Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.23, NO.6, Dec. 2013 ISSN 2288-2715(Online)
http://dx.doi.org/10.13089/JKIISC.2013.23.6.1181

GOOSE 프로토콜 환경에서 Snort 기반의 침입 탐지 시스템 개발*

김 형 동,[1†] 김 기 현,[2] 하 재 철[1‡]
[1]호서대학교 정보보호학과, [2]에스지에이(주)

Development of Intrusion Detection System for GOOSE Protocol Based on the Snort*

Hyeong-Dong Kim,[1†] Ki-Hyun Kim,[2] Jae-Cheol Ha[1‡]
[1]Dept. of Information Security, Hoseo University, [2]SGA Co., Ltd.

요 약

디지털 변전 자동화 시스템의 국제 표준인 IEC 61850에서는 IED(Intelligent Electronic Device)간의 상호 통신을 위해 GOOSE(Generic Object Oriented Substation Event) 프로토콜을 사용하고 있다. 그러나 GOOSE 프로토콜은 TCP/IP 프로토콜과 유사하게 이더넷에 기반하여 운용되므로 다양한 형태의 보안 위협에 노출되어 있다. 따라서 본 논문에서는 소프트웨어 기반의 공개 침입 탐지 시스템(Intrusion Detection System)으로 사용되는 Snort를 이용하여 GOOSE 프로토콜에 대한 IDS를 개발하였다. 개발된 IDS에는 디코딩 과정과 전처리 과정을 통해 GOOSE 패킷에 대한 키워드 탐색 기능과 DoS 공격 탐지 기능이 구현되어 있다. 또한, GOOSE 네트워크 실험 환경을 구축하고 GOOSE 패킷 생성 및 송・수신 실험으로 통해 IDS 시스템이 정상적으로 동작함을 확인하였다.

ABSTRACT

The GOOSE(Generic Object Oriented Substation Event) is used as a network protocol to communicate between IEDs(Intelligent Electronic Devices) in international standard IEC 61850 of substation automation system. Nevertheless, the GOOSE protocol is facing many similar threats used in TCP/IP protocol due to ethernet-based operation. In this paper, we develop a IDS(Intrusion Detection System) for secure GOOSE Protocol using open software-based IDS Snort. In this IDS, two security functions for keyword search and DoS attack detection are implemented through improvement of decoding and preprocessing component modules. And we also implement the GOOSE IDS and verify its accuracy using GOOSE packet generation and communication experiment.

Keywords: GOOSE Protocol, IDS, Snort, DoS Attack Detection

I. 서 론

오늘날 인터넷 활용이 보편화되고 유비쿼터스 사회로 진화하면서 우리가 사용하는 대부분의 전자기기들은 인터넷을 이용할 수 있도록 변화되고 있다. 최근에는 첨단 IT 기술들을 융합한 스마트 그리드와 같은 인프라 구축 사업들이 정부 주도하에서 대규모로 추진되

접수일(2013년 9월 9일), 수정일(2013년 10월 22일), 게재확정일(2013년 10월 23일)
* 본 논문은 교육부의 재원으로 지원을 받아 수행된 산학협력 선도대학(LINC) 육성사업의 연구결과입니다.
† 주저자, karuceace@nate.com
‡ 교신저자, jcha@hoseo.edu(Corresponding author)

고 있다. 특히, 전력망 개선과 관련한 핵심 기술 중 하나는 고속 네트워크 기술을 활용한 변전소의 설비 및 기기의 자동화와 관련한 연구 및 개발이다.

한편, 변전소 자동화 국제 규격이 IEC 61850으로 단일화된 이후 변전 시스템에 적용 가능한 IED(Intelligent Electronic Device)를 비롯한 다양한 제품들이 개발되어 사용되고 있다[1, 2]. IEC 61850 기반의 변전 자동화 시스템(Substation Automation System)은 IED, 상위 운영시스템, 게이트웨이 등으로 구성되며 기존의 제어 케이블에 의한 정보 전달 기능을 이더넷 환경으로 전환하여 운영하고 있다[3, 4].

GOOSE(Generic Object Oriented Substation Event)는 디지털 변전 자동화 시스템 국제 표준인 IEC 61850에서 IED간의 통신을 위해 개발되었으며 멀티캐스팅 전송 방식을 채택하고 있다. 하지만 GOOSE에서는 전송 효율성과 구현의 용이성을 강조한 반면, 보안에 관한 고려 요소는 거의 없어 사용자의 운영 환경에 따라 보안 기능을 별도로 개발하여 사용하여야 한다. 따라서 GOOSE 프로토콜의 안전한 사용을 위해서는 프로토콜 운용상 발생할 수 있는 보안 취약점들을 사전에 분석하고 보안 기능들을 추가할 필요성이 있다[5-7].

한편, Snort는 패킷 로깅 및 실시간 트래픽 분석 기능은 물론 네트워크 기반 침입 탐지 기능까지 갖춘 소프트웨어 기반 툴로서 개발 소스가 공개된 프로그램이다[8-10]. Snort의 구성은 크게 스니퍼, 전처리기, 탐지 엔진, 경고 및 로깅 모듈로 구성되어 있으며 이더넷 환경에 기반한 TCP/IP 프로토콜을 중심으로 개발되었다. 특히, Snort에서는 악의적인 패킷에 대해 탐지 엔진 부분에 해당 시스니처(signature)와 매칭되는지 여부를 판단하는 오용(misuse) 탐지 기법을 사용한다.

