Network Attack Detecting Based On Net-Flow

You Wu **Ruoyu Wang**

ywu80@hawk.iit.edu rwang33@hawk.iit.edu

I. Abstract

Today, the security of the Internet have become a critical issue. Since, we did almost activities in the network especially for some financial and secure task which always lead to some very serious consequences if the Internet we depend on were under attack. Hence, it counts a lot that we need to have a secure network to allow us to conduct our activities, otherwise we need to detect the attack and fix it. So, in this paper we are going to introduce a way to determine whether the pc or server (we call it as Aim-Node below) is under attack or not by monitoring the Net-Flow (the flows that come out from the Aim-Node) with the methodologies of machine learning and mathematics. Finally, we learn that if there is a discrepancy of the Net-Flow form the Aim-Node compared with the history information, we consider it is under attack.

II. Introduction

Managing enterprise networks is a huge responsibility and with organizational environment, network administrators are now tasked with deploying advanced network services, maintaining network performance, and reducing costs with fewer resources. With these new challenges, administrators are facing tremendous pressure to maintain network uptime and prevent any organization-wide operational loss due to network problems. One of the biggest factors impacting the network performance is network traffic and bandwidth usage. With more personal devices hogging enterprise network bandwidth, network managers are deploying new policies to maintain the quality of service. Although there are multiple ways to manage your network, flow-based network monitoring is the most sought after approach to managing networks today.

And what we want to do is to implement a very low resource cost way to detect the attacks which will affect with the Net-Flow of the Aim-Node and can restore the history data information in order to let the improvement of the performance of the Aim-Node could be easier to do since we can summarize the defects of the network which the Aim-Node based on from the history data.

So, we are going to create an application in order to observe the Net-Flow and then set up a database to record the Net-Flow from the server or computer that we want to observe. After that, we use the methodologies of machine learning to be aware of the habits of the aim node. Then we use some mathematical methods to figure out whether there is an attack or not.

Concepts which defined and discussed about below:

Aim-Node: The Aim-Node is the pc or server we are going to observe.

Detector: The server to monitor the Aim-Node.

Net-Flow: The Net-Flow is a data to describe the packages were sent out from the Aim-Node.

A. Overview the Data Obtain Design

Our design to implement the application discloses an active detection-based host internet protocol (IP) flow estimation method and relates to the technical field of host IP flow statistics. The method comprises the following steps of: initiating ICMP communication to a target host on an optional host; calculating an added value INC of the target host returned to IP-ID fields of an IP packet of the optional host in a time interval T on the optional host; and dividing the added value INC by T to obtain the IP flow of the target host. The method can instantly obtain IP flow statistical information of an optional remote target host on an internet, and tough requirements of a traditional method on environment and access authority are removed during measurement; therefore, massive detection resources are saved.

It has been proved that it is extremely difficult to set a flow listener in the computer from wide places and obtain the flow-data to analysis the net attack. In this paper we provide a new method to get the flow from the Aim-Node and we even do not touch the packages which are sent to any possible destinations.

Figure 1. Connection between Aim-Node and Remote Detector

B. Overview the Data Analysis Design

Since obtain the Net-Flow data of the Aim-Node, we build a database to record the all the data of every interval and practice the detector to learn the habit of the Aim-Node which is supposed to compare it with the real-time Net-Flow and analysis in order to make a decision whether the Aim-Node is under attack or not.

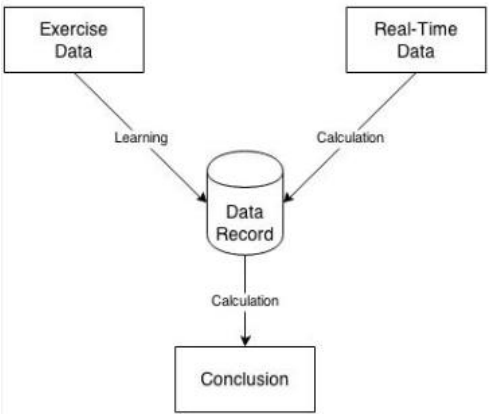


Figure 2. Analysis Model

III. Background

A. The way to detect network attack

There are thousands of million network detector tools around the world and there two type of styles of this kind of applications. One is to analysis the packages which means we have to analysis the information that the package carrying based on some rules, so this method will lead to a lot of resource to analysis every package which is very complicated. On the other hand, the second way to detect the attack is based on the Net-Flow, monitoring the Net-Flow of the Aim-Node will be easy to implement since if the attack has been lunched the Net-Flow always will be obvious unusual, but this method will fail to detect the attack which will Not lead to any Net-Flow issues. Our design is based on the second method.

