Agent Based Campus Internal Network Intrusion Detection Model

Adedokun, E.A1, Mu’azu, M.B2, Bajoga3, B.G and Dajab, D.D4
1234Department of Electrical and Computer Engineering Ahmadu Bello University Zaria
Email: wale@abu.edu.ng

Abstract

The aim of the research is the modeling and implementation of an intrusion detection system (IDS) with focus on internally generated intrusions on a multi-Disciplinary campus network using Ahmadu Bello University (ABU) Zaria as a case study. A.B.U Zaria campus network is a complex network covering three campuses, 12 faculties and over a hundred Departments and as such there are varied types of users with diverse application requirements. Traffic data for a period of 24 months (January 2011 to December 2012) comprising of local to local and local to remote were used in developing the knowledge base of the IDS. The overhead of each packet derived from properties and composition of each packet is determined and used to form the behavior base of the IDS. The overhead determined is compared to a threshold called the safe score. This is the maximum allowable overhead for an Ethernet frame and has a value of 0.45. The developed model hybridizes the knowledge –based and behavior-based detection techniques and is implemented as an agent based application (using autonomous and adaptic agents or workers) in a program written in C-sharp(C#). If any packet has an overhead greater than the safe score, it is logged for further analysis. In the validation stage, the model processed 3,422,000 packets and 2328 packets were logged. Using Wireshark for the analysis of the logged packets, 298 packets (13% of the logged packets) were determined not to be malicious but of applications and or protocols not in the original knowledge-base and as such the knowledge base was updated. The IDS model developed is capable of updating its knowledge base, can sniff traffic directly on the network and utilizes less than 15.5% of ROM and CPU capacity at peak traffic period. It also showed about 50% improvement in the number of unknown applications and protocols identified by Netflow analyzer upon implementation.

Key words: Campus Internal Network, IDS, Adaptic Agents

I. Introduction

With the increasing growth in the use of computers for teaching, learning, research and even administrative activities on campuses, network function is increasingly becoming powerful. As campus networks keep growing in size and functionality, network security issues have also become major challenges on campus networks for a long time [4]. Like any other network, campus networks have suffered various attacks ranging from unauthorized access from outside the network to attacks by users within the network and also attack by users within the network to remote sites.

A lot of work has been focused on attacks from remote sites into the network or malware attacks in form of viruses, spyware etc. [2,3,5]. Not much work has been done to address internal attacks carried out by legitimate users within the same network. Such attacks prove more difficult to detect and arrest without resulting in disruption of some network services that may be required by legitimate users especially on a multi-disciplinary campus network. Campus networks have proved to be one of the most challenging networks to manage due to diversity of network services required by the various members of campus community, based on their diverse needs and activities. Most times when these unwanted network traffics are noticed on a campus network, it is always a challenge finding simple and quick means to arrest them in such a manner that would not disrupt authorized legitimate network activities and services.

An intrusion detection system (IDS) inspects all inbound and outbound network activity and identifies suspicious patterns that may indicate a network or system attack from someone attempting to break into or compromise a...
system. IDS are mostly designed to identify what might have gotten past the firewall. This work is thus aimed at developing such a system that can detect unwanted network traffic generated within a network capable of degrading the quality of service (QoS) or leading to a complete denial of service on a typical multi-disciplinary campus network.

II. Network Setup for Data Collection

Two Network monitoring systems were installed in the Data Centre of the Ahmadu Bello University Zaria. The two systems were placed on the live network to collect live network data in real time. The two machines had different operating systems to ensure that the data collected for use as the training data for the model will have universal application across major operating systems. One of the servers runs Windows Server 2008 and also has installed and configured on it, PRTG and Netflow Analyser. While the second monitoring system runs a free and open source operating system (Ubuntu Server Version 11) and has installed and configured on it two other collectors, NFSEN and NTOP.

