체는 하나 이상의 콘텍스트를 보유한다(여러 부분 집합에 속할 수 있다는 의미이다). contextEngineID의 개념은 〈그림 8〉을 보면 이해할 수 있다. MIB 뷰는 데이터베이스의 테이블 뷰와 비슷한 개념으로 SNMP에서는 읽기접근, 쓰기접근, 통지접근을 위한 세 가지 MIB 뷰를 생성해놓고 있다. 또

한 MIB 뷰에 마스크(mask)를 씌어 원하는 정보만 얻을 수 있다. 마스크의 개념은 비트 스트림으로 설명이 가능한데, 1111에 1110과 AND 연산을 하면 마지막 비트가 0이 되는 것과 같은 원리이다. 1110이 마스크 비트고, 원본 데이터는 1111이다. 〈표 3〉을 통해 MIB 뷰를 이해하도록 하자.



〈그림 8〉 EngineID의 개념

MIB 뷰	허용 함수	허용 관리자	보안 수준(Level)
인터페이스 테이블	Set	Sang Won	인증 암호화(Authentication Encryption)
인터페이스 테이블	Get / GetNext	Tae Young, Sang Won	인증 (Authentication)
시스템 그룹	Get / GetNext	John	보안 처리 없음 (None)
…	…	…	…

〈표 3〉 MIB 뷰(접근 제어 테이블)

다음은 보안문제에 대해 생각해보자. 버전3 이전은 보안상 많은 취약점을 드러냈다. 설계당시부터 보안을 무시한 그룹이 있었기 때문이다. 또한 보안기능이 포함되면 무거운 시스템이 되기 마련이다. 최대한 가벼운 솔루션을 만들기 위해 초기에는 UDP만을 지원했는데, 보안 기능까지 포함시키면 배보다 배꼽이 더 클 거라고 판단했을 것이다. 하지만 이제는 달라졌다. 아무리 작은 시스템이라도 보안 기능을 포함해도 될 만큼 하드웨어적으로 충분한 진보가 있었다. 그런 의미에서 SNMPv3 부터는 다양한 공격 위협으로부터 방어책을 마련했다. 〈표 4〉는 대표적인 공격 유형과 방어가능여부를 정리한 것이다. 〈그림 7〉의 msg SecurityParameters 필드는 공격에 방어하기 위해 사용되는 파라미터를 포함한다. 〈표 5〉에 파라미터를 모두 정리했다.

재전송 공격은 공격자가 중간에 패킷을 가로채 그 패킷을 재전송하는 것을 의미한다. 재전송이 허용되면 암호화를 해봤자 무용지물이다. 신용카드로 포인트 구매 중에 가로챈 패킷은 패킷을 서버에 전송하면 언제나 충전이 가능하다. 이와 같은 과금 시스

공격	방어가능여부	방법
재전송(Replay)	O	타임스탬프 이용
위장(Masquerade), 무결성(Integrity)	O	MD5 / SHA-1
노출(Disclosure)	O	DES
DoS(Denial of Service)	X	-
트래픽 분석	X	-

〈표 4〉 SNMPv3의 보안기능

msgAuthoritativeEngineID	재전송 방지에 사용하는 필드
msgAuthoritativeEngineBoots	
msgAuthoritativeEngineTime	
msgUserName	위장 방지, 무결성 보장을 위한 필드
msgAuthenticationParameters	
msgPrivacyParameters	정보노출(Disclosure)

〈표 5〉 msgSecurityParameters의 상세 구분

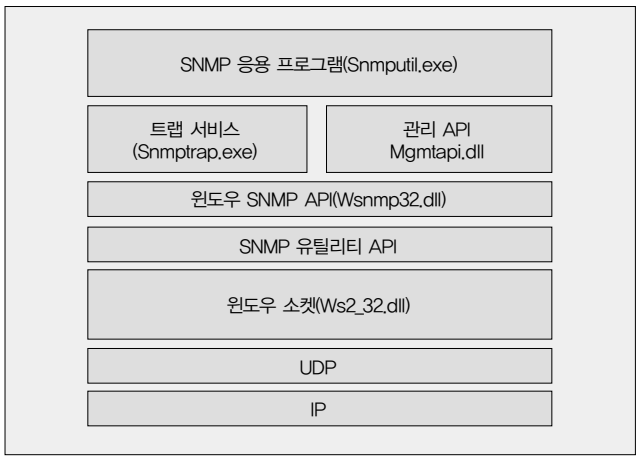
템이나 정보유출과 관련해 큰 파장을 일으킨다. 하지만 패킷을 생성할 때 유효기간을 포함하면 간단히 방어할 수 있다. SNMP는 로컬 타이머와 유효시간을 비교하는 루틴으로 공격자를 방어한다. 또한 위장을 방지하기 위해 인증하는 방식을 사용했다(MD5와 SHA를 이용한다. 본인을 증명하는 인증서를 확인하는 절차를 포함한다). 안전한 해시함수를 제작하는 원리를 다시 한 번 정리해보길 바란다.

