DDoSniffer: Detecting DDOS Attack at the Source Agents

Vicky Laurens[1], Alexandre Miège[1], Abdulmotaleb El Saddik[1] and Pulak Dhar[2]

[1]Multimedia Communications Research Laboratory, University of Ottawa

[2]Cistech Limited, 210 Colonnade Road, Unit 3, Nepean, ON K2E 7L5

{*vicky*, *amiege*, *abed*}*@mcrlab.uottawa.ca*, *and* pulak@cistech.ca

ABSTRACT

Distributed Denial of Service (DDoS) attacks are an important and challenging security threat. Despite the existing defence mechanisms, attackers manage to build large sets of impersonated hosts. Our approach consists in detecting DDoS directly on these hosts. We classify ongoing attacks as *connection attacks* or *bandwidth attacks*. The former are defined as attacks that generate connections with four packets or fewer; the latter as attacks that create connections with traffic ratios larger than usual. We developed a software tool, DDoSniffer, which enforces those principles. We show that it is capable of detecting a broad range of attacks within seconds.

1. Introduction

Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks are aimed to thwart legitimate users from accessing shared resources. In general, DoS/DDoS attacks prevent legitimate clients from accessing Internet services provided by Web servers, FTP servers, mail servers, DNS servers, *etc*. A DDoS attack differs from a DoS attack in that several agents are impersonated and used by the attacker to send a large number of requests to the targeted server. In both cases, the users are not aware that their computers are being used to mount an attack.

The DoS/DDoS attacks first became to the attention of the general public with the highly publicized DDoS attack to the Internet root servers (DNS) in October 2002 [1], then with the

DDoS attacks to Google, Yahoo, Microsoft, Lycos, and AltaVista [4]. DDoS quickly became a major threat over the Internet. In the last six months of 2006, an average of 5,213 DoS attacks per day was observed by Symantec [5]. In the 2004 CSI/FBI ranked DoS attacks as being the second most damaging threat with financial losses of more than US$26 million. Even new criminal extortion methods based on DoS attacks appeared, like for online gaming companies [7].

Despite the availability of several commercial solutions and ongoing academic research work, attackers are still successful, for several reasons. First, home and universities computers, which are the main targets of the recruitment phase, are more and more connected via high-speed Internet subscriptions. Second, personal firewalls, which can help in some cases to protect end-users' computers, are only used by 67% of the people **Error! Reference source not found.**. Third, attack tools have made great progress and are easy to use by non-expert users. Recent DDoS trends include blended threats that could be referred as "all in one": recruitment, spreading, and attack payload in a single program tool. Finally, new vulnerabilities in host computers are constantly discovered. [5] indicates that during a six month period, 2,526 new vulnerabilities were found; 79% of them were ranked as easy to exploit [5]. Obviously, DDoS detection is still an important field of research.

The appearance of new DDoS techniques and tools, along with new vulnerabilities, makes it very difficult to keep up to date. As a consequence, detection must be carried out based on generic attack features rather than on attack signatures. For this reason, we introduce the notions of *connection attacks* and *bandwidth attacks*. We show how these two types of attacks can be characterized and then how they can be detected. Up to this point, our research is focused on the TCP-based attacks, merely because 90% of attacks use TCP [6].

We must point out another important feature of our approach: the detection location. Figure 1 shows the different networks involved in a DDoS attack. Defence mechanisms can be deployed at the victim network, the intermediate network, and the source network [9], but in practice they are mostly deployed at the victim network. This is the case for instance of the MDADF scheme developed in [2] where, based on packet-marking techniques implemented at the intermediate network routers, the firewalls of the victim networks are able to filter part of the DDoS attacks. Instead, our approach consists in detecting DDoS on the hosts where attack packets are generated, that is, at the source network.

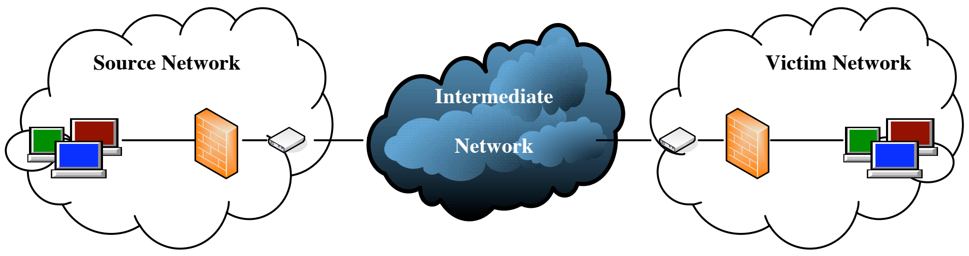

Figure 1: Deployment locations of DDoS defence mechanisms.

Historically, source-end defence mechanisms have lacked deployment motivations mainly because a DDoS attack inflicts the most damage on the attack target. However, source networks, - e.g. universities, companies, ISPs - may also suffer from DDoS attacks. With new DDoS attack trends, sensitive data can be gathered from agents after the intrusion. Indeed Symantec reported that 54% of identified threats are able to expose confidential information [5]. Especially companies should take this threat into account. Second, when an attack occurs, the source network traffic is affected and the overall performance can drop. Finally, by letting a DDoS attack being launched from its network, an ISP or a company could have its public image damaged.

As a consequence, whoever is in charge of the source network will benefit from having all end-users' computers protected. We do not claim that detecting DDoS attacks directly on impersonated hosts is a response to everything, but it certainly has to be considered. DDoS detection has this distinctive feature that only complementary mechanisms involving all networks and agents can permit to seriously decrease DDoS attacks appearances and consequences. In other words, the solution we provide can, and should be, enforce on top of existing solutions.

An important part of this contribution is the development of a software tool, *DDoSniffer*. This work enables us to run many tests and evaluate our approach. The performance results demonstrate that it is feasible to detect a broad range of TCP-based DDoS attacks on users' computers participating in an attack.

The remainder of the paper is organized as follows. Section 2 discusses the related work. Section 3 introduces the principle of our approach and the attack detection methodology. In section 4, we focus on DDoSniffer's design and functionalities. Section 5 explains the experimental setup and the performance results. In Section 6, the conclusion and future work are discussed.

2. Related work

Since our approach consists in detecting DDoS on the hosts, we focus in this section on solutions deployed at the source network. However none of the exiting solutions aims at detecting attack traffic at each particular agent computer [16]. They are implemented at the edge of networks in order to have access to the whole incoming and outgoing network traffic.

Gil and Poletto propose MULTOPS (Multi-Level Tree for Online Packet Statistics) [17]. It is deployed on network monitors and routers. It consists of a tree of nodes (containing traffic ratio statistics) that contracts or expands itself depending on traffic patterns. Attack detection is based on the heuristic that under normal operation, the traffic flows to and from one direction are proportional. MULTOPS has two modes of operation: the first mode identifies the victim of an attack, and the second identifies the source(s) of an attack. Depending on the operation mode, a flow is classified as an attack if the traffic ratio drops below a threshold (victim-oriented mode) or if the traffic ratio surpasses a threshold (attacker-oriented mode). Once flows are classified as an attack, they are rate-limited. One drawback of MULTOPS is that if IP spoofing is employed, MULTOPS may impose collateral damage by dropping legitimate packets when operating in victim-oriented mode. Moreover, MULTOPS might not detect the attack at all when operating in attacker-oriented mode. Another drawback of MULTOPS is possible misclassification of non-TCP traffic due to the fact that the heuristic of proportional flows is based on TCP operations (packet acknowledgements).

Kommareddy *et al*. proposed Stub-Domain DDoS Detection **Error! Reference source not found.**, a system consisting of two components: one for identifying attack flows, and the other for eliminating false detections by analysing inputs from the first component. Traffic monitors are attached to routers at the Autonomous System (AS) where the attack packets are generated. This is the first source-domain system designed to work in stub-networks, that is, networks that only carry packets to and from local hosts. This system works with asymmetric traffic and bases its decision on traffic ratio as in MULTOPS. Different monitors communicate with one another in order to decide on suspicious flows and decrease the number of false detections.

Mirkovic proposed D-WARD (DDoS Network Attack Recognition and Defense) **Error! Reference source not found.**, which detects and discards DDoS attack traffic originating at the source network. It collects two-way traffic statistics from the border router and compares these statistics with previously built legitimate traffic models. When an anomaly is detected, rate-limiting policies are applied. Rate limiting attack traffic is preferred over blocking attack traffic to allow faster recovery from false detections. Due to the fact that D-WARD applies rate-limiting policies, the system is quite complex in terms of implementation. For example, D-WARD distinguishes between flow traffic and connection traffic in such a way that legitimate traffic is favoured during an attack.

Wang *et al*. proposed FDS (Flooding Detection System) [6], which monitors traffic at leaf routers that connect end hosts to the Internet. The FDS can be located either at the first-mile leaf router (near the source) or at the last-mile leaf router (near the targeted victim). It bases detection on the relationship between SYN packets and FIN packets. Under normal circumstances, a SYN packet will eventually generate a FIN within a normal flow. The authors deal with the case of connections terminated with RST packets by empirically recognizing some of the total RST packets as active RST (which is equivalent to a FIN packet).

S. Jin and D. Yeung proposed a covariance model to detect DDoS attacks [18]. SYN flooding attacks are used as an application of the proposed method. As with FDS, this detection method can be implemented near the targeted victim or near the source of the attack. This covariance analysis model is a statistics-based method, and for SYN flooding attacks, all TCP flags are used as features in the model. The correlations among the features (the TCP flags) differentiate normal traffic from attack traffic. One advantage of this method over other statistical methods is the absence of assumptions about packet distribution.

In section 5.6, we compare the performance of our tool with the contributions we described.

3. Principle of the approach

3.1 DDoS attacks classification

In general, DDoS attacks are classified either as *flooding attacks* or *vulnerability attacks*. The key factor in the former type of attacks is to send enough packets in order to consume the victim's resources (CPU cycles, network bandwidth, memory, *etc*.). A typical example of flooding attacks is the TCP SYN Flooding attack that is usually based on breaking up the three-way handshake that occurs when a TCP connection is opened. The other type of attack, the vulnerability attack, is based on exploiting one or more vulnerabilities existing in the targeted system, like a bug in an application program, in a protocol, or in an operating system. The case described in [14] of the Cisco 600 family routers that could be locked by sending a specific URL if the router was configured to allow Web Access is a good example of a vulnerability attack.

In this paper we suggest a new classification by introducing the concepts of *connection attack* and *bandwidth attack*: in their basis, TCP-based DDoS attacks generally break typical TCP operations in two ways, by generating connections with four packets or fewer (*connection attack*) or by generating connections with traffic ratios (incoming/outgoing) larger than four usual (*bandwidth attack*). Note that we use the term ‘packet’ even though a TCP datagram should be called a ‘segment’. This last term is seldom used, whereas the term packet commonly appears in the literature. However, we emphasize that in the following we are working at the transport layer level (with TCP), not at the network layer level.

We claim that our classification makes it easier to characterize and detect an attack because it is based on how the TCP semantics is broken. The two following subsections are dedicated to these two types of attack.

3.2 Connection Attacks

Four or fewer packets. We characterize connection attacks as attacks that violate typical behaviour of TCP connections by generating connections with four packets or fewer. Let us make a brief recall on the TCP protocol. TCP is connection-oriented, in other words, a TCP connection has to be established between a client and a server in order to be able to send data to each other. TCP establishment is asymmetric; it starts with the three-way handshake (SYN-SYNACK-ACK), then the connection enters the established state in which data flows between the two end-points. To terminate the connection, either end-point sends a FIN packet which is acknowledged by the other end-point. At this point the TCP connection is said to be *half-closed*. The connection is completely closed when the other end-point sends a FIN packet, which is also acknowledged. Thus, opening a connection requires three packets and closing it requires four packets. It is also possible to terminate a connection by sending an RST packet, which does not require acknowledgement from the other end. As a consequence, normal TCP connections should have more than four packets. In other words, connections with only four or less packets are suspicious due to the reasons mentioned below.

In the case of a flooding attack, connections participating in the attack will have four packets or fewer: the three-way handshake packets (if completed) and probably one more packet with a service request. SYN flooding attacks are the most common attacks among its class. However, there are several distinct attack vectors to mount a DDoS flooding attack. Table 1 shows several flooding attack vectors, the corresponding server's response, and the number of packets per connection.

Table 1: TCP flooding attack vectors.

Attack Packet	Server Response	Number of Packets per Connection
SYN (to an opened port)	SYN-ACK	1: if IP spoofing is employed 2: if the TCP/IP stack is modified to not send the final ACK 3: if the TCP/IP stack is not modified 4: if the connection is established and a service request is sent
SYN (to closed port)	RST	2
ACK	ACK	≤ 4 (or a traffic ratio larger than usual)
RST	No response	1
NULL	RST	3
FIN (to a closed port)	RST	2

The case of a vulnerability attack is more complex since it depends on how the vulnerability is exploited. Precisely, if exploiting the vulnerability generates new connections with four packets or fewer, we should be able to detect the ongoing attack.

Number of suspicious connections. We have just shown how to characterize a suspicious connection. However a single suspicious connection is not enough to generate a DDoS attack. It is rather intuitive to consider that if only a single suspicious connection is detected, it would be excessive to raise an alert. The number of suspicious connections is thus a key factor. During an attack, the number of connections with four packets or fewer will grow very fast. Thus, it is very important to specify how many suspicious connections must be detected in order to conclude on an attack. Unfortunately, that threshold is empirical. It cannot be inferred from the TCP protocol specification itself and can only be determined by monitoring operational traffic. We call this threshold *NEWCONN*. Note that a large value of *NEWCONN* may delay attack detection, but it also prevents false positives in the case where legitimate packets get lost due to network congestion. Similarly, a shorter value of *NEWCONN* expedites attack detection, but it also increases the number of false positives. *NEWCONN* value will be defined in section 5.2.

3.3 Bandwidth Attack

As defined earlier, bandwidth attacks violate the semantics of the TCP flow control mechanism. In the TCP protocol, communication reliability is carried out by exchanging control and supervision packets. These packets are used for instance to acknowledge the packets delivery or to control the packet flow speed in order to avoid buffer overflows [12]. Conforming to the flow control specification, a TCP client will send packets according to the TCP server's permission. This permission is based not only on the server's resources, but also on the network conditions. For instance, if some packets are lost due to network congestion, it will force a legitimate client to slow down its transmission rate to adapt to the current situation.

Due to the TCP control mechanisms the difference between received and sent packets during a TCP connection is expected to be relatively small and stable in time. The traffic ratio of a connection, called *Connection Ratio* afterwards, is given by the following equation:

$$\textit{Connection Ratio} = \textit{Number of outgoing packets} \,/\, \textit{Number of incoming packets} \quad (2.1)$$

Let us consider now an impersonated client (attacker). It will not respect the flow control mechanisms. It will not wait for the server's acknowledgements; instead it will keep sending aggressively attack packets in order to consume the victim's resources. Therefore, an attack connection will have a traffic ratio larger than usual. Determining benchmarks for bandwidth attack detection (based on traffic ratio) poses a big challenge because the traffic ratio of a given connection depends on the following factors:

- TCP implementation: The TCP specification does not define an explicit algorithm to manage the window of sequence numbers. Particularly, the number of packets that have to be received prior to sending an acknowledgment is not fixed (sliding-window size). Moreover, a TCP client/server usually does not acknowledge receiving a packet immediately; it waits to see if there is data to be sent in the same direction as the ACK [13] (piggybacking technique). For these reasons, the connection ratio is not equal to one.
- User's behaviour: the TCP protocol uses different algorithms to manage diverse types of data. FTP and electronic mail applications, for instance, generate bulk data whereas Telnet produces interactive data. On the other hand, delayed acknowledgments, selective acknowledgments, the Nagle algorithm [14], window scaling [11] and other extensions to the original TCP specification cause variations in the traffic ratio. Moreover, these variations depend on user's behaviour since the user decides which applications to run.

Detecting bandwidth attacks implies that we are able to distinguish attack traffic from legitimate traffic based on their connection ratio. Therefore, a threshold must be defined. Let us call *R* this particular value of the connection ratio. Traffic with a connection ratio lower than R would be considered as legitimate traffic; in contrast, traffic with ration higher than R would be flagged as attack traffic.

However, determining a value of *R* is not immediate. As the distribution of traffic is unknown, heuristics observations are necessary to determine typical values for the traffic ratio. As for the *NEWCONN* factor introduced in section 3.2, *R* must be set carefully in order to get as less false positive and false negative as possible. In section 5.2, we show how the ratio *R* is set.

In this section, we defined a novel DDoS attack classification. Characterizing ongoing attacks is becoming very fuzzy because attackers are able to change attack vectors very easily. Consequently, it is important to detect attacks regardless of their methodology and the means they use. By monitoring both the number of connections with four packets or fewer and the traffic ratio per connection, we are able to detect most TCP-based DDoS attacks.

4. DDoSniffer

4.1 Objectives

We present *DDoSniffer*, the tool we developed based on the detection principles described in the previous section. DDoSniffer will enable us to validate our approach, and also to determine the key parameters *NEWCONN* and *R*.

As explained in section 1, our approach consists in detecting DDoS attacks by monitoring the network traffic at each agent participating in an attack. Therefore, DDoSniffer is designed to be deployed on an user's