# 콘텐츠 보호를 위한 경량화 침입탐지 기술
# Lightweight Intrusion Detection Technology for Content Protection

박성준, 김봉한(청주대학교 컴퓨터정보공학과)

**차 례**

1. 서론
2. 침입탐지 기술의 분류
3. 경량화 침입탐지 설계
4. 시나리오를 통한 침입탐지
5. 결론

■ keyword : | 정보보호 | 콘텐츠보호 | 침입탐지 | IDS |

## 1. 서론

최근 몇 년간의 세계 주요 해킹 사고 및 국내 보안 사고로 인해 사회적 파장이 커지면서 사이버보안의 중요성 및 사이버 테러 대응이 주요 이슈로 부상하고 있다. 세계적으로 인터넷 활용범위 확대와 정보통신기술에 대한 의존도 증가 등으로 사이버보안 및 정보보호의 중요성에 대한 인식이 확산되고 사이버보안의 필요성이 점차적으로 증가하고 있으며 이 시점에서 사용자가 서비스거부공격 및 스니핑 공격으로부터의 탐지 기능 및 패킷 탐지를 이용하여 사이버 공격에 대해 즉시 대처 할 수 있는 침입탐지 시스템이 필요하게 되었다.

사이버 테러나 해킹 사고 등 네트워크상에서의 공격들이 일어날 때 일반 사용자는 이런 공격이 눈에 보이지 않기 때문에 그 피해의 크기나 심각성을 짐작하기가 쉽지 않을 뿐만 아니라 자신의 개인 정보들을 제대로 보호하는 것에 대해서도 소홀하다. 또한 사용자가 무분별하게 여러 서버에 접속하고 있을 때, 외부의 불법적인 침입자의 악성 IP를 알고 싶어도 알기가 어려운 실정이다.

따라서 본 논문에서는 실제로 일어난 이벤트를 포함하는 로그를 사용하므로 보다 정확하고 사용자와 파일의 접근활동, 파일의 허용의 변화, 새로운 실행파일을 설치하려는 시도 그리고 특정한 서비스의 접근을 감시할 수 있을 뿐만 아니라 NIDS(Network based Intrusion Detection System)에서 놓치는 공격 탐지인 시스템 내부에서의 공격을 확인하고 대응할 수 있고 다양한 로그 자료를 통해 정확한 침입탐지를 할 수 있는 경량화된 침입탐지 기술을 구현하고자 한다.

## 2. 침입탐지 기술

### 2.1 침입탐지 기술의 개념

침입(Intrusion)이란 정보 접근, 정보 조작, 시스템 무기력화 등에 대한 고의적이고 불법적인 잠재 가능성과 자원의 무결성(integrity), 기밀성(confidentiality), 가용성(availability)을 저해하는 일련의 행위들의 집합 또는 컴퓨터 시스템의 보안정책을 파괴하는 행위를 말한다. 권한이 없는 사용자의 정보시스템에 대한 계획적이거나 우연한 접근 또는 행위도 포함된다. 침입탐지(Intrusion Detection)이란 침입을 시도하거나, 침입 행위가 일어나고 있거나, 침입이 발생한 것을 확인하는 절차이다.

침입탐지 기술은 컴퓨터 시스템에 대한 비인가자의 접속, 정보의 비정상적인 사용, 오용, 남용 등의 침입 행위를 규정하여 컴퓨터 시스템 또는 네트워크 상에서 침입이 발생했거나, 침입 행위가 일어나고 있거나, 침입 시도를 탐지하는 시스템으로 가능하면 실시간으로 처리하는 기술을 의미한다.

### 2.2 침입탐지 기술의 분류

#### 2.2.1 오용 탐지

특정 공격에 관한 분석 결과를 바탕으로 패턴을 설정

하고 패턴(시그니처)과의 비교를 통하여 일치하는 경우 불법 침입으로 간주하는 기법이다. 장점으로 오탐률(False Positive)이 낮으며 추론기반, 지식베이스를 이용하여 트로이목마, 백도어공격 등의 탐지가 가능하다. 단점으로 새로운 공격 탐지를 위해 지속적인 공격 패턴 갱신 필요하고 패턴에 없는 새로운 공격에 대해서는 탐지 불가능하다. 또한 속도 문제로 대량의 자료를 분석하는 데는 부적합하다.