Because it is easy to set an application in the Aim-Node and check every package that were in and out and still can have the data of the package so that it will contribute to an accurate Net-Flow. But our plan is to monitor the Aim-Node without touching the package and no matter how we do it, we need some development libraries. In this paper we use the Winpcap to operate the network card which is the core of the application.

WinPcap is the industry-standard tool for link-layer network access in Windows environments: it allows applications to capture and transmit network packets bypassing the protocol stack, and has additional useful features, including kernel-level packet filtering, a network statistics engine and support for remote packet capture.it consists of a driver, that extends the operating system to provide low-level network access, and a library that is used to easily access the low-level network layers. This library also contains the Windows version of the well-known libpcap Unix API. WinPcap is the packet capture and filtering engine of many open source and commercial network tools, including protocol analyzers, network monitors, network intrusion detection systems, sniffers, traffic generators and network testers. Some of these networking tools, like Wireshark, Nmap, Snort, ntop are known and used throughout the networking community.

B. Mathematic way to analyze

In statistics and probability theory, the **standard deviation** (**SD**) (represented by the Greek letter sigma, **σ**) shows how much variation or dispersion from the average exists. A low standard deviation indicates that the data points tend to be very close to the mean (also called expected value); a high standard deviation indicates that the data points are spread out over a large range of values.

The standard deviation of a random variable, statistical population, data set, or probability distribution is the square root of its variance. It is algebraically simpler though in practice less robust than the deviation. A useful property of the standard deviation is that, unlike the variance, it is expressed in the same units as the data. Note, however, that for measurements with percentage as the unit, the standard deviation will have percentage points as the unit.

In addition to expressing the variability of a population, the standard deviation is commonly used to measure confidence in statistical conclusions. For example, the margin of error in polling data is determined by calculating the expected standard deviation in the results if the same poll were to be conducted multiple times. The reported margin of error is typically about twice the standard deviation—the half-width of a 95 percent confidence interval. In science, researchers commonly report the standard deviation of experimental data, and only effects that fall much farther than one standard deviation away from what would have been expected are considered statistically significant—normal random error or variation in the measurements is in this way distinguished from causal variation. The standard deviation is also important in finance, where the standard deviation on the rate of return on an investment is a measure of the volatility of the investment.

C. Machine learning makes the judge

Machine learning is a branch of artificial intelligence, concerns the construction and study of systems that can learn from data. For example, a machine learning system could be trained on email messages to learn to distinguish between spam and non-spam messages. After learning, it can then be used to classify new email messages into spam and non-spam folders.

The core of machine learning deals with representation and generalization. Representation of data instances and functions evaluated on these instances are part of all machine learning systems. Generalization is the property that the system will perform well on unseen data instances; the conditions under which this can be guaranteed are a key object of study in the subfield of computational learning theory.

There are a wide variety of machine learning tasks and successful applications. Optical character recognition, in which printed characters are recognized automatically based on previous examples, is a classic example of machine learning. So, we use the machine learning as an ideal solution to solve the data.

IV. Approach:

In this section, we will introduce our approaches

(1) Obtaining the Net-Flow information.
(2) Data utilization and analysis.

A. Application Approach

(1) Previous Design

The main idea of our first implementation is that from checking the amounts packets of an Aim-Node during a time interval, our application will determine whether this Aim-Node is suffering an attack. And to count the packets flow, we utilize the property of the sequence number of packets. As we know, if an Aim-Node receive a network request, no matter it accepts or rejects, it will send packets back to the request sender. Meanwhile, the sequence number is keep increasing from the system starts up. So using this property, we can get the real-time packets flow of an Aim-Node. Below are the detail implementation of our experiments.