The core router is a Cisco router and has a Cisco switch connected to it as the public switch. On the switch, the Netflow feature is enabled such that it will allow the Netflow collector to analyse probed aggregate flows sent to it.

Sample of Collected Data

Table 1: Traffic Summary from Netflow Analyser

<table>
<thead>
<tr>
<th>Application</th>
<th>Traffic</th>
<th>% of Total Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSPF</td>
<td>936.64MB</td>
<td>94%</td>
</tr>
<tr>
<td>Unknown</td>
<td>43.10MB</td>
<td>4%</td>
</tr>
<tr>
<td>http</td>
<td>377.32KB</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>ICMP</td>
<td>79.11KB</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Ntp</td>
<td>1.36MB</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>https</td>
<td>86.52KB</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>domain</td>
<td>274.31KB</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Smtp</td>
<td>1.1KB</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>dzoglsrver</td>
<td>29.27KB</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Others</td>
<td>&lt;1%</td>
<td></td>
</tr>
</tbody>
</table>

III. Model Development

The hybrid model developed using the behavioural pattern of the network traffic and the knowledge base built up based on the 24 months of data collected is based on an agent-based program written in C# (C-Sharp) programming language. Its strong programming features as such as automatic garbage collection, delegates and events management (necessary for development of agent-based applications) make it suitable for this work. Since the work is essentially concerned about internally generated traffic, the protocol usage on the network was examined. Some traffic properties such as the source IP, destination IP, packet sizes and percentages were considered in the development of the model and determination of the metrics (or constraints in this case).

Two separate constraints are loaded by the model:

i. Constraints for defined (known) traffic;
Agent Based Campus Internal Network Intrusion Detection Model

ii. Constraints for undefined (unknown) traffic

Depending on the properties of a packet, the constraints are calculated based on overhead generated by network frames. It is noted that most parameters that form the properties of a frame (preamble, destination address, source address cyclic redundancy check etc) have standard values, however, the value of the payload of each frame which is also the message carried by each frame differs depending on the size and composition of the message. The minimum and maximum frame sizes were used to calculate the default minimum and maximum overheads. The largest frame consists of 1538 Bytes, as in Table 2 with a 9.6µs inter-frame gap; therefore

Table 2: Total Maximum Frame Physical Size (Source: IEEE 802.3)

<table>
<thead>
<tr>
<th>Frame Part</th>
<th>Maximum Size Frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inter Frame Gap (9.6µs)</td>
<td>12 Bytes</td>
</tr>
<tr>
<td>MAC Preamble (+ SFD)</td>
<td>8 Bytes</td>
</tr>
<tr>
<td>MAC Destination Address</td>
<td>6 Bytes</td>
</tr>
<tr>
<td>MAC Source Address</td>
<td>6 Bytes</td>
</tr>
<tr>
<td>MAC Type (or Length)</td>
<td>2 Bytes</td>
</tr>
<tr>
<td>Payload (Network PDU)</td>
<td>1500 Bytes</td>
</tr>
<tr>
<td>Check Sequence (CRC)</td>
<td>4 Bytes</td>
</tr>
<tr>
<td><strong>Total Frame Physical Size</strong></td>
<td><strong>1538 Bytes</strong></td>
</tr>
</tbody>
</table>

Overhead = \( \frac{\text{Frame Size} \, - \, \text{Payload Size}}{\text{Frame Size}} \)

Thus, for the largest size of Ethernet frame:

Overhead = \( \frac{1538 \, - \, 1500}{1538} \)

= 0.0247

Unknown packets are initially treated as suspicious at this stage. Therefore the constraints for unknown traffic will take into consideration the minimum frame size allowable on Ethernet. The minimum frame payload is 46 Bytes (dictated by the slot time of the Ethernet LAN architecture). The maximum frame rate is achieved by a single transmitting node. This implies a frame consisting of 72 Bytes (as in Table 3) with a 9.6µs inter-frame gap (corresponding to 12 Bytes at 10Mbps). The total utilized period (measured in bits) corresponds to 84 Bytes.