정보노출(Disclosure)에 대한 방어는 CBC모드를 지원하는 DES를 이용해 메시지를 암호화한다. 이처럼 기본적인 보안기능을 지원하지만 아직까지 Dos나 트래픽 분석에 의한 위협은 크게 개선되지 않았다. 최근 이 부분의 취약점을 악용한 피해 사례가 발생했다. 공격자가 관리스테이션과 에이전트 간의 교환을 차단해서, 시스템관리에 막대한 손실을 입힐 수 있다. 또한 에이전트로부터 전달받는(Trap, Response) 정보를 관리스테이션이 아닌 공격자가 가로챈다면 범인에게 대문을 열어준 게 된다. 현재까지는 성능(Light Weight)과 안정성을 중심으로 발전해왔다면, 앞으로는 확실한 보안체계를 갖춰서 안전한 네트워크 관리가 가능한 방향으로 노력해야한다. 국내도 UCC와 IPv6를 접목한 서비스 등 수많은 IPv6 시범사업이 진행 중이며 많은 장비가 개발 중이다.

머지않아 전 세계의 모든 다비이스는 커다란 인터-네트워크로 연결돼 중앙관리체계가 갖춰질지도 모른다. 불법 디바이스는 원천적으로 사라질 것이다. 이때쯤 되면 팀 버너스리의 월드와이드 웹과 경쟁할만한 이노베이션이 등장할 수도 있을 것이다. 물론 중앙관리체계를 갖추고 제품시장을 독점한다는 의미는 아니다. 중앙관리국은 SNMP와 같은 표준 인터페이스만을 제공할 뿐, 시장판도는 크게 달라지 않을 것이다. 그만큼 SNMP와 같은 네트워크 관리 솔루션은 앞으로 전망이 밝다.

SNMP 라이브러리

SNMP를 직접 구현할 필요는 없다. 이미 닷넷과 자바진영에 구현된 라이브러리가 존재한다. RFC를 확장하고 싶다면, 이들 클래스를 상속받아 오버라이드 하면 된다. 우선 닷넷에서는 Malcolm Crowe에 의해 제안된 Snmp.dll과 Mib.dll를 이용해 SNMP 응용 프로그램을 작성할 수 있다. Snmp.dll은 RFC1213을 충족시키며 MIB-2를 사용한다. Mib.dll은 RFC1157기반의 객체 ID(OID)를 변환하고 mib 파일을 관리한다. 윈도우에서 SNMP를 테스트 할 수 있는 가장 일반적인 유틸리티는 snm putil.exe이다. 〈그림 9〉와 같이 Mgmtapi.dll을 통해 SNMP Utility API를 이용해 윈도우 소켓(Ws2_32.dll)에 전달하는 과정으로 동작한다.



〈그림 9〉 Snmputil.exe와 윈도우 SNMP 서비스

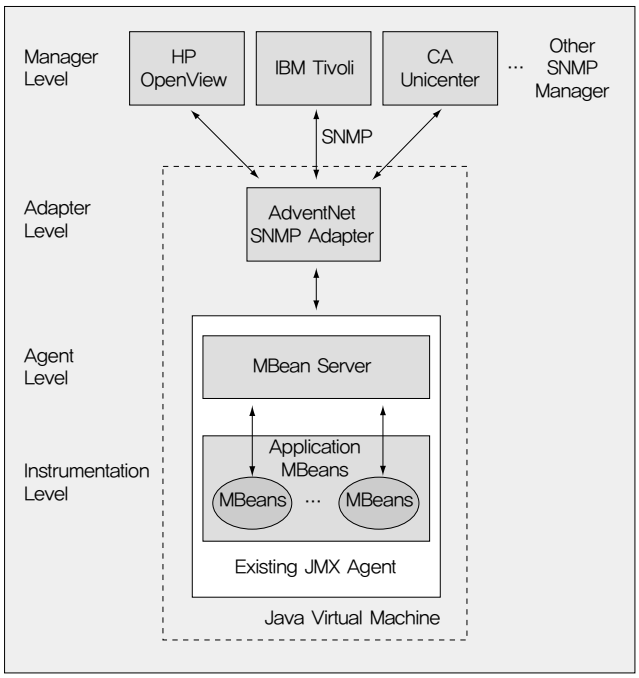
다음은 이 유틸리티의 사용 예제이다. best..student.imaso.co.kr 도메인에 있는 리소스로 접속해 GET 함수를 이용해 MIB의 system.sysContact.0 정보를 얻는다고 가정하자. 앞서 객체 ID를 얻는 방법을 설명한 것과 유사하지만 노드번호를 system이나 sysContact라는 문자열로 매핑 한 형태여서 좀 다르게 보일 수도 있다. 가독성이 높기 때문에 매핑 테이블을 가지고 문자열 형태의 객체 ID를 이용한다.