We chose C++ as our develop language because of the auxiliary tools using in the project. Here we identify the run-time sequence of our first design

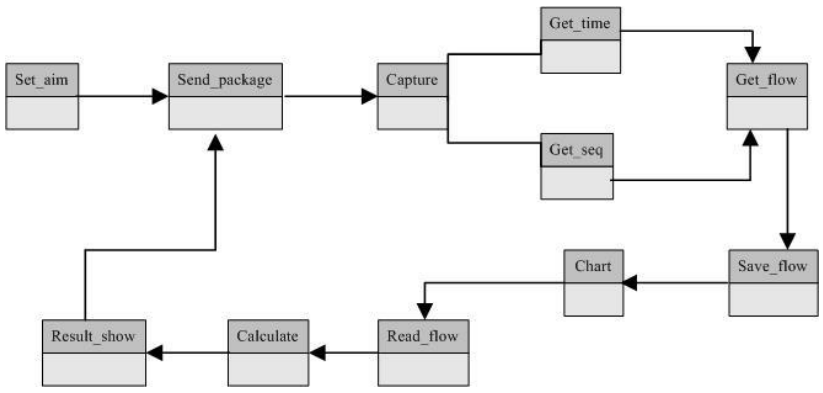


Figure 3. Class Diagram

But unfortunately we use this model does not work in almost cases, because the network in real world are always based on routers or gateway, so that we can`t just get the reply IP_ID as we want, actually they just send back the ICMP package with the same IP_ID which the detector sent to the Aim-Node.

(2) Final Design

Not like the most methods of flow detection, instead of setting listener on the aim node, we set a remote server as a listener to detect the aim nodes. In addition, the Aim-Node will run a very low resource cost application to send an ICMP package interval (the interval depends on the user).

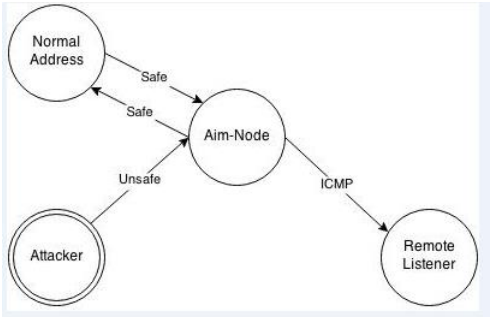


Figure 4. Platform Architecture

Here is the Extensive Finite State Machine.