본 논문에서는 변전소 자동화 시스템에서의 정보 전송을 위해 사용되는 GOOSE 프로토콜에 대한 침입 탐지 시스템(Intrusion Detection System)을 개발하고자 한다. 즉, GOOSE 프로토콜이 TCP/IP와 같은 이더넷 환경에서 사용되는 점과 Snort 소스가 공개되어 있는 점을 고려하여 Snort를 활용한 IDS를 구현하고자 한다[11]. 이를 위해 변전소 자동화 시스템에서 사용하는 GOOSE 프로토콜을 분석하고 이 프로토콜이 Snort 침입 탐지 시스템에 탑재될 수 있도록 전처리기를 설계하였다. 전처리기에는 패킷의 디코딩 기능뿐만 아니라 키워드 탐색 기능과 DoS(Denial of Service) 공격 탐지 기능이 구현되어 있다. 그리고 GOOSE 패킷 생성 기능 및 송·수신 실험을 통해 Snort의 침입 탐지 기능이 정상적으로 동작함을 확인하였다.

본 논문의 2장에서는 Snort의 구조 및 개발에 필요한 핵심 기능을 설명한다. 3장과 4장에서는 GOOSE 프로토콜에 대한 패킷 분석과 더불어 전처리기에 키워드 탐색 기능과 DoS 공격 탐지 기능을 개발한다, 제 5장에서는 GOOSE 패킷 생성 및 송·수신 실험을 통해 시스템이 정상적으로 동작함을 확인하였다. 마지막으로 6장에서 결론을 맺는다.

II. Snort 구조 분석

Snort의 구조는 크게 스니퍼(sniffer), 전처리기(preprocessor), 탐지 엔진(detection engine), 경고 및 로깅(alert and logging)으로 나누어진다. 다음 Fig. 1은 Snort의 구조를 나타낸 것이다. 스니퍼에서는 패킷 수집 및 각 프로토콜에 따라 디코딩을 수행하는데 전처리기에서는 패킷이 탐지 엔진으로 보내지기 전에 패킷 재조합이나 비정상 행위 탐지 기능을 수행한다. 탐지 엔진에서는 사전에 정의된 규칙 또는 사용자가 정의한 규칙과 패킷의 페이로드를 비교하여 매칭되는지 확인한다. 마지막으로 경고 및 로깅에서는 규칙의 설정에 따라 탐지된 패킷을 처리하게 된다. Snort 구성 요소들은 플러그인 형태로 이루어져 있어 사용자가 필요한 기능을 수정하거나 변경할 수 있으며 쉽게 추가할 수 있어 확장성이 매우 뛰어나다.

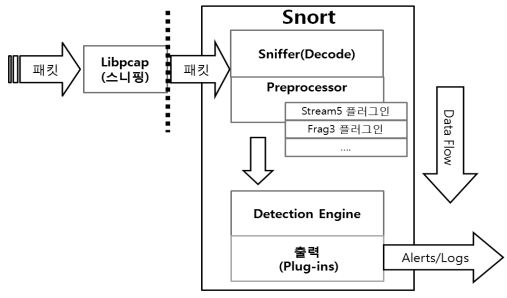

Fig.1. Structure of Snort

2.1 스니퍼

스니퍼는 네트워크 모드를 수집(promiscuous) 모드로 동작시킴으로써 네트워크 이더넷 카드로 들어

오는 모든 패킷을 수집하는 기능을 수행한다. 스니퍼에서는 패킷 수집 라이브러리인 Libpcap을 이용하여 패킷을 캡처한다. 또한 네트워크 트래픽을 감시 및 분석하고 사용자가 보기 쉬운 형태로 변환하는 디코딩 기능을 수행한다.

2.2 전처리기

전처리기는 디코딩된 패킷을 받은 후 탐지 엔진으로 패킷을 보내기 전에 사전 처리하는 단계이다. 이 단계에서는 분할된 패킷을 재조합하기도 하며, 탐지 엔진에서 탐지할 수 없는 특별 행위를 탐지한 후 경고 및 로깅을 남긴다. 전처리기는 각각의 기능들이 플러그인 형태로 적용되기 때문에 시스템 처리 능력 및 상황에 따라 전처리 기능들을 활성화 혹은 비활성화해서 운용할 수 있다. 만약 시스템의 처리 속도가 낮은 상태라면 불필요한 플러그인 기능을 비활성화시킴으로써 전처리기의 속도를 향상시킬 수 있다. 따라서 패킷에 대한 특별한 사전 처리가 필요한 경우에는 전처리기에 플러그인을 신규로 만들어 추가할 수 있다.