Table 3: Total Minimum Frame Physical Size (Source: IEEE 802.3)

<table>
<thead>
<tr>
<th>Frame Part</th>
<th>Minimum Size Frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inter Frame Gap (9.6µs)</td>
<td>12 Bytes</td>
</tr>
<tr>
<td>MAC Preamble (+ SFD)</td>
<td>8 Bytes</td>
</tr>
<tr>
<td>MAC Destination Address</td>
<td>6 Bytes</td>
</tr>
<tr>
<td>MAC Source Address</td>
<td>6 Bytes</td>
</tr>
<tr>
<td>MAC Type (or Length)</td>
<td>2 Bytes</td>
</tr>
<tr>
<td>Payload (Network PDU)</td>
<td>46 Bytes</td>
</tr>
<tr>
<td>Check Sequence (CRC)</td>
<td>4 Bytes</td>
</tr>
<tr>
<td><strong>Total Frame Physical Size</strong></td>
<td><strong>84 Bytes</strong></td>
</tr>
</tbody>
</table>
Overhead = \frac{\text{Frame Size} - \text{Payload Size}}{\text{Frame Size}}

Thus, for the smallest size of Ethernet frame

Overhead = \frac{84 - 46}{84} = 0.4523

From the analysis carried out in determining the default minimum and maximum frame sizes, the maximum overhead allowed on the Ethernet frame is 0.45. As such any frame with score above the maximum overhead will be logged for further analysis.

The flowchart for the model is as shown in Figure:

Figure 1: IDS Flow Chart
Agent Based Campus Internal Network Intrusion Detection Model

The program functions as follows:

i. When a data packet is received by the model from a Netflow enabled port on the network, the agent (also called worker) first checks the packet’s header information for the source and destination IP to determine the origin and destination of the packet and whether the packet is a local packet i.e. whether it is internally-generated and internally-bound traffic or not. Confirmed local packets are sent to the next stage of the programme while confirmed non-local packets are ignored (the model treats only internally generated traffic alone) and allowed to follow the normal route without being processed by the model.

```csharp
private void Packet_Processor(Packet _inputPacket)
{
    if (!this.Running) return;
    var ipv4Dgram = _inputPacket.Ethernet.IpV4;
    if (transportProps == null) return;
    var sourceIp = ipv4Dgram.Source;
    var destinationIp = ipv4Dgram.Destination;
    var sourcePort = transportProps.SourcePort;
    var destinationPort = transportProps.DestinationPort;
    var size = ipv4Dgram.Length;
    string transport = "";
    if (this.Running != true) return;
}
```

ii. Then the program checks the protocol information of verified local traffic to determine whether the packet is carrying data of a defined (known) protocol and/or application (UDP, TCP etc). At this stage, packets are divided into two depending on whether their properties are on the knowledge base or not.

```csharp
if ((transportProps as PcapDotNet.Packets.Transport.TcpDatagram) != null)
{
    transport = "TCP";
}
{
    transport = "UDP";
}
else
{
    Log.Success("Something else");
    while (true) ;
}
```

```csharp
if (transport == "TCP")
{
    try
```
private const int KNOWN_PACKET_SIZE = 1500;
private const int UNKNOWN_PACKET_SIZE = 84;
...
...
private void LoadFunctionObjects()
{
    _UnknownPacketClassifier = new Func<Packet, bool>((p) =>
    {
        if (packetScore > this.getSafeValue()) return true;
        return false;
    });

    _KnownPacketClassifier = new Func<Packet, bool>((p) =>
    {
        // Further implementation...
    });
}

iv. After the constraints are loaded by the program, the relevant properties of a packet are extracted and used to calculate its risk score by the worker. The risk score used by the program is the maximum overhead calculated with default ISO Ethernet standard values of maximum and minimum allowable frames sizes. A unified scale is used for packets containing ‘known’ data and ‘unknown’ data (though from (iii), it was stated clearly that different constraint sets were used for each type). This is to reduce the possibility of false positive bias in the program.