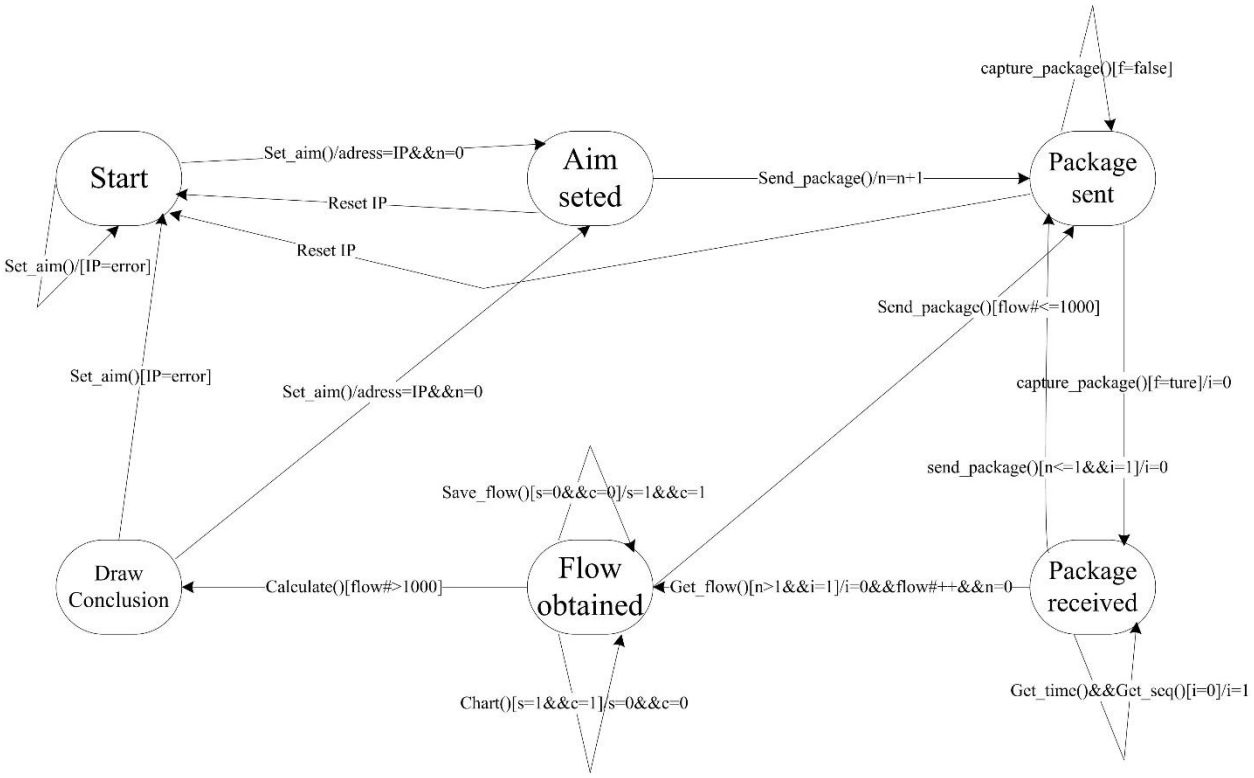


Figure 5. EFSM Diagram

(3) Algorithm

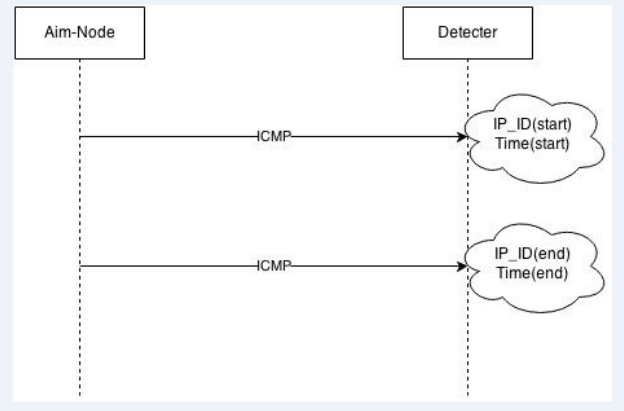


Figure 6. Algorithm for calculating Net-Flow

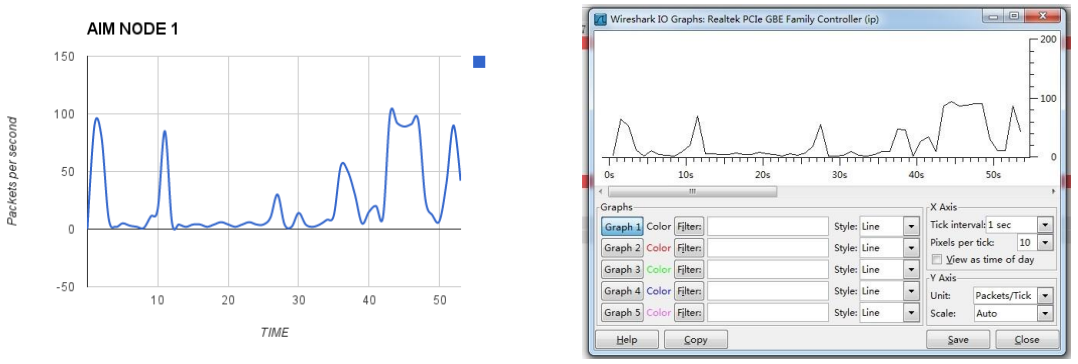
$$NetFlow_{out} = \frac{(IP\_ID_{end} - IP_{ID_{start}}) - Count}{T_{end} - T_{start}}$$

IP_ID is the sequence number of the real-time ICMP package. And the ***T*** is the time when we get the seq.

Count is how many ICMP package we have sent to the Aim-Node.

(4) Correctness Validation

In order to make sure that the Net-Flow we get is correct, we use the wire shark to compare the Net-Flow of the Aim-Node with our application. As shown in Figure 7, (a) shows the result we get from our own application, while (b) shows the Net-Flow captured by WireShark. And in the 60 seconds duration, the Net-Flow is unequivocal correct by our application. So in the application part, we did what we actually want to implement and the result turn to be right.



