A Scheme For Fast Network Traffic Anomaly Detection

Xiaofen Wang, Liaojun Pang†, Qingqi Pei, Xuejun Li
Mini. of Edu. Key Lab. of Comp. Netw. & Info. Secu., Xidian Univ., Xi'an 710071, China
E-mail: xiaofen-wang85@163.com

Abstract — **Fast network traffic anomaly detection is an important issue in network security. At present, the Internet environment becomes more and more complex than ever before. The unexpected anomaly occurs constantly, which brings great loss to Internet services. This makes people put forward higher requirements for the speed and real-time of network traffic anomaly detection. How to quickly perceive the traffic anomaly is a new demand in the network anomaly detection. Due to the shortage of the conventional detection methods in terms of speed, a fast anomaly detection scheme is proposed in this paper, which is based on the iterative estimation of Hurst parameter. The anomaly can be determined through the value of estimation. Meanwhile, by on-line we can estimate continuously and refresh the value of Hurst according to the latest network traffic, and then we can get the latest traffic data and speed up the process of anomaly detection. Analyses show that this scheme is effective in fast anomaly detection.**

Keywords: Anomaly detection, Fast, Hurst, Iteration

I. INTRODUCTION

The rapid development of the Internet brings a large number of network security issues, and network traffic anomaly detection is an important aspect in the network security. Conventional network traffic anomaly detection methods are mostly based on features matching, and such methods must be pre-established feature library, which should be trained and learned[4][5]. These methods can be broadly divided into the off-line mode and the on-line mode. The off-line mode means that both the data collection and the feature library generation will be achieved before test. This method is weak in flexibility, which is obvious to conclude. Network conditions or even changes in the system would give rise to all states reset. However, through the on-line mode we can overcome the disadvantages of the off-line mode. We can update the feature library continually using the on-line learning, thus the flexibility and the accuracy have been improved[8][9].Largely what we want to point is that both of these modes require prior knowledge of the network distribution and establishment of feature library first, so their self-adaptability and the ability to cope with unexpected situations are somewhat poor. And such a feature library is established by subjective, thus they cannot reflect the characteristics of the network itself. Therefore, it is still difficult to ensure its accuracy. The focus of all the approaches above is the accuracy of the anomaly detection. However, with the development of the Internet technology, the environment of the Internet becomes more and more complex than ever before, and various sudden anomalies and attacks often make detection system unprepared. The fast and real-time of the anomaly detection becomes more and more important, which makes us need a more rapid and accurate detection method to deal with new problems emerging.

Bellcore's monitoring of traffic packets on the LAN (Local Area Network) as well as many research institutions' analysis of the Internet traffic data on the WAN (Wide Area Network)[1][2] shows that exactly or asymptotical self-similarity model is more suitable for us to describe "burst arrival traffic", and this feature is generally characterized with the Hurst parameter. The traffic load changes will affect the "self-similar" feature[2], and thus affect the value of Hurst. In this paper, the iterative method[6] is used to estimate the Hurst parameter, and a scheme for fast network traffic anomaly detection is proposed based on it. According to analyses, the scheme is proved rapid and real-time. We can use it as a method of the on-line anomaly detection.

This paper is organized as follows. In Section II, some definitions and theories about the "self-similar" feature are given; In Section III, basic idea of our program is described; In Section IV, the performance of our scheme is analyzed, and its advantage is illustrated.

II. SELF-SIMILAR STOCHASTIC PROCESS

The real network traffic has the feature of the statistical self-similar. A self-similar process is a stochastic process with scale invariance in the statistical sense. Let $X=\{X_j : j=1,2,\cdots\}$ be a covariance stationary stochastic process, that is to say X is a process with constant mean $\mu=E[X_t]$, finite variance $\sigma^2=E[(X_t-\mu)^2]$, and its autocorrelation function $r(k)=E[(X_t-\mu)(X_{t+k}-\mu)]/\sigma^2 (k=0,1,2,\cdots)$ that depends only on k. We suppose X has an autocorrelation function of the following form:

$$r(k) \sim k^{-\beta} L_1(k), \text{ when } k \to \infty \qquad (1)$$

† Corresponding author.
Email address: ljpang@mail.xidian.edu.cn (Liaojun PANG)

Where $0<\beta<1$, L_1 meets that for $\forall x>0$, We all have $\lim\limits_{t\to\infty} L_1(tx)/L_1(t)=1$.