#### 2.2.2 비정상행위 탐지

비정상 행위 침입탐지는 사용자의 행동양식을 분석한 후 정상적인 행동과 비교해 이상한 행동, 급격한 변화가 발견되면 불법 침입으로 탐지하는 방법이다. 정량적인 분석, 통계적 분석을 사용하며 형태 관찰, 프로파일 생성, 프로파일 기반으로 이상 여부를 확인(로그인 횟수, 패킷 및 I/O 트래픽 등)하여 정상인지 비정상인지를 판단한다.

장점으로 인공지능 알고리즘 사용으로 스스로 판단하여 수작업의 패턴 업데이트 불필요하고 알려지지 않은 새로운 공격을 탐지할 수 있다. 단점으로 오탐률(False Positive)이 높은 편이다. 즉, 정상적인 사용인데, 비정상으로 탐지하는 비율이 높다. 정상과 비정상 구분을 위한 임계치 설정에 어려움이 있다.

#### 2.2.3 네트워크기반 침입탐지

네트워크 기반 침입탐지는 감지기(미러링 포트, 태핑 장비 등)를 이용하고 무차별 모드(promiscuous mode)에서 동작하는 NIC에 설치되어 있다. NIC가 모든 트래픽을 캡처한 정보를 분석기로 전송해 특정 유형의 규칙이 있는지를 분석한다. 네트워크 기반 침입탐지는 자신으로 향하는 트래픽은 관찰할 수 없다. 장점으로 네트워크에서 실행되어 서버의 성능 저하가 없고 네트워크에서 발생하는 여러 유형의 침입을 탐지할 수 있으며 침입자의 IDS공격에 대한 방어가 가능하며 존재 사실도 숨길 수 있다. 단점으로 네트워크 패킷이 암호화되어 전송될 때 침입 탐지 불가능하고 네트워크 트래픽이 많이 증가하게 되면 성능이 저하하는 문제가 발행하며 오탐률(False Positive)이 높은 편이다.

#### 2.2.4 호스트기반 침입탐지

호스트기반 침입탐지는 서버나 PC와 같이 호스트에 Agent형태로 설치하여 비정상적인 행동을 탐지한다. 로그나 입출력 패킷을 검사하여 시스템의 중요한 파일이나 설정값을 삭제 또는 변경하여 시스템을 위험에 노출시키는 행위를 탐지한다. 호스트기반 침입탐지는 네트워크 트래픽 분석 및 모니터링이 목적이라면, H-IDS는 시스템 자체의 위험성을 탐지하는데 목적이 있다. 장점으로 기록되는 다양한 로그 자료를 통해 정확한 침입방지, 호스트에 대한 명백한 침투에 대해 탐지, 트로이목마, 백도어, 내부자에 의한 공격탐지 및 차단이 가능하다. 단점으로 침입자에 의한 로그 자료의 변조 가능성이 있으며 DoS공격으로 침입탐지 기능이 무력화할 수 있다. 또한 호스트 성능에 의존적이며, 리소스 사용으로 서버에 부하가 발생한다.

**표 1. 호스트 IDS와 네트워크 IDS 특징 비교**

| 구분 | 호스트기반 침입탐지 | 네트워크기반 침입탐지 |
|---|---|---|
| 장점 | -시스템의 각종 자원 정보 파악 용이<br>-시스템 별 정확한 탐지 및 분석 수행 가능<br>-시스템 별 실시간 로그 확인 가능<br>-내부 사용자 및 사용자 레벨에서의 공격 시도 탐지 가능 | -각 운영체제 별 에이전트 소프트웨어 불필요<br>-네트워크 기반 공격 패턴 탐지 가능<br>-전체 네트워크 트래픽 모니터링 및 사용현황 정보제공 기능 |
| 단점 | -각 호스트 별로 설치되므로 설치/배포/관리가 용이치 않음<br>-시스템 부하 발생 및 성능 저하 가능성 제기<br>-운영체제 별로 에이전트가 개발 및 관리되어야 함<br>-패킷 헤더 분석 기능이 미약하므로 모든 종류의 공격 탐지가 어려움<br>-공격자가 시스템에 침입시에는 IDS 로그정보 접근 및 삭제가 가능함 | -네트워크 성능저하 및 병목 현상 발생 가능성<br>-시스템 레벨의 해킹 및 파일변조 공격은 탐지가 용이치 않음<br>-NIDS를 경유하지 않는 공격은 탐지 불가능<br>-호스트 IDS에 비해 공격 탐지 효율이 떨어짐 |