(a) (b)

Figure 7. Accuracy Ensurance

B. Data Analysis

In this section, we will introduce our data analysis from three critical parts: experiment environment set-up, data storage and analysis algorithm.

(1) Experiment Environment Set-up

As we said above, our design is that the remote detector sends an ICMP message to an Aim-Node. After getting the Aim-Node replies, we can calculate the Net-Flow of the Aim-Node from the information inside the ICMP message. However, this ideal architecture won't always work in the real world. For example, for the Aim-Node connected in an internal network, the gateway will sent back the message without editing the identification in the header. So that we cannot obtain the net flow of the Aim-Node.

So to overcome the obstacle, we come up with an idea. If the Aim-Node is blocked by the gateway and cannot be got through by the remote detector, why not let it report its status by itself. Based on this, we generate a new design. We just set a tiny application on the Aim-Node, which is low cost (approximate 1MB memory cost and about no network cost). The app on the Aim-Node initiatively sends an ICMP message to the remote detector every minute. Thus we can be aware of the Net-Flow on the Aim-Node.

Finally, we set up a small internal network architecture. As shown in Figure 7, we connect Aim-Node, remote detector and attacker into one internal network.

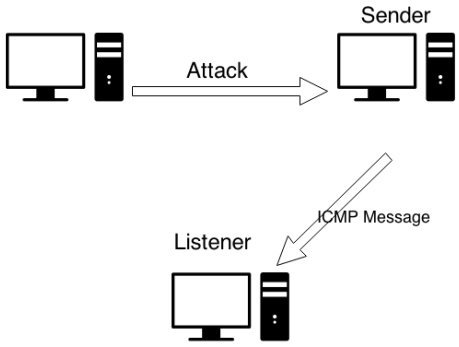
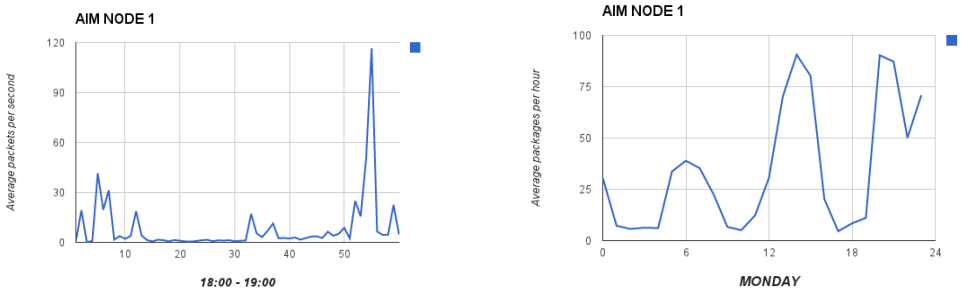


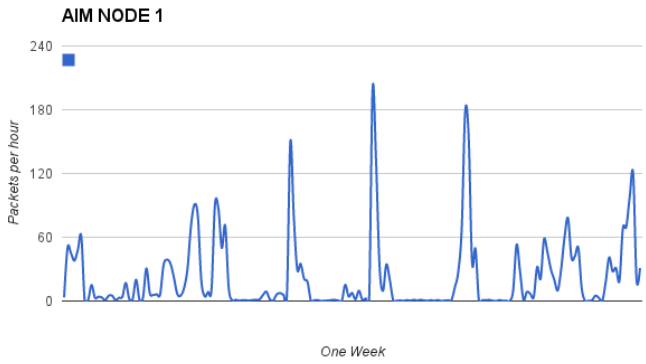
Figure 8. Platform Architecture

(2) Data Utilization

After the correctness of our design and implementation has been verified, the real data is started to be collected.



(a) (b)



(c)

Figure 9. Test Cases

As Figure 9 shown above, the data is collected based on various time duration. In Figure 9(a), the horizontal axis is one hour, while the time interval is one minute, since the tiny application set on the Aim-Node which is called sender is set to send an ICMP message per minute in our experimentation. Then the vertical axis is package number per minute. After that, we extend the time duration from one hour to one day (shown in Figure 9(b)) and one week (shown in Figure 9(c)).