2.3 탐지 엔진

탐지 엔진은 전처리기로부터 패킷을 받아 기본 규칙이나 사용자가 미리 정의한 규칙과 비교하여 일치하는 패킷이면 해당 경고 및 로깅을 남긴다. 탐지 엔진에서 사용하는 규칙은 일정한 형식을 가지고 있는데 하나의 규칙은 규칙 헤더와 규칙 옵션으로 구분한다. 규칙 헤더는 기본적으로 해당 패킷 탐지 시 취할 행동(alert, log, pass 등), 네트워크 패킷의 종류(TCP, UDP 등), 출발지와 목적지 IP 주소 및 포트 등으로 이루어져 있다. 규칙 옵션은 부가적으로 설정하는 다양한 탐지 옵션들로 구성되어 있다. 결국 탐지 엔진은 유입된 패킷의 페이로드와 규칙 옵션 필드의 컨텐츠(content)와 비교하여 일치할 경우 시스템에 경고 및 로깅을 남기고 만약 이상 패킷이 탐지되지 않았다면 그냥 패킷을 통과시키게 된다.

2.4 경고 및 로깅

경고 및 로깅은 전처리기 혹은 탐지 엔진에서 이상 패킷이 탐지 되었을 때 마지막으로 패킷을 전달하는 곳이다. 즉, 유입된 패킷이 유해하다고 탐지가 되었을 때 설정된 액션에 따라 syslog를 이용하거나 특정 디렉토리의 특정 파일 또는 데이터베이스에 로그를 남기고 경고를 발생하게 된다.

III. GOOSE 프로토콜 및 패킷 구조

3.1 GOOSE 프로토콜 개요

1990년대부터 미국과 유럽을 중심으로 변전소 자동화 시스템을 표준화하기 위한 연구가 IEC(International Electrotechnical Commission) TC(Technical Committee)57과 TC59를 중심으로 진행되어 왔으며, 그 결과로 2005년 IEC 61850 국제 표준이 제정되었다. IEC 61850에서는 변전 시스템의 통신 및 제어를 위해 기존의 복잡한 구리선 형태의 전송망을 개방형 이더넷 기반 광케이블 디지털 방식으로 변환하기 위한 자동화 기능을 정의하고 있으며 여기에서 사용하는 통신 프로토콜 중 대표적인 것이 GOOSE 프로토콜이다.

이더넷 기반 망에서 운용되는 TCP, UDP, ICMP, IP 등의 프로토콜은 OSI 7계층 중 2, 3, 4계층을 통과하여 정보를 전달하지만 GOOSE 프로토콜에서는 3계층인 네트워크 계층과 4계층인 전송 계층을 통과하지 않는다. 즉, GOOSE 프로토콜은 빠른 연결과 적은 통신량을 지향하는 프로토콜로서 물리 주소로 통신을 하는 MAC 주소 기반 프로토콜이다.

3.2 GOOSE 프로토콜 구조 분석

Fig. 2는 GOOSE 프로토콜에서 사용하는 패킷 구조를 나타낸 것이다. 처음 목적지 및 출발지 MAC 주소부터 Ethertype까지는 이더넷 헤더에 해당하며, APPID(Application ID)부터 APDU 전까지가 GOOSE 프로토콜 헤더이다. GOOSE 프로토콜 Ethertype은 0x88b8로 정의되어 있으며 APPID는 OSI 7 계층 중 응용 계층에서 사용되는 서비스 번호이다. Reserved 필드는 예약 필드로서 추후 보안적인 측면에서 사용될 예비 필드이고 각각 2바이트를 할당한다.

마지막 APDU(Application Protocol Data Unit)는 응용 계층에서 실체 간 주고받는 데이터 단위로서 응용 프로토콜 제어 정보와 응용 계층 사용자 데이터를 포함한다. APDU 구성 요소는 다음과 같이 13개의 특별한 전송 내용을 담고 있다.

Tag 0: gocbRef
(Goode Control Block Reference)
Tag 1: timeAllowedtoLive
Tag 3: datSet(Dataset)
Tag 4: goID(GOOSE Identifier)
Tag 5: stNum(Status Number)
Tag 6: sqNum(Sequence Number)
Tag 7: test
Tag 8: confRev(Configuration Revision)
Tag 9: ndsCom(Needs Commission)
Tag 10: numdataSetEntries
(Number of Dataset Entries)
Tag 11: allData(Sequence of data)
Tag 12: security(any optional,
reserved for digital signature)

Header MAC (0~11)	Destination Address
	Source Address
Priority Tagged (13~15)	TPID
	TCI
(16~17)	EtherType
Length start (18~19)	APPID
(22~25)	Reserved 1 Reserved 2
From 26	APDU

Fig.2. Structure of GOOSE packet

IV. GOOSE 침입 탐지 시스템 개발

Snort를 이용하여 GOOSE 프로토콜에 대한 침입 탐지 시스템을 개발하기 위해서는 패킷 디코딩, 전처리기, 탐지엔진 그리고 경고 및 로깅 기능을 구현해야 한다. 특히, GOOSE는 이더넷 망을 이용하지만 TCP/IP 프로토콜과 전혀 다른 패킷 포맷 구조를 가지고 있기 때문에 기존 시스템에 각 기능을 추가하는 형식으로 침입 탐지 시스템을 설계하였다.