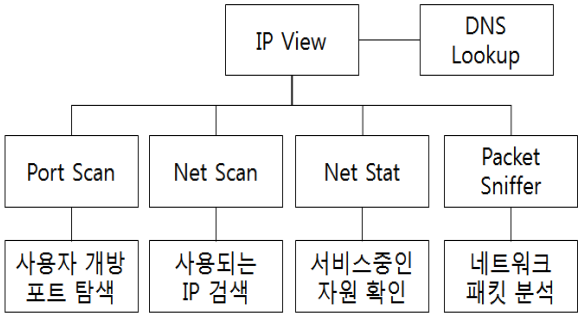
## 3. 경량화 침입탐지 설계

### 3.1 전체 구성도

경량화 침입탐지를 설계 및 구현하기 위하여 그림 1과 같이, IP View를 이용하여 로컬 컴퓨터의 호스트명과 아이피를 구할 수 있게 하였고 Net Scan을 이용하여 로컬 시스템에 사용되고 있는 포트 및 서비스 중인 네트워크 프로세스들의 상태 정보와 네트워크 연결 상태를 확인하는 기능을 구현하였으며, Port Scan을 통해 관리하고 있는 시스템이나 기타 시스템에 대해서 어떤 포트가 열려 있고 닫혀 있는지 확인하는 기능과 로컬 시스템의 지정된 범위에 대한 포트 정보를 검사하고 파일로 결과를 저장하는 기능을 구현하였다.

또한 Net Stat을 이용하여 로컬 시스템에 사용되고 있는 포트 및 서비스 중인 네트워크 프로세스들의 상태 정

보와 네트워크 연결 상태를 확인하고, 정보를 파일로 저장하는 기능을 구현하였다. Packet Sniffer를 이용하여 네트워크 패킷을 분석하는 도구로 사용할 수 있게 구현하였고 포렌식 수사에서 사용되는 DNS Lookup을 이용하여 특정 도메인에 대한 IP 주소와 호스트 정보를 검색할 수 있게 하여 네트워크 엔지니어나 서버 관리자들에게도 유용한 정보를 제공할 수 있게 구현하였다.




▶▶ 그림 1. 전체 구성도

## 3.2 IP View 상세 설계도

IP View 모듈은 컴퓨터의 호스트명과 IP를 구하는 기초적인 네트워크 모듈이다. System.Net 네임스페이스는 다양한 네트워크 프로토콜에 대한 간단한 프로그래밍 인터페이스를 제공하고, System.Net 네임스페이스에 대한 구성 설정에 프로그래밍 방식으로 액세스하여 웹 리소스에 대한 캐시 정책을 정의하고, 네트워크 트래픽 데이터와 네트워크 주소 정보에 액세스하는 기능을 제공하는 클래스, 메서드, 속성 등의 인터페이스를 포함한다.

```
private void btnOk_Click(object sender, EventArgs e)
{
  string hostname = null;
  IPAddress[] ips;
  hostname = Dns.GetHostName();
  ips = Dns.GetHostAddresses(hostname);
  foreach (IPAddress ip in ips)
  {     this.lbIp.Items.Add("호스트명 : " + hostname);
        this.lbIp.Items.Add("아이피 : " + ip.ToString());
  }   }
```