protected override void WorkerMethod()
{
    try
    {
        result = ListenSocket.Receive(buffer, SocketFlags.DontWait);
    }
    catch (Exception e)
    {
        continue;
    }

    if (result > 0)
    {
        Packet _thisPacket = new Packet(buffer, DateTime.Now, DataLinkKind.Ethernet);
        if (!_thisPacket.IsValid) continue;

        if (QualifierFunction(_thisPacket))
        {
            try
            {
                AffirmativeSocket.Send(_thisPacket.Buffer);
            }
            catch (Exception positiveSocketException)
            {
                continue;
            }
        }
    }

v. The calculated overhead is compared against the risk score which is a given threshold defined on the unified scale. If the overhead exceeds the threshold (risk score), the packet is suspected to be malicious and its content and details are subjected to further analysis. However, if the packet’s overhead is lower than the threshold, then the packet details and content are allowed into the network without further analysis. The failed packets are further examined with a view of achieving the following:

a. Identify malicious or intrusive traffic and logging them for determination of the best preventive actions.

_loggerTask = new Action<Packet>((p) =>
Agent Based Campus Internal Network Intrusion Detection Model

```csharp
  _logInstance.Error("Suspicious packet detected");
```

b. Provide a feedback to the database of known traffic with the aim of updating the database with traffic that was termed unknown but found not to be intrusive.

Figure 2: Screenshot of the IDS Interface

IV. Model Testing and Validation

The model was validated on the Ahmadu Bello University network, severally with a two hour test period each time and eventually for a period of sixty days (two months). The model was installed on a high end laptop and connected to the core switch at the University Data center. It was installed on a Netflow enabled port to capture and analyse network traffic on the live network.

During the validation period, the model logged about suspicious 2328 packets out of the 3,422,000 packets processed by the model. Using Wireshark for the analysis of the logged packets, 298 packets (13% of the logged packets) were found not to be malicious but contain applications and or protocols that were not in the original knowledge-base of the model, hence, the knowledge base was updated.
V. Result Comparison

During the period of testing the model, and knowledge base update, the output of sniffers on the network was compared to the initial output that was used as training data for the model. It was discovered that the percentage of unknown traffic detected was consistently reducing as the knowledge base of model was updated and consequently the core router.

Table 4: Traffic Summary from Netflow Analyser during Model Validation Period

<table>
<thead>
<tr>
<th>Application</th>
<th>Traffic</th>
<th>% of Total Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSPF</td>
<td>1.29 GB</td>
<td>95%</td>
</tr>
<tr>
<td>Unknown</td>
<td>28.39 MB</td>
<td>2%</td>
</tr>
<tr>
<td>http</td>
<td>21.12 MB</td>
<td>2%</td>
</tr>
<tr>
<td>ICMP</td>
<td>5.74 MB</td>
<td>1%</td>
</tr>
<tr>
<td>Ntp</td>
<td>3.33 MB</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>https</td>
<td>2.01 MB</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Microsoft-ds</td>
<td>1.48 MB</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Snmp</td>
<td>2.31 MB</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>dzoglserver</td>
<td>29.27 KB</td>
<td>&lt;1%</td>
</tr>
<tr>
<td>Others</td>
<td>&lt;1%</td>
<td></td>
</tr>
</tbody>
</table>

VI. Conclusion:

The IDS model developed employed the use 24 months data as its knowledge base and calculated overhead of packets to determine the behavioral pattern of each packet. The model is capable of updating its knowledge base, can sniff traffic directly on the network and utilizes less than 17% of ROM and CPU of the host system capacity at peak traffic period. It also showed about 50% improvement in the number of unknown applications and protocols identified by Netflow analyzer upon implementation.

REFERENCE