4.1 GOOSE 디코딩 기능 설계

Snort에서는 캡처 라이브러리를 통해 패킷이 캡처되면 Decode 함수를 호출한다. 해당 프로토콜의 Decode 함수는 decode.c 파일에 있는데, 내부의 DecodeEthPkt 함수에서 IP, IPv6, ARP 등이 정의한 Ethertype에 맞게 디코드 함수를 호출한다. 앞서 언급한 바와 같이 GOOSE 프로토콜의 TYPE은 0x88b8이다. Fig. 3은 decode.c 파일에 포함된 DecodeEthPkt 함수에서 GOOSE 타입에 따라 GOOSE 디코더를 호출하도록 코드를 작성한 것이다. Fig. 3과 같이 설계가 되면 GOOSE 패킷이 들어올 경우 GOOSE 디코드 함수를 자동으로 호출하게 된다.

```
...생략
case ETHERNET_TYPE_IPV6 :
   DecodeIPV6(p->pkt + ETHERNET_HEAD
   ER_LEN, cap_len -ETHERNET_HEADER
   _LEN), p);
 PREPROC_PROFILE_END(decodePerfStats);
    return;

case ETHERNET_TYPE_GOOSE :
   DecodeGOOSE(p->pkt + ETHERNET_HEA
   DER_LEN, cap_len -ETHERNET_HEADER
   _LEN), p);
 PREPROC_PROFILE_END(decodePerfStats);
    return;
```

Fig.3. Code of DecodeGOOSE function

GOOSE 디코더를 설계하기 위해서는 앞 장에서 분석한 GOOSE 패킷을 저장하기 위한 구조체를 미리 정의하여야 한다. Fig. 4는 goose.h 헤더 파일에 정의한 GOOSE 패킷 헤더 구조체이다. 즉, APPID와 길이 그리고 예약 영역을 정의한 후 GOOSE 프로토콜의 헤더 정보를 저장된다.

```
typedef struct _GOOSEHdr{
     uint16_t appid;
     uint16_t len;
     uint16_t res1;
     uint16_t res2;
}  GOOSEHdr;
```

Fig.4. Structure of GOOSE packet header

그 다음으로 GOOSE 패킷을 디코딩하기 위한 함

수를 독립적으로 개발하여야 한다. Fig. 5는 GOOSE 패킷을 디코딩하기 위해 개발된 Decode-GOOSE 함수이다.

```
void DecodeGOOSE(const uint8_t * pkt,
                 uint32_t len, Packet *p)
{
   pc.goose++;
   if(len < GOOSE_HEADER_LEN)
   {
      DEBUG_WRAP(DebugMessage(DEBU
      G_DECODE, "GOOSE:GOOSE short
      Header(%d bytes) \n", len));
      return;
    }
    p->gooseh = (GOOSEHdr *) pkt;
}
```

Fig.5. DecodeGOOSE function

DecodeGOOSE 함수에서는 먼저 GOOSE 패킷에 대한 카운터를 증가시킨 후에 정의한 GOOSE 헤더의 길이와 패킷의 길이를 비교하게 된다. 만약 유입된 패킷의 헤더의 길이가 GOOSE 헤더 길이보다 작을 경우 경고 메시지를 출력하고, 그게 아니라면 패킷 구조체에 있는 gooseh 변수에 GOOSE 헤더 시작점을 저장하게 된다.

4.2 전처리기 및 키워드 매칭 기능

본 논문에서는 전처리기 단계에서 전력망 관리자가 지정한 키워드를 설정하여 자동 탐색되도록 설계하였다. 사용자는 전처리 단계에서 패킷을 검사하여 APDU 내부의 필요한 단어를 검색하거나 이상 정보를 검출하는데 활용할 수 있다. 키워드를 탐지하는 방법은 Aho-Corasick 패턴 매칭 알고리즘[14]을 사용하였다. Aho-Corasick 알고리즘은 Aho와 Corasick에 의해 트리 구조를 기반으로 개발된 패턴 매칭 기법으로서 여러 개의 검색어를 단 한 번의 검색으로 추출할 수 있는 기법이다. Aho-Corasick 알고리즘의 탐색 방법은 탐색하고자 하는 키워드 패턴을 이용하여 Fig. 6과 같은 맵을 먼저 생성하고, 패킷이 들어오면 순차적으로 검색하여 패턴의 일치 여부를 확인한다. Fig. 6은 검색하고자 하는 키워드 셋이 {he, she, his, hers}라고 가정했을 때의 맵을 도시한 것이다.

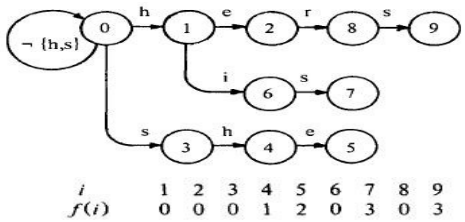

Fig.6. Example of pattern map in Aho-Corasick algorithm

이 맵에서 각 노드에 적힌 숫자는 State ID이며, 화살표 위에 적힌 알파벳은 다음 State로의 이동 조건이다. 만약 만족하는 값이 들어오지 않을 경우 Fail 함수가 동작한다. 예를 들면 State 4에서 e가 들어오면 State 5로 이동하게 되지만, 그 외의 문자들이 들어오면 Fail 함수 $f(i)$가 동작하게 되어 State 1로 이동하게 된다.