The way we collect the data is 24/7. In our architecture, the detector observes the Aim-Node uninterruptedly. And these data will turn into history information of the Aim-Node, with which we can learn about the working habits of the Aim-Node. So that after recording these data in the database, we can detect attacks with comparing the real-time Net-Flow with the history information stored in the database. The more abundant history information we recorded, the higher accuracy we will get to detect attacks.

(3) Algorithm

Here we choose to use standard score to represent the possibility of network attack.

$$z = \frac{x - \mu}{\sigma}$$

where:

x is the real-time package number;

μ is the mean of the package number before z;

σ is the standard deviation of the package number before z.

The absolute value of z represents the distance between the real-time package number and the mean in units of the standard deviation. Z is negative when the real-time package number is below the mean, positive when above.

To generate the μ and σ, we use the records in the database.

$$\mu = \frac{x_n + x_{n+1} + \cdots + x_{before\ current}}{t}$$

$$\sigma = \sqrt[2]{\frac{(x_n - \mu)^2 + (x_{n+1} - \mu)^2 + \cdots + (x_{before\ current} - \mu)^2}{total\ number\ of\ records}}$$

V. Experiments

A. Attack Methods And Tools

Here we use to two main methods to generate the result of the Aim-Node is under attack. First one is injecting real attack with DOS simulators.

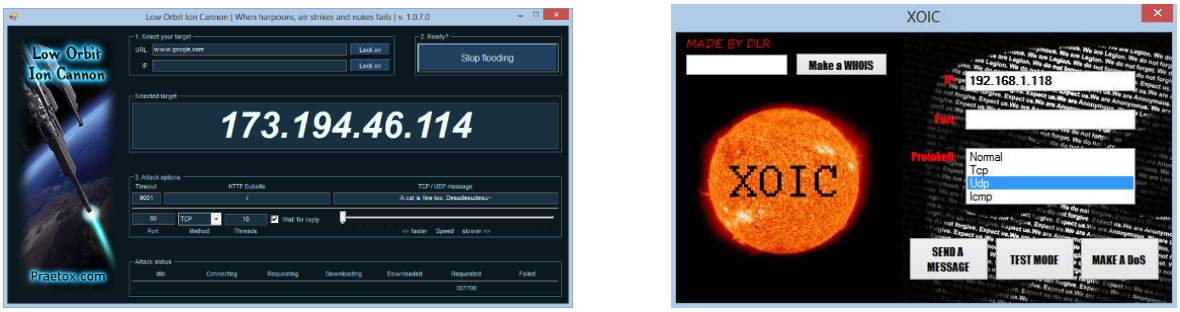


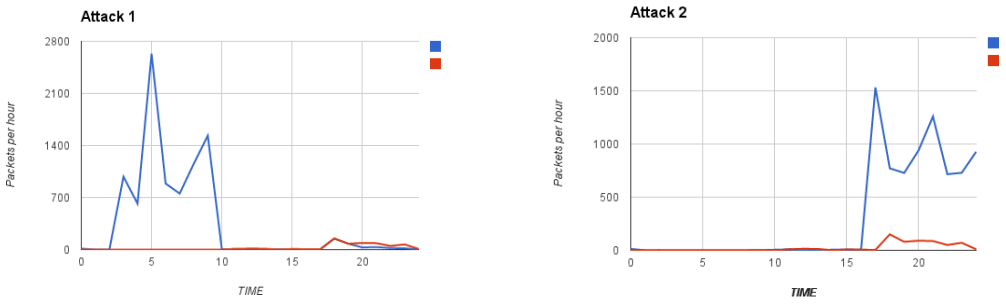
Figure 10. DOS attack simulators

As shown in Figure 10, we choose two most popular DOS simulators to launch real attack to our Aim-Node. They are very easy to use without losing efficiency.

The second method is launching safe application with heavy network load (P2P etc.) to simulate a DOS attack.

B. Experiment Results

The results are shown below.



(a) (b)

Figure 11. Attack Results

Figure 11(a) shows the results of launching attack at the time duration of low network traffic. While Figure 11(b) shows the results of launching attack at the time duration of high network traffic. The horizontal axis is time while the vertical axis is packets number per hour. The red lines represent the history information of the Aim-Node, while the blue lines represent the attack we inject in. We can see that with our design and implementation, the attack can be detected correctly.