사용예: "snmputil get best..student.imaso.co.kr public system.sysContact.0"

〈리스트 1〉은 Mgmt()의 인스턴스를 생성해 객체 ID를 분석하며, ManagetSession을 이용해 접속할 에이전트를 설정한다. sess.mib.OID를 이용해 객체 ID를 매핑 된 노드번호 형태인 uint 형으로 변환해 Get의 파라미터로 전달한다. 다음은 자바에서 제공하는 API에 대한 설명이다. 동작구조는 〈그림 10〉과 같다. 관리스테이션(HP, IBM, CA)이 JMX 에이전트의 정보를 이용할 수 있는데, SNMP Adaptor 모듈이 JVM 내부와 외부의 인터페이스 역할을 한다. 윈도우에서 API DLL을 이용해 커널 서비스를 받을 수 있는 것처럼 자바에서도 Adaptor 인터페이스를 이용해 VM 서비스를 받는 구조이다. 자바 API의 특징으로는 WAS의 성능관리에 최적화됐다는 것이며, JMX 기반 Rule 엔진을 지원해 트랩발생을 위한 MBean Rule이나 한계치(Threshold)를 설정할 수 있다는 것이다. 또한 이식성이 뛰어나다는 것은 이미 알고 있는 자바의 특징이다. 특히 엔터프라이즈 웹 애플리케이션 시장에서 강세를 보이는 자바 기반 솔루션에 API 수준에서 최적화된 WAS 성능관리 기능을 제공한다는 것은 눈여겨 볼만한 부분이다. 예제코드는 API관련 문서에 충분이 정리돼 있으니 참고하길 바란다.

〈리스트 1〉 닷넷(C#)을 이용한 SNMP 구현 예제

```
RFC1157.Mgmt mib = new RFC1157.Mgmt();
ManagerSession sess=new
ManagerSession(best..student.imaso.co.kr,public);
ManagerItem mi=new ManagerItem(sess,mib.OID(mgmt.mib-
2.system.sysContact.0));
Console.WriteLine(mi.Value.ToString());
// Get, GetNext 등의 질의
Universal[] results = sess.Get(..);
```



〈그림 10〉 SNMP Adaptor 구조(자바)

끝으로 WMI(Windows Management Instrumentation)에 대해 간단히 소개하는 것으로 네트워크 관리에 대한 내용을 마무

리 하겠다. WMI는 윈도우에서 제공하는 관리도구의 일종으로 WINAPI를 제공한다. SNMP 프로토콜을 이용하고 있으며 원격 컴퓨터의 정보를 얻고 프로그램을 원격으로 실행하거나 시스템을 종료시킬 수도 있다. 〈리스트 2〉는 원격에서 메모장(Notepad)을 실행하는 예제이고, 〈리스트 3〉은 원격 컴퓨터를 재부팅하는 예제이다. 간혹 윈도우원격(mstsc)으로 접속해서 로그인하고 메모장 띄우면 되지 않느냐고 묻는 사람이 있을 수 있다. 하지만 관리 서버가 한 두 대가 아닌 10대만 돼도 관리가 불가능해진다.

WMI를 이용해 간단한 스크립트를 만들고 실행하는 것만으로 10대의 서버에서 동작하는 어떤 서비스를 재시작할 수 있다고 생각해보면, WMI의 유용함을 알 수 있다. 물론 관리자 계정을 안다는 가정에서 기능들이 허용된다. 이전 시간에 다룬 커버로스를 이해했다면 윈도우 계정이 커버로스 기반으로 관리된다는 것을 기억하기 바란다.