## 3.3 Net Stat 상세 설계도

네트워크 연결 상태를 체크하는 'Net Stat'콘솔 프로그램을 윈도우 버전으로 구현한 것이다. 이 프로그램을 통해 로컬 시스템에 사용되고 있는 포트 및 서비스 중인 네트워크 프로세스들의 상태 정보와 네트워크 연결 상태를 확인하고, 이 정보를 파일로 저장하는 기능을 가지고 있다. NetView()는 주기적으로 while 반복문을 실행하면서 네트워크 상태 정보를 [IvNetState] 컨트롤에 나타내는 작업을 수행한다.

```
private void NetView()
        while(true)
          { this.CheckBool = true;
            NCheck();
            this.lvNetState.Items.Clear();
            TcpConnectionInformation[]
            tcpConnections = ipProperties.GetActiveTcpConnections();
            int i = 0;
            foreach(TcpConnectionInformation NetInfo in tcpConnections)
             {              string[]  AddInfo  =  new  string[6]      14:

                 { NetInfo.LocalEndPoint.Address.ToString(),
                   NetInfo.LocalEndPoint.Port.ToString(),
                   NetInfo.RemoteEndPoint.Address.ToString(),
                   NetInfo.RemoteEndPoint.Port.ToString(),
                   NetInfo.State.ToString(), i.ToString());
                   OnView.Invoke(false, AddInfo);
                   i++;          }
              this.CheckBool = false;
              NCheck();
              Thread.Sleep(30000);          }
```

OnNewView() 사용자 정의 메서드는 델리게이트에 대입되어 수행되며 [IvNetState] 컨트롤에 네트워크 연결 정보를 나타내는 작업을 수행한다.

```
private void OnNewView(bool flags, string[] AddInfo)
{
    if (flags == true)
        this.lvNetState.Items.Clear();
    else
    { int i = Convert.ToInt32(AddInfo[5]);
      this.lvNetState.Items.Add(AddInfo[0]);
      this.lvNetState.Items[i].SubItems.Add(AddInfo[1]);
      this.lvNetState.Items[i].SubItems.Add(AddInfo[2]);
      this.lvNetState.Items[i].SubItems.Add(AddInfo[3]);
      this.lvNetState.Items[i].SubItems.Add(AddInfo[4]);
      if (AddInfo[1] == LocPort)
      this.lvNetState.Items[i].SubItems[0].BackColor = Color.GreenYellow;
      if (AddInfo[2] == RemoAdd)
       this.lvNetState.Items[i].SubItems[0].BackColor = Color.LightPink;
      if (AddInfo[3] == RemoPort)
       this.lvNetState.Items[i].SubItems[0].BackColor = Color.Aqua;
    } }
```

## 3.4 DNS Lookup 상세 설계도

DNS Lookup은 도메인을 이용하여 IP 주소를 구하는 프로그램이다. btnSearch_Click() 이벤트 핸들러는 [검색] 버튼을 더블클릭하여 생성한 핸들러로 도메인에 대한 IP 정보를 [listAddr] 컨트롤에 나타내는 작업을 수행한다.

```
private void btnSearch_Click(object sender, EventArgs e)
{
    string HostName = null;
    if (this.txtHost.Text.Contains("://")==true)
        {  HostName = this.txtHost.Text.Replace("http://","");  }
    else {HostName = this.txtHost.Text;}
    try
     {   IPHostEntry ipe = Dns.GetHostEntry(HostName);
         IPAddress[] addrs = ipe.AddressList;
         if (listAddr.Items.Count > 0)
         listAddr.Items.Clear();
         foreach (IPAddress addr in addrs)
      {   listAddr.Items.Add(addr);    }
}  }
```