Here we have launched 20 different attack. All of them have been detected correctly. This demonstrates our design and implementation are pretty good.

VI. Discussion

A. Contributions

In this paper, there are three critical contributions we have made listed below.

(1) As we said above, if the Aim-Node blocked in a gateway or router, we cannot get its Net-Flow information through the blocking. After we re-design our application architecture, we set a tiny sender on the Aim-Node. This tiny application only occupies approximate 1MB memory, which is invisible for the Aim-Node. Meanwhile, the sender is set to only send one ICMP message to the detector every minute, which leads to no consequences on the network performance of the Aim-Node.

(2) For detecting the attacks, we set up a database to record the Net-Flow of the Aim-Node as history information. As we all know, for improving the performance and consummating the architecture, the abundant history information is an indispensable aspect. So the information recorded in the database not only can be used to detect the attack currently, but also can be used in the future to do further analysis.

(3) As we said above, in our design we set one hour as the minimum unit of observation. So in our database, there are only history information of each different hour. Thus, the minimum detecting time is one hour, which means we can detect an attack after it has been launching for one hour. Here come a problem. Some kinds of Aim-Node is time insensitive, which means they cannot tolerate a DOS attack for one hour, such as financial servers.

What can we do to solve the problem? Don't worry. Here is the solution.

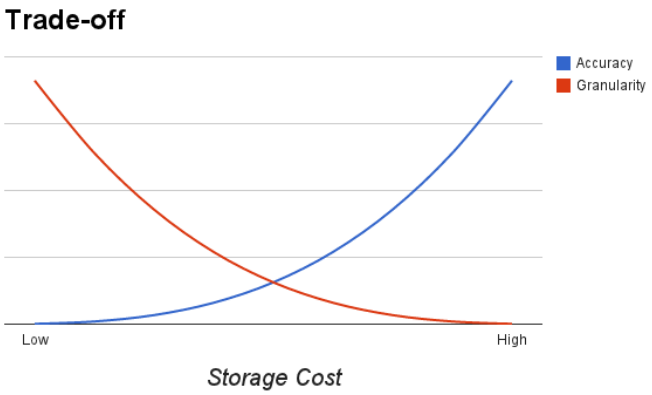


Figure 12. Trade-off

As Figure 12 shown above, to meet the different requirement of various Aim-Nodes, the minimum time unit can be reset. As we see, the accuracy represents the minimum time the detector can detect an attack, which is relative to the minimum time unit. And the granularity represents the minimum time unit. For example, if the Aim-Node need the detector to detect an attack within one minute, the minimum time unit should be set to one minute. Meanwhile, the database should record the history information for each different minute, which is heavy storage cost. So this figure shows that the price of high accuracy is storage cost. For different Aim-Node, parameters can be set differently.

B. Limitations

(1) In some situations, some safe applications with heavy network load will lead to high Net-Flow, which will be detected as an attack. This will be considered as a defect of our design and implementation.

(2) Not like the DDOS attack, some other kinds of attacks, which won't lead to heavy Net-Flow for the Aim-Node, cannot be detected by our design.

(3) Various operating system, which run on the Aim-Node, may have different policy to identify the IP_ID. Hence, we cannot calculate the Net-Flow accurately.

VII. Conclusion And Future Work

In our project, we introduce a novel method to detect network attack, which based on the Net-Flow. Not like the prior work, the information inside the message is not taken into consideration. However, we only counts the packages number during a time interval, which is low resource cost for the Aim-Node. In our design, a tiny sender is set on the Aim-Node to send ICMP message initiatively to the remote detector. And as shown above, our design and implementation are quite efficient with some limitations.

For the future work, we want to test our design on some huge servers to ensure that our application can be widely used in the real world. As our experiments described above, about one thousand packets per second can lead to network paralysis in our small internal network. However, such level of Net-Flow will not be treated as an attack for servers, since the Net-Flow for them can reach hundreds of thousands packets per second which is far higher than it in our internal network.
Furthermore, to overcome the mis-detection, such as P2P connection, we need to access the process table to differentiate the safe applications and the attacks. Hence, our design and implementation will be more strong and accurate.