논문에서는 GOOSE 전처리기에 패킷 내부의 키워드 검사를 위해 Aho-Corasick 알고리즘을 사용하여 키워드 매칭 기능을 설계하였으며 Fig. 7은 전처리기를 수행하는 과정을 나타낸 것이다.

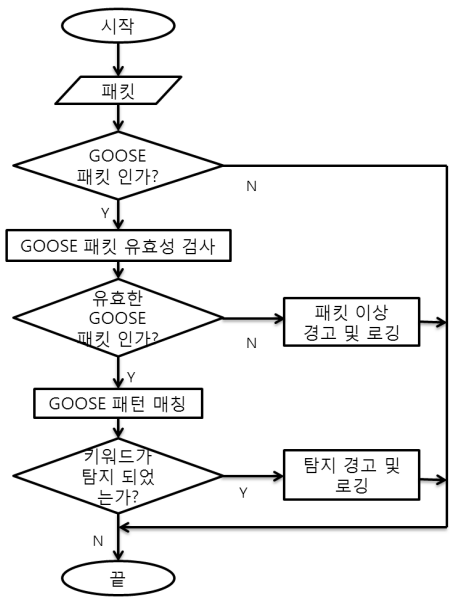

Fig.7. Operation procedure of GOOSE pre-processing

Fig. 7에서 볼 수 있듯이 먼저 패킷이 들어오게 되면 패킷의 타입을 보고 GOOSE 패킷인지를 확인하는 절차를 수행한다. 그리고 패킷에 대한 유효성을 검사

한 후에 설정된 키워드들과 매칭되는지 확인하고, 탐지되었을 경우 경고 및 로깅을 남기도록 설계되었다.

이를 위해 GOOSE 전처리기에 수행에 필요한 정보 및 데이터 구조체 포맷을 정의하는 헤더(goose.h) 파일과 실질적으로 GOOSE 패킷을 처리하는 파일(goose.c)을 구현하였다. 실제로 전처리기 과정을 수행하는 goose.c 파일 안에 GOOSE 전처리기를 초기화하는 함수, 패킷을 처리하는 함수, 키워드 탐색하는 함수, 종료하는 함수 등을 설계하여 코딩하였다.

Fig. 8은 패킷을 처리하는 GOOSE_Process 함수 중 일부분을 나타낸 것이다. 그림에서 보는 바와 같이 GOOSE 전처리기를 처리할 때 먼저 전체적인 환경 설정 파일을 확인한다. 그 다음 GOOSE 프로토콜의 타입과 실제 패킷의 타입이 일치하는지를 검사한 후 일치한다면 설정한 키워드를 확인하는 과정으로 진행된다.

```
...생략...
if(config == NULL)
  return;
if(ntohs(p->ether_header->ethernet_type) ==
ETHERNET_TYPE_GOOSE){
...생략...
 keyword = _dpd.serchAPI->search_instance_
find(goose_keyword_search_mpse,(const char
*)p->pkt_data, p->gooseh->length - 8, 0,
GOOSE_SearchKeywordFound);
}
```

Fig.8. A part of GOOSE_Process function

4.3 DoS 공격 탐지 기능 개발

DoS 공격은 인터넷망이나 전력망에서 가장 위협적인 공격 중 하나이다. 특히, 전력망과 같은 기간망은 서비스 지연이나 파괴로 인해 실시간적으로 큰 피해를 가져올 가능성이 있기 때문에 사전 탐지 조치가 중요하다. 기존 Snort에서는 SYN 플러딩, ICMP 플러딩과 같은 TCP/IP에 대한 DoS 공격을 규칙화하여 탐지하는 방법을 이미 사용하고 있다[12, 13].

본 논문에서는 GOOSE 패킷에 대한 DoS 공격에 대비한 침입 탐지 기능을 전처리기 내부에 개발하고자 한다. 하지만 TCP/IP와 다른 패킷 구조를 가지고 있기 때문에 기존 인터넷망 DoS 공격 탐지 방법을 활용할 수는 없다. 고도화된 TCP/IP DoS 공격 탐지 방법을 변경하여 활용할 수도 있지만, GOOSE와 같은 경량화 프로토콜 특성에 맞게 트래픽 양을 분석하고 실시간으로 트래픽 분포를 검사하여 DoS 공격 여부를 판단하고자 한다.

DoS 공격 탐지 방법은 크게 두 단계로 구성하였는데 첫 번째는 정상 트래픽을 분석하여 일정 시간당 GOOSE 평균 트래픽 양과 표준 편차를 정의하는 것이다. 즉, 아래와 같이 정상 상태라고 판단할 수 있는 경우의 임계치(threshold) P_t를 지정하게 된다.

$$P_t = \mu + k\delta \tag{1}$$

여기서 μ는 초당 패킷의 수, δ는 표준 편차 그리고 k는 임계치를 결정하는 상수 값이다. 패킷이 아래와 같은 정규 분포 특성을 가지고 유입된다고 가정하면 상수 값 k에 따라 DoS 공격의 판단 범위를 정할 수 있다.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \tag{2}$$

즉, Fig. 9에서 보는 바와 같이 k를 1로 가정하면 68.3%만을 정상 패킷으로 판단하며 임계치보다 많은 패킷이 유입되면 DoS 공격으로 판단한다. 따라서 k를 2로 하면 95.4%를, 3으로 하면 99.7%를 신뢰할 수 있으므로 관리자가 적절한 임계치를 정하여 운영할 수 있도록 설계하였다.