WMI를 쓰다보면 데이터베이스의 SQL문과 비슷한 스크립트를 발견한다. 〈리스트 4〉에도 선택문(Select)을 사용했다. WMI를 이용한 응용프로그램을 제작하는 것은 결국 리소스 쿼리문을 작성하는 문법에 부딪친다. 데이터베이스 응용프로그램을 개발해본 경험이 있다면, 쿼리문 작성하는데 대부분 시간이 소요된다는 것을 알 것이다. 이와 동일한 원리다. 쿼리문을 작성해 MIB로부터 원하는 정보를 얻고 데이터를 갱신할 수 도 있다. 물론 에이전트로가 윈도우로 제한된다는 단점이 있지만, 윈도우 기반 에이전트라면 쉽고 안전하게 시스템을 관리할 수 있다.

〈리스트 4〉는 원격서버로 접속해 사용가능한 공간이 20% 미만인 모든 드라이브 파티션을 나열하는 스크립트이다. 'select FreeSpace,Size,Name from Win32_LogicalDisk where DriveType=3' 구문은 데이터베이스 쿼리문과 유사하다. DriveType 3은 하드디스크를 의미하며 if문을 이용해 20% 미만 여부를 체크한다. Win32_LogicalDisk와 같은 MIB 그룹을 미리 정의해서 쉽게 응용 프로그램을 작성할 수 있다. 마찬가지로 〈리스트 2〉에서는 Win32_OperatingSystem을, 〈리스트 3〉에서는 Win32_Process를 이용했다. 이들 그룹은 개별 클래스라고 생각할 수 있다. 즉 〈리스트 2〉에서 Reboot() 메소드는 Win32_ OperatingSystem 클래스에서 제공하는 멤버함수이다.

시스코를 중심으로 수많은 업체가 네트워크 관리 장비와 소프트웨어에 관심을 갖고 있다. 시스코의 자격검정은 이미 IPv6을 반영했다. 국내의 경우 정보통신부가 여러 시범서비스를 한다는 점에서 IPv6는 대세로 자리 잡고 있다. 디바이스마다 IP가 할당되면, 모든 전자기기에 네트워크 기술이 융합돼 시너지 효과를 창출한다. 특히 지난 시간에 다뤘던 멀티미디어 관련 기술과

〈리스트 2〉 WMI를 이용한 예제(원격 메모장 실행)

```
set process =
GetObject("winmgmts:{impersonationLevel=impersonate}
!Win32_Process")
result = process.Create ("notepad.exe",null,null,processid)
WScript.Echo "메서드에서 반환된 결과 = " & result
WScript.Echo "새 프로세스의 ID : " & processed
출처 :http://cdmanii.tistory.com/152
```

〈리스트 3〉 WMI를 이용한 예제(원격 재부팅)

```
Set OpSysSet =
GetObject("winmgmts:{impersonationLevel=impersonate,(Remot
eShutdo
wn)}//alexn-pc ").ExecQuery("select * from
Win32_OperatingSystem
where Primary=true")
for each OpSys in OpSysSet
OpSys.Reboot()
Next
```

〈리스트 4〉 WMI를 이용한 예제(디스크 공간 체크)

```
Set DiskSet =
GetObject("winmgmts:{impersonationLevel=impersonate}")
.ExecQuery("select FreeSpace,Size,Name from
Win32_LogicalDisk where
DriveType=3")
for each Disk in DiskSet
If (Disk.FreeSpace/Disk.Size) < 0.20 Then
WScript.Echo + Disk.Name + " 드라이브의 공간이 부족합니다."
End If
Next
```

IPv6가 접목되면 진정한 멀티미디어 방송통신융합 이노베이션을 이루게 될 것이라 확신한다. 이와 더불어 수많은 디바이스를 관리할 네트워크 관리 시스템과 전문가의 수요가 필요하다. 이런 관점에서 4회에 걸친 필자의 글이 도움이 됐길 진심으로 바란다. 지금까지 설명한 보안 솔루션들을 충분히 활용하기 바란다.

이달의 디스켓 : SNMP.zip

참고 자료

1. Network Security Essentials, Second edition (William Stalling)
2. Essential SNMP (OReilly, Douglas Mauro)
3. http://www.snmp.com/snmpv3/v3white.html
4. http://www.ibr.cs.tu-bs.de/ietf/snmpv3/
5. http://www.ietf.org/html.charters/snmpv3-charter.html
6. Alko Pras의「SNMPv3 Architecture」
7. http://www.wtcs.org/snmp4tpc/
8. http://msdn2.microsoft.com/en-us/library/aa394572.aspx (WMI)
9. http://mc4j.org/confluence/display/mc4j/Home (JMX)