Let $X_k^{(m)}=(X_{km-m+1)}+\cdots+X)/m(k=1,2,3\cdots)$ denote $\{X_m\}$'s m-order smoothing process, and let $r^{(m)}(m=1,2,3\cdots)$ denote the aggregated time series $X^{(m)}=(X_1^{(m)},X_2^{(m)},\cdots)$'s autocorrelation function.

Definition 1. A process X with the self-similarity parameter $H=1-\beta/2$, if its m-order smoothing process $X^{(m)}$ has the same correlation structure as X , i,e., $r^{(m)}(k)=r(k)$ for all $(m=1,2,\cdots,k=1,2,\cdots)$, then the process X is called exactly second-order self-similar process.

Definition 2. A process X with the self-similarity parameter $H=1-\beta/2$, if its m-order smoothing process $X^{(m)}$ has an autocorrelation function of the following form:

$$r^{(m)}(1)\to 2^{1-\beta}-1, \text{when } m\to\infty \quad (2)$$

$$r^{(m)}(k)\to\frac{1}{2}\delta^2(k^{2-\beta}), \text{when } m\to\infty \ (k=2,3,\cdots) \quad (3)$$

In the function above, $\delta^2(f)$ indicates the second difference operator acting on f , that is:

$$\delta^2(f(k))=f(k+1)-2f(k)+f(k-1) \quad (4)$$

Then the process X is called asymptotically second-order self-similar process.

Definition 3. Wide-sense stationary self-similar stochastic process obeys [11]:

$$\rho_k=H(2H-1)k^{2H-2}, k\to\infty \quad (5)$$

Here H indicates the Hurst parameter with $0.5<H<1$, and the value of H increases as the degree of self-similarity increases. Because $\sum_k \rho_k=\infty$, it is called the long-range dependence. In other words, even if k is large enough, sequence still has great correlation.

By formula (5), we get the following iterative function of H [6].

$$H_{i+1}=\sqrt{(\rho_k k^{2-2H_i}+H_i)\times 0.5}, \ k\to\infty \quad (6)$$

In a large number of the existing self-similar stochastic models, by comparison with real traffic data, we consider that only FBM (fractional Brownian motion) and fractional ARIMA (Auto-Regressive and Moving Average) process are suitable for modeling burst network

III. BASIC IDEA OF OUR SCHEME

We try to design a fast method to detect the network traffic anomaly, and this method must reflect accurately the characteristics of the network itself, so we start from the fractal characteristics presented by network traffic, and make use of Hurst parameter which is used to describe the fractal characteristics to determine occurrence of anomaly.

The flow chart of the method can be shown in Figure 1:

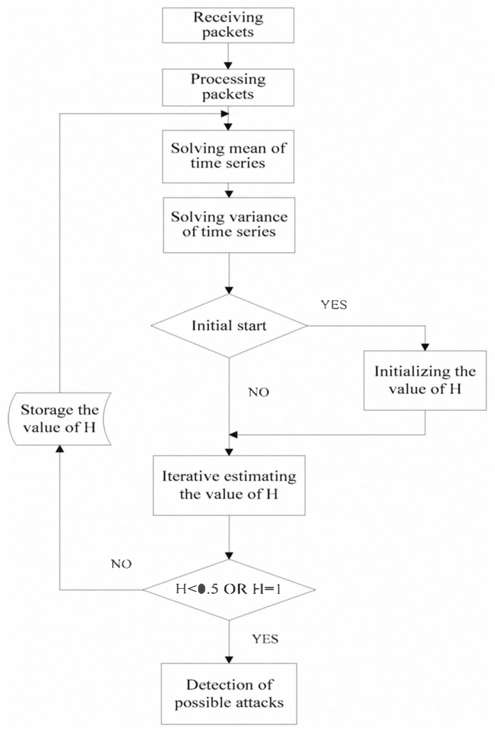
Receiving packets
Processing packets
Solving mean of time series
Solving variance of time series
Initial start
YES
Initializing the value of H
NO
Storage the value of H
Iterative estimating the value of H
NO
H<0.5 OR H=1
YES
Detection of possible attacks

Figure 1. The general framework of our scheme

Now we will describe a few key steps of their principle and ideas of implementation in details.