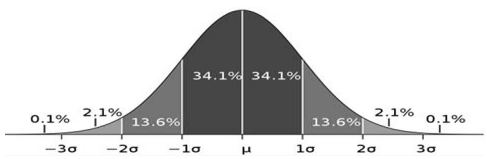

Fig.9. Threshold and reliability for DoS attack

두 번째 단계에서는 실제 DoS 공격을 탐지하는 단계로 고정된 시간동안 실제 유입되는 트래픽 양을 산출하여 임계치와 비교함으로써 DoS 공격 여부를 탐지하는 단계이다. 다음 Fig. 10은 DoS 공격을 탐지하는 알고리즘을 구체적으로 도시한 것이다.

DoS 공격 탐지 프로그램도 GOOSE 전처리기 내에서 헤더(dos.h)와 패킷을 처리하는 코드(dos.c)파일로 작성할 수 있다. 헤더 파일에는 DoS 탐지를 수행할 때 필요한 값들과 구조체를 정의하는데 Fig. 11의 (a)는 헤더 파일에서 GOOSE 패킷의 환경에 대한 구조체이다.

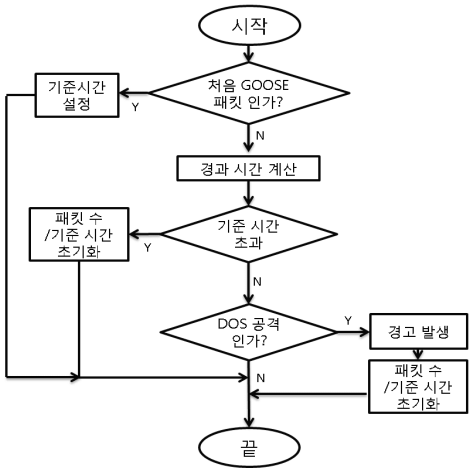

Fig.10. Intrusion detection of DoS attack

```
typedef struct _GOOSEConfig
{
    uint32_t                    setGooseFlag;
    uint16_t                    baseTime;
    double                      replayTime;
    uint32_t                    goosePktThreshold;
} GOOSEConfig;
```

(a) GOOSE configuration structure

```
typedef struct _GOOSEDOS
{
    uint32_t                    nGoosePkt;

    struct timeval              checkTime;

} GOOSEDOS;
```

(b) DoS packet structure

Fig.11. Intrusion detection header of DoS attack on GOOSE protocol

이 구조체에서 baseTime은 설정한 단위 시간 정보를 나타내며, goosePktThreshold는 패킷의 수를 의미한다. 또한 (b)는 GOOSE DoS의 구조체를 나타낸 것이다. 이 구조체에는 공격 분석을 위한 유입된 전체 GOOSE 패킷의 수와 처음 패킷이 들어온 시간 등을 저장한다. 또한, 헤더 파일에는 Fig. 11에 나타낸 구조체 정보뿐만 아니라 GOOSE 패킷 프레임 사이즈 및 DoS 공격 문자열 등 각 값에 대한 정의도 포함되어 있다.

실제적으로 DoS 공격을 탐지하는 기능은 dos.c 파일에 설계하였다. 즉, dos.c 파일 내에 초기화 함수, 환경 설정, GOOSE 패킷 처리, DoS 공격 탐지 함수 등을 설계하였다. DoS 공격 탐지는 단위 시간을 계산하고 설정된 값과 비교한 후 조건이 성립됨을 판단함으로써 이루어진다. 다음 Fig. 12는 DoS 공격인지를 판단하는 코드의 핵심 부분이다. 여기에서 df_sec는 최초 패킷으로부터 경과된 시간을 의미하며 baseTime은 설정된 단위 시간이고, goosePkt-Threshold는 DoS 공격 탐지를 위해 설정된 패킷 수를 나타낸다. 이 두 값을 비교한 후 DoS 공격인지 아닌지를 판단하게 된다.

```
if((df_sec < conf->baseTime) &&
   (gooseDoS.nGoosePkt  >=
        Conf->goosePktThreshold))
{
   gooseDoS.nGoosePkt = 0;
   gooseDoS.checkTime = g_pktTime;
    _dpd.alertAdd(GENERATOR_SPP_GOOSE,
                GOOSE_DOS_ATTACK, 1, 0, 3,
                GOOSE_DOS_ATTACK_STR, 0);
    return;
}
```

Fig.12. Core part for DoS attack detection function

V. GOOSE 전처리기 운영 실험

본 논문에서는 GOOSE 프로토콜상에서 공격자의 침입을 탐지하는 기능을 이더넷 기반의 공개 소프트웨어인 Snort에 전처리기 형태로 설계한 후 기능의 동작 여부를 검증하였다. 컴퓨터 실험에서는 키워드 탐색 실험 및 DoS 공격을 유도하기 위해 GOOSE 프로토콜상에서 생성된 패킷을 미리 파일 형태로 (goose.pcap) 저장한 뒤 발신용 툴에 의해 전송되도록 하였다[15]. 패킷 발신용 툴은 단위 시간당 전송할 수 있는 패킷의 양을 조절할 수 있으며, 최대 5,000개의 패킷까지 한 번에 전송하도록 구성하였다. Fig. 13은 실제 GOOSE 패킷 구조를 캡쳐하여 나타낸 것이다.