## 3.5 Net Scan 상세 설계도

네트워크 스캐너 프로그램은 내부 네트워크네 연결된 컴퓨터를 스캔하는데 지정된 IP에 따라 검색한다. 이러한 프로그램은 내부 공가니 분리되어 있고 사용자 컴퓨터에 정확히 IP가 부여되어 사용되지 않는 환경에서 어떤 IP가 사용되는지를 쉽게 검색할 수 있는 장점이 있다. NetworkCheck() 사용자 정의 메서드는 지정된 IP 대역에 연결된 컴퓨터를 검색하여 IvScan 컨트롤에 나타내는 작업을 수행한다.

```
private void NetworkCheck()
  {
   for (int a = aa[0]; a <= bb[0]; a++)
    {
     for (int b = aa[1]; b <= bb[1]; b++)
      {
       for (int c = aa[2]; c <= bb[2]; c++)
        {
         for (int d = aa[3]; d <= bb[3]; d++)
          { this.pgrScan.Value = d;
            ipScan = IPAddress.Parse(a.ToString() + "."
             + b.ToString() + "." + c.ToString() + "." + d.ToString());
            Ping pingSender = new Ping();
            PingOptions options = new PingOptions();
            options.DontFragment = true
            string data = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
            byte[] buffer = Encoding.ASCII.GetBytes(data);
            int timeout = 150;
            PingReply reply = pingSender.Send(ipScan, timeout,
                    buffer, options);
           if (reply.Status == IPStatus.Success)
            { hostname = GetHostName(ipScan);
              macaddr = GetMacUsingARP(ipScan.ToString());
                        ListViewItem itm = new ListViewItem();
                        itm.Text = ipScan.ToString();
                        itm.SubItems.Add(hostname);
                        itm.SubItems.Add(macaddr);
                        this.lvScan.Items.Add(itm);
            }
        }
  } }
```

## 3.6 Port Scan 상세 설계도

Port Scan은 관리하고 있는 시스템이나 기타 시스템에 대해서 어떤 포트가 열려 있고 닫혀 있는지 확인하는 기능을 구현한 프로그램이다. 로컬 시스템의 지정된 범위에 대한 포트 정보를 검사하고 파일로 결과를 저자하는 기능을 가지고 있다. PortScanner() 사용자 정의 메서드는 지정된 포트 정보에 따라 순차적으로 OPEN 여부에 대한 확인을 진행하며 결과를 파일로 저장한다.

```
private void PortScanner()
  {
   int i, intstart, intend;
   this.lblFile.Text = "생성파일 : " + strFile;
   StreamWriter sw = new StreamWriter(strFile);
   scanIp = IPAddress.Parse(this.txtIp.Text);
   intstart = Convert.ToInt32(this.txtStart.Text);
   intend = Convert.ToInt32(this.txtEnd.Text);
   sw.WriteLine();
   for (i = intstart; i <= intend; i++)
   {   this.pgbScan.Value = i;
       try
       {   IPEndPoint endpoint = new IPEndPoint(scanIp, i);
           Socket sSocket =
             new Socket(AddressFamily.InterNetwork,
             SocketType.Stream, ProtocolType.Tcp);
           sSocket.Connect(endpoint);
           this.lvScan.Items.Add(new ListViewItem(new
               string[] {i.ToString(), "open" }));
           continue;
       }   }
  this.btnStart.Enabled = true;
  this.btnFile.Enabled = true;
  Process myProcess = new Process();
  myProcess.StartInfo.FileName = strFile;
  myProcess.Start();
  PortScan.Abort();
     }
```

## 3.7 Packet Sniffer 상세 설계도

Packet Sniffer은 네트워크 패킷을 분석하기 위한 분석 도구로도 사용되는 네트워크 패킷 스니핑 프로그램을 구현했다. 이 프로그램은 패킷을 상세히 분석하지는 않지만, IP, TCP, UDP로 분류하여 다양한 정보를 확인할 수 있다. 패킷 분석기 또는 패킷 스니퍼라고 하며, 네트워크의 일부나 디지털 네트워크를 통하는 트래픽의 내용을 저장하거나 가로채는 기능을 하는 소프트웨어 또는 하드웨어이다. 프로토콜 분석기라고도 불리며, 특정한 종류의 네트워크에서는 이더넷 스니퍼(ethernet sniffer) 또는 무선 스니퍼(wireless sniffer)라고 불린다. 데이터 스트림은 네트워크를 통해 흐르며, 스니퍼는 각 패킷을 잡아내서 디코딩하여, 적절한 RFC나 다른 규격에 따라 내용을 분석한다.

```
private void tsbtnStar_Click(object sender, EventArgs e)
  {  try
     {  if (!bContinueCapturing)
        { this.tsbtnStar.Enabled = false;
          this.tsbtnStop.Enabled = true;
          bContinueCapturing = true;
          mainSocket = new Socket(AddressFamily.InterNetwork,
              SocketType.Raw, ProtocolType.IP);
          mainSocket.Bind(new
           IPEndPoint(IPAddress.Parse(this.tscbIp.Text), 0));
          mainSocket.SetSocketOption(SocketOptionLevel.IP,
               SocketOptionName.HeaderIncluded, true);
          byte[] byTrue = new byte[4] { 1, 0, 0, 0 };
          byte[] byOut = new byte[4] { 1, 0, 0, 0 };
          mainSocket.IOControl(IOControlCode.ReceiveAll, byTrue, byOut);
          mainSocket.BeginReceive(byteData, 0, byteData.Length,
          SocketFlags.None, new AsyncCallback(OnReceive), null);
              }       }
     }
```

## 4. 시나리오를 통한 침입탐지

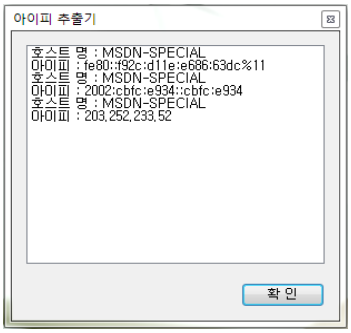
설계를 통하여 표 2와 같이, C#을 사용하여 소스코드를 구현하였고 GUI 구현 및 이미지 제작은 Adobe Photoshop을 사용하여 구현하였다.

**표 2. 구현환경**

| 구성요소 | 구현 환경 |
|---|---|
| 하드웨어 | 데스크탑<br>노트북 |
| 소프트웨어 | C#<br>Visual Studio 2005<br>Adobe Photoshop CC |

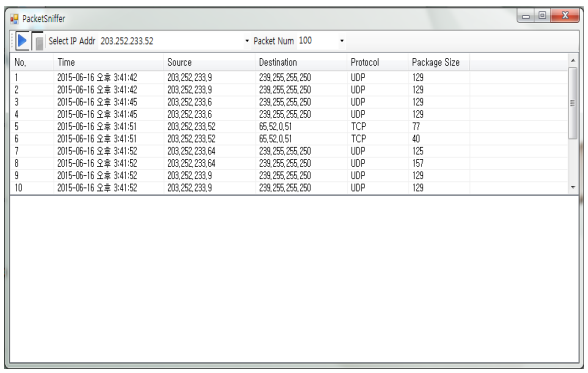
그림 2는 구현된 경량화 침입탐지의 초기 화면이고 그림 3은 IP View의 구동 화면이다. 사용자의 호스트명과 IP 주소를 나타낸다.


▶▶ 그림 2. 초기화면



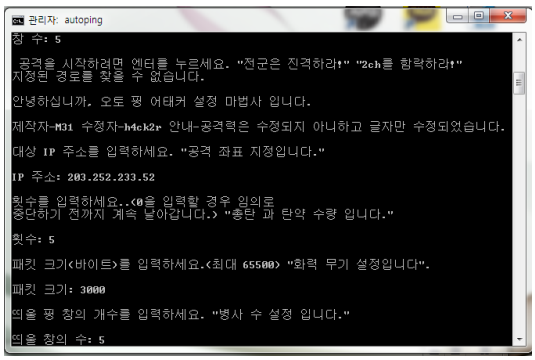

▶▶ 그림 3. IP View 실행 화면

Packet Sniffer를 실행 시키고 아이피 주소를 선택한 후 패킷 사이즈를 정하여 실행 버튼을 클릭하게 되면 그림 4와 같은 화면이 출력되며, 시간 별로 아이피 주소와 프로토콜, 수신 되고 있는 패킷 정보가 출력 된다.



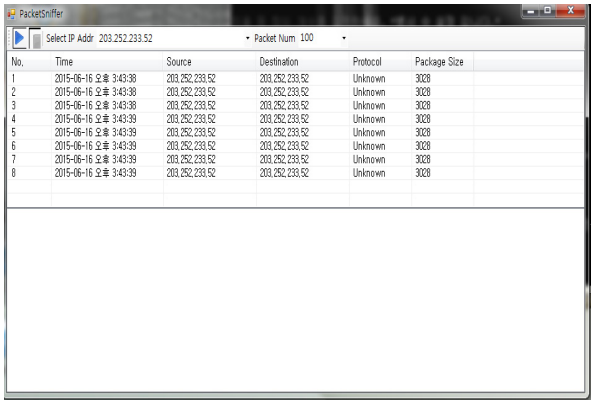

▶▶ 그림 4. Packet Sniffer 정상 패킷 수신 실행 화면

그림 5와 같이, 패킷 공격 프로그램을 이용하여 패킷을 전송할 IP 주소를 입력하고 패킷 크기를 설정한다.(IP 주소는 '203.252.233.52'이며 패킷 크기는 '3000'으로 설정함.)




▶▶ 그림 5. 패킷 공격 시도

공격을 시행하면 Packet Sniffer에서 수신 되는 패킷을 감지한다. 그림 6과 같이 패킷 크기가 공격에서 설정한대로 3000이상으로 수신 되며 정보가 출력 되는 것을 볼 수 있다.




▶▶ 그림 6. Packet Sniffer 이상 패킷 수신 화면

## 5. 결론

구현한 경량화 침입탐지는 외부의 침입자로부터 내부의 네트워크 및 자원을 보호하기 위한 목적으로 인증 및 허가되지 않은 사용자를 탐지하며 무분별한 여러 네트워크상에서의 공격을 예방할 수 있는 1차적인 방법이다. 구현된 침입탐지는 컴퓨터 사용에 익숙하지 않은 사용자들도 패킷 탐지나 로그 저장 등을 할 수 있도록 사용성, 단순성, 경량화에 중점을 두었다.

경량화 침입탐지 기술은 사용자의 IP 주소, 도메인으로 IP를 확인할 수 있는 기능, 컴퓨터로 송신 및 수신 되는 패킷을 검사 할 수 있고 패킷의 크기도 확인할 수 있다. 그리고 TCP, UDP 등의 정보를 검사할 수 있으며 출력된 정보를 저장할 수 있게 하였다. 이것을 통해서 사용자들은 자신의 IP로 송, 수신 되는 패킷과 기타 정보를 알 수 있으며 비정상적으로 큰 패킷들이 수신 될 경우 서비스거부공격 등 공격에 대비할 수 있다. 경량화 침입탐지 기술은 상용 침입탐지시스템과 같이 공격을 능동적으로 방어하는 기술을 적용하지 않지만 소형화, 경량화하여 프로세스에 오버로드를 주지 않는 장점을 가지고 있다.

## 참 고 문 헌

[1] 박상현, 뇌를 자극하는 C# 5.0 프로그래밍, 한빛미디어, 2014
[2] 조호묵, 이귀붕, 김성수, C#으로 배우는 보안 프로그래밍, 가메출판사, 2014
[3] 조호묵, C# 5.0 프로그래밍 실전 프로젝트, 가메출판사, 2015
[4] 제이슨 안드레스, 라이언 린, 해킹과 침투 테스트를 위한 코딩, 비제이퍼블릭, 2012
[5] 히로시 유키, 알기 쉬운 정보보호 개론, 인피니티북스, 2012
[6] 강유, 정수현, 강유의 해킹&보안 노하우, 에이콘, 2003
[7] 크리스 샌더즈, 와이어샤크를 활용한 실전 패킷 분석, 에이콘출판, 2007

## 저 자 소 개

● 박 성 준(Seong-Joon Park)

▪2016년 : 청주대학교 컴퓨터정보공학사
<관심분야> : SW개발, 네트워크보안

● 김 봉 한(Bong-Han Kim)

▪1994년 : 청주대학교 전자계산학과 공학사
▪1996년 : 한남대학교 전자계산공학과 공학석사
▪2000년 : 한남대학교 컴퓨터공학과 공학박사
▪현재 : 청주대학교 컴퓨터정보공학과 교수
<관심분야> : 네트워크보안, 모바일앱