A. Receiving Traffic Packets

First we must capture network packets, and here we use Winpcap(windows packet capture) library to achieve this. Winpcap works on Windows platform, and we use it to capture network original packets, collect and count network traffic information. The aim of this method is achieved in three steps. 1) get the NIC (Network Interface Card) list; 2) open the Device List, and set accordingly; 3) capture packets and save. The main algorithms involved are shown as follows:

```
  Capture(CardName)
  {
   //get the NIC list
     pcap_findalldevs();
   //open the NIC, and set to promiscuous mode
     pcap_open_live();
   //establish file for the storage of the captured network packets
  pcap_dump_open();
 //read the packets information from NIC or files continuously
  while(re=pcap_next_ex()>=0)
  {
 // store the captured packets into file
 pcap_dump();
 }
 }
```

Here, CardName is a NIC name the user chooses to capture network packets. After a period of time since starting a thread, it can be seen that packets have been successfully captured, and stored into file.

B. Processing Traffic Packets

The captured packets can not be directly used, and we have to deal with the packets, and extract some useful information from these packets. A complete packet has the most primitive data information like the IP header, TCP header. According to requirements of this scheme, we only need to extract the packet arrival time and length information, and store the information in the form of linked list. In order to estimate the Hurst parameter, we must do time-series division to the information, which requires us divide the traffic data by equal time interval, and each interval is recorded as a time slot, the length of packets received within a time slot is recorded as a sample value of this time slot.

The packet arrival time and length information we extract from the original packets can be stored in the form of structure, specifically defined as follows:

```
struct DATA{
            unsigned int Data_Long;
            time_t Arrive_time;
            };
```

The algorithm for time-series division is shown as follows:

```
 void Sequence(CPtrList list,int* counter)
  {
  // get the first place of the linked list
  POSITION pos = list.GetHeadPosition();
  // Traverse the linked list
  while(pos)
     {
      DATA* pNode;
     //get a node in the linked list
      pNode = (DATA*)list.GetNext(pos);
     //get the arrival time information
      time_t time = pNode->Arrive_time;
     //determine the time interval.
      double temp = difftime(time,BeginTime);
      int c = temp/STEP;
      counter[c]++;
     }
  }
```

Here, STEP represents the time interval, and difftime() is a library function for determining the time interval.

C. Estimating the Value of H by Iteration

Our aim is to design a scheme for fast network traffic anomaly detection. We want to use the iterative method[6] to estimate the value of Hurst and apply it to anomaly detection. This method can greatly improve the speed of estimating the value of H, and speed up the process of anomaly detection. Also we can use the previous estimate as the initial value of next estimation on-line, and further improve the speed of iteration. The flow chart is shown in figure 2:

We set a symbol "flag" to indicate whether system starts for the first time. If the system starts for the first time, the initial value of iteration is set to 0.5[6], and "flag" is 0; if not, the iteration starts from the value of last estimation, and "flag" is 1. Because the values of two consecutive estimates are close, by this way we can further reduce the number of iteration, and improve the speed of iteration.

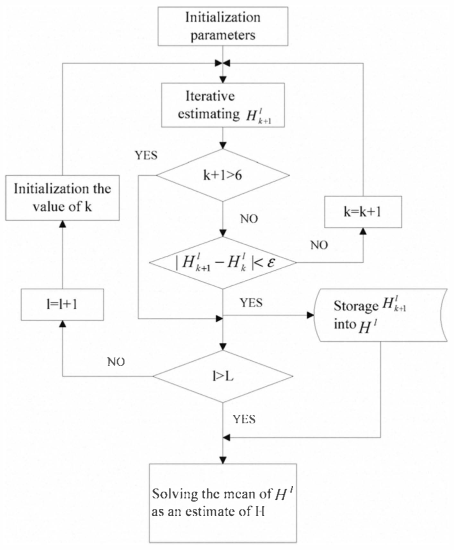
Initialization parameters
Iterative estimating H_{k+1}^l
YES
k+1>6
NO
$|H_{k+1}^l - H_k^l| < \varepsilon$
NO
k=k+1
Initialization the value of k
YES
l=l+1
Storage H_{k+1}^l into H^l
NO
l>L
YES
Solving the mean of H^l as an estimate of H

Figure 2. The framework of estimating the value of H

By improving the algorithm in [6], the proposed iteration algorithm is shown as follows:

```
for(k = 0,l = 0; l < L; l ++)
{ // indicate whether system starts for the first time
 if (flag = = 1)
 // iteration start from the value of last estimation
 $H_0^l = H_{pre}$;
//the initial value of iteration is set to 0,5
 else $H_0^l = 0.5$;
//iteration
while($| H_{k+1}^l - H_k^l | > \varepsilon$)
        (
  k=k+1;
  $H^l_{k+1} = sqrt((\rho_1 - H^l_k)*0.5)$ ;
  if (k > 6)
   break;
  }
  k = 0;
  $H^l = H^l_{k+1}$ ;
}
```

D. Determining the Network Traffic Anomaly

After estimating the value of Hurst, we need to determine whether the anomaly occur or not by the latest estimates. The "self-similar" feature of the network traffic is characterized with the Hurst parameter. Acute traffic changes in short time will greatly affect traffic density, and destroy the "self-similar" feature of traffic. The value of H increases as the degree of self-similarity increases. When the attack occurs, the "self-similar" feature decreases or even disappear, and the value of H will tend to 0.5[1]. When extreme bursty network traffic occurs, the value of H will jump to 1[7].

Generally we need a threshold to determine whether anomaly occurs. We try to determine the anomaly while estimating the value of H to ensure the accuracy and real-time of detection. Flow chart is shown in figure 3:

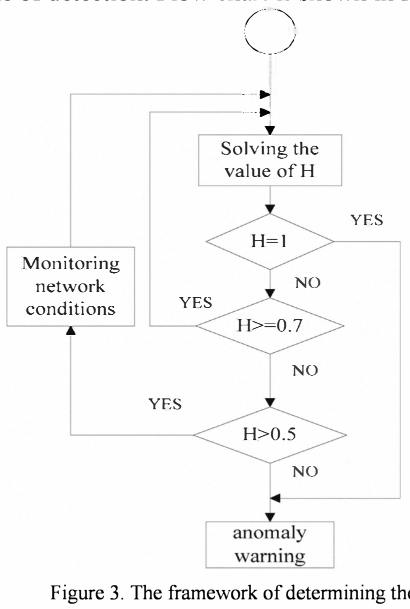

Figure 3. The framework of determining the anomaly

Ⅳ. ANALYSES AND DISCUSSIONS

A. Scheme Rationality

Under normal network conditions, the value of H is about 0.8[1]. But when the attack occurs, the degree of network traffic self-similarity will decrease, along with it, the value of H will quickly reduce. This character can be used as a basis for determining the anomaly.

The value of H is larger than o.7 in normal conditions [1], so we do not monitor network in this condition without the value of H is 1[7]. When the anomaly occurs, the "self-similar" feature of traffic is disturbed, and thereby the value of H decreases. When the value of H is larger than 0.5 and smaller than 0.7[1], the anomaly is likely to happen, so the network is in monitoring state and continue to estimate the value of H; Moreover, when the value is smaller than 0.5[1], the "self-similar" feature of traffic disappears, so we should send out a signal of warning and examine the network. If the value of H jumps to 1[7], then also a signal of anomaly warning should be sent out.

In order to achieve the purpose of fast implementation, we choose an iterative method[6] to estimate the value of H, which is characterized by accurate and quick estimation the value of H, and thus we can ensure a real-time detection. What we have to do is to improve the algorithm and apply it to determine the anomaly by certain threshold, thus we can achieve the fast network traffic anomaly detection.

We design a complete idea of our scheme, and describe the implementation of each part of our scheme. Analyses show that this scheme can be used as a method of fast network traffic anomaly detection Thereby we could prove that the overall structure can be achieved.

B. Scheme Performance

Compared with the existing other schemes, our scheme makes use of the H parameter, which is simple and can reflect the characteristics of the network itself, and, at the same time, we use iterative method[6] to estimate the value of H, which greatly improve the speed of estimation, and achieve rapid detection for network traffic anomaly.

In the Windows environment, we use MATLAB to simulate our algorithm. From [6], we know that, when system starts for the first time, the initial value of H is 0.5, and the average time for iteration is 0.005s, and the average number for iteration is 6. By the on-line mode, we can use previous estimates as the initial value of the next estimation, and analysis shows that we can reduce the iterative number by this way.

As the iterative number reduces, the iterative time also significantly reduces. When we use 1000 latest sample data, the iteration number dropped to 1.2[6], and the time reduced to 0.001s. Thus, as time goes by, due to adopt the latest samples for iteration, the iteration time will be reduced largely. By increasing the sample data, we can increase the precision of iterations, and thereby further reduce the iteration time. The core concept of our scheme is the estimation of the H parameter, so the speedup of this step plays a significant role in fast implementation of the scheme.

Ⅴ. CONCLUSIONS

In this paper, a scheme for fast network traffic anomaly detection is proposed. Compared with conventional anomaly detection methods, our method can reflect the characteristics of the network traffic itself, and it is simple and intuitive. Analyses show that it can be fast implemented, and it is easy to optimize. In future work, we would like to optimize the scheme and to apply it to fast security test area.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant No. 60803151; the Key Program of NSFC-Guangdong Union Foundation under Grant No.U0835004; the Natural Science Foundation of Shannxi Province, China under Grant No. 2007F37; the National Science Foundation for Post-doctoral Scientists of China under Grant No. 20070410376; the 111 Project under Grant No. B08038.

REFERENCES

[1] Leland WE, Taqqu MS, Willinger W, Wilson DV. On the self-similar nature of ethernet Traffic (extended version). IEEE/ACM Trans. on Networking, 1994,2(1):1−15.

[2] Paxson V, Floyd S. Wide area traffic: the failure of Poisson modeling. IEEE/ACM Trans. on Networking, 1995,3(3):226−244.

[3] Dang T D ,Molnar S. On t he Effect s of Non-Staionarity in Long Range Dependent Test s[R] . Budapest Univ Technology and Economics Tech. Rep. Budapest ,Hungary ,1999

[4] Teng, H.S., Chen, K., Lu, S.C.: Adaptive real-time anomaly detection using inductively generated sequential patterns. IEEE Symposium on Security and Privacy (1980) 278–284

[5] Lane, T., Brodley, C.: Approaches to online learning and conceptual drift for user identification in computer security. ECE and the COAST Laboratory Tech. Rep. (Coast TR 98-12), Purdue University (1998)

[6] Li Lin-feng Qiu Zheng-ding An Iterative Method to Estimate Hurst Index of Self-similar Network Traffic. Journal of Electronics & Information Technology, (2006)12-2371-03

[7] Wang Xin, Fang Bin-xing: An Exploratory Development on the Hurst Parameter Variety of Network Traffic Abnormity Signal.

[8] Cannady, J.: Next generation intrusion detection:

[9] Autonomous reinforcement learning of network attacks. 23rd National Information Systems Security Conference (2000) 1−12

[10] Hossain, M., Bridges, S.M.: A framework for an adaptive intrusion detection system with data mining. 13th Annual Canadian Information Technology Security Symposium (2001)

[11] Grossglauser M, Bolot J. On the relevance of long-range dependence in network traffic. Computer Communication Review, 1996, 26(4):15 −24.

[12] Adas A. Traffic models in broadband networks. IEEE Communications Magazine, 1997, 35(7): 82−89.