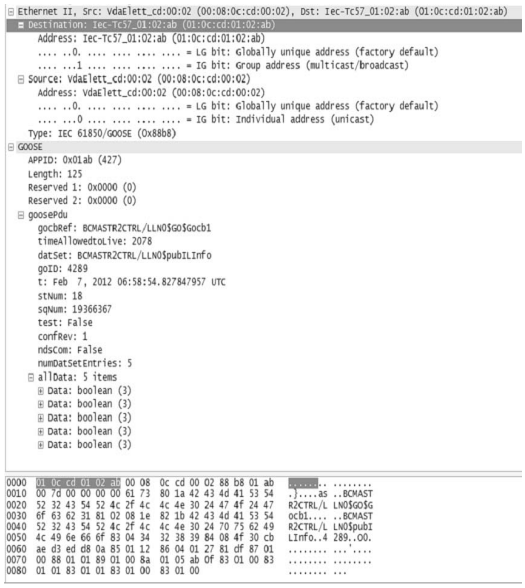

Fig.13. Captured GOOSE packet

실험을 위한 환경은 다음과 같다.

- CPU : Intel duo CPU 3.2GHz
- RAM : 2GB
- 실험 환경 : 우분투 10.2.4, Centos 5.3

이와 같은 패킷이 침입탐지 시스템에 유입되었을 때 디코딩 과정에서 패킷을 정확히 부호화하였으며 전처리기의 플러그인 동작을 수행하였다. 논문에서는 설계한 GOOSE 전처리기를 플러그인 형식으로 개발하고 이 플러그인을 활성화하는 방법으로 실험하였다.

Fig. 14의 (a)는 GOOSE 전처리기를 통해 탐지하고자 하는 키워드가 정확히 검출됨을 보이는 화면을 나타낸 것이다. 실험에서는 총 20개의 패킷을 대상으로 검사하였으며 그 중에서 해당 키워드를 검색한 패킷이 모두 16개라는 것을 나타내고 있다. 또한 그림의 (b)는 논문에서 설계한 DoS 공격 탐지 기능에 대한

```
GOOSE Preprocessor:
GOOSE Keyword Set :
[attack, bob, hack, bad, backdoor, flow, boom, CTRL, gcb, exploit,
mail, sell, confine, getuid, getsystem, root, chmod, su, setup, admin]
    GOOSE packets decoded: 20
        keyword detected: 16
         invalid packets: 0
Detect Keyword
 attack: 3
 admin: 5
 gcb: 2
 backdoor: 1
 su: 5
```

(a) Experimental result of keyword search operation

```
Sep  3 17:54:34 localhost dhclient: DHCPACK from 192.168.138.254
Sep  3 17:54:34 localhost dhclient: bound to 192.168.138.140 — renewal in 850 s
econds.
Sep  3 18:08:44 localhost dhclient: DHCPREQUEST on eth0 to 192.168.138.254 port
67
Sep  3 18:08:44 localhost dhclient: DHCPACK from 192.168.138.254
Sep  3 18:08:44 localhost dhclient: bound to 192.168.138.140 — renewal in 784 s
econds.
Sep  3 18:21:48 localhost dhclient: DHCPREQUEST on eth0 to 192.168.138.254 port
67
Sep  3 18:21:48 localhost dhclient: DHCPACK from 192.168.138.254
Sep  3 18:21:48 localhost dhclient: bound to 192.168.138.140 — renewal in 786 s
econds.
Sep  3 18:28:50 localhost kernel: device eth0 entered promiscuous mode
Sep  3 18:29:01 localhost sa_sensor: [150:1:1] GOOSE DoS attack detected [Classi
fication: Detection of a Denial of Service Attack] [Priority: 2] 00:30:A7:02:5C:
56 -> 01:0C:CD:01:00:14
Sep  3 18:29:05 localhost sa_sensor: [150:1:1] GOOSE DoS attack detected [Classi
fication: Detection of a Denial of Service Attack] [Priority: 2] 00:09:8E:FE:81:
14 -> 01:0C:CD:01:01:14
```

(b) System log of DOS attack

Fig.14. Experimental result GOOSE preprocessing operation

실험 결과 화면이다. 논문에서는 DoS 공격이 탐지되었을 때 시스템 로그(/var/log/messages)에 출력하도록 미리 설계하였다. 따라서 GOOSE에 대한 DoS 공격이 탐지되면 시스템 로그에는 그림과 같은 메시지가 출력된다.

VI. 결 론

본 논문에서는 스마트 그리드 구현의 핵심 요소 중 하나인 변전소 자동화 시스템에 사용되는 GOOSE 프로토콜에 대한 패킷 탐색 및 침입 탐지 시스템을 개발하고자 하였다. GOOSE 프로토콜이 이더넷 기반 환경에서 운영되는 점을 고려하여 이더넷 기반의 공개 패킷 분석 소프트웨어인 Snort를 이용하여 구현하였다. Snort에서 사용하는 모듈 중 디코딩 모듈과 전처리기 모듈을 이용하여 패킷 분석 및 패킷 탐지 기능을 설계하였다. 개발된 전처리기에는 주요 키워드 검색 기능과 DoS 공격 탐지 기능이 플러그인 형태로 탑재되어 있다. 또한 개발된 소프트웨어를 실제 컴퓨터 통신 환경에서 실험한 결과 패킷이 정상적으로 분석되고 송·수신 절차에 따라 정확히 침입 탐지 기능이 수행됨을 확인하였다. 본 논문에서 구현한 GOOSE 침입 탐지 시스템은 공개 소프트웨어로 개발되어 유연성이 뛰어나며 향후 고도의 네트워크 침입 공격을 방어하는 시스템으로 확장하여 사용할 수 있다.

References

[1] International Electrotechnical Commission, "Communication Network and System in Substation," IEC 61850, April,

2004.

〔2〕 N. N. Dinh, G. S. Kim and H. H. Lee, "A study on GOOSE communication based on IEC 61850 using MMS ease lite," International Conference on Control, Automation and Systems (ICCAS'07), pp. 1873-1877, Oct. 2007.

〔3〕 I. Ali, and M. S. Thomas, "GOOSE based protection scheme implementation & testing in laboratory," Innovative Smart Grid Technologies (ISGT'11), pp. 1-7, Jan. 2011.

〔4〕 N. H. Lee and B. T. Jang, "A Study on the Communication Test of Substation Automation System based on IEC 61850," The Transactions of the Korean Institute of Electrical Engineers, 57(1), pp. 14-19, Jan. 2008.

〔5〕 H. S. Yang, S. S. Kim, and H. S. Jang, "Optimized security algorithm for IEC61850 based power utility system," Journal of Electrical Engineering & Technology, vol 7, no. 3. pp. 443-450, 2012.

〔6〕 H. Falk, "Securing IEC 61850," Power and Energy Society General Meeting-Conversion and Delivery of Electrical Energy in the 21st Century, pp. 1-3, July 2008.

〔7〕 J. Hoyos, M. Dehus, and T. X. Brown, "Exploiting the GOOSE protocol: A practical attack on cyber- infrastructure," IEEE Globecom Work shops: Smart Grid Communication: Design for Performance, pp. 1508-1513, Dec. 2012.

〔8〕 J. Beale, J. C. Foster, J. Faircloth, and B. Caswell, Snort 2.0 Intrusion Detection, 2nd Ed., Syngress, Feb. 2003.

〔9〕 R. U. Rehman, Intrusion Detection with SNORT: Advanced IDS Techniques Using SNORT, Apache, MySQL, PHP, and ACID, Prentice Hall, May 2003.

〔10〕 A. R. Baker and J. Esler, Snort IDS and IPS Toolkit (Jay Beale's Open Source Security), 4th Ed., Syngress, Oct. 2010.

〔11〕 Korea Information Security Agency(KISA), IDS implementation using Snort, May 2005. Available at http://twoseven.kr/linux2/files/snort.pdf

〔12〕 M. K. Son, D. H. Lee, K. J. Kim, "Designing and Realization of the System for the Improvement of Processing Capability of Intrusion Detection by Using O/S Information," Journal of Information and Security, 6(2), June 2006.

〔13〕 K. W. Lee, J. W. Ji, H. W. Chun, S. J. Youk, and G. Lee, "Traffic Analysis Technique for Intrusion Detection in Wireless Network," Journal of Security Engineering, 7(6), Dec. 2010.

〔14〕 A. V. Aho and M. J. Corasick, "Efficient String Matching : An Aid to Bibliographic Search," Comm. ACM, vol. 18, no. 6, pp. 333-340, 1975.

〔15〕 Sourcefire Inc., "Snort Users Manual 2.9.3," The Snort Project, May 2012. Available at http://www.snort.org/assets/166/snort_manual.pdf

〈저자소개〉

김 형 동 (Hyeong-Dong Kim) 학생회원
2012년 2월: 호서대학교 정보보호학과 졸업
2012년 3월~현재: 호서대학교 대학원 정보보호학과 석사 과정
〈관심분야〉 부채널 공격, 무선 네트워크 보안, 스마트폰 보안,

김 기 현 (Ki-Hyun Kim) 정회원
1993년 2월: 경북대학교 전자공학과 졸업
1995년 2월: 경북대학교 전자공학과 석사
2011년 8월: 충북대학교 컴퓨터공학과 박사
2009년 7월~현재: 에스지에이㈜ 연구센터장
2013년 3월~현재: 호서대학교 정보보호학과 겸임교수
〈관심분야〉 시스템 및 네트워크 보안, 보안관제, 전자문서 보안

하 재 철 (Jae-Cheol Ha) 종신회원
1989년 2월: 경북대학교 전자공학과 졸업
1993년 8월: 경북대학교 전자공학과 석사
1998년 2월: 경북대학교 전자공학과 박사
1998년 3월~2007년 2월: 나사렛대학교 정보통신학과 부교수
2007년 3월~현재: 호서대학교 정보보호학과 교수
〈관심분야〉 정보보호, 네트워크 보안, 부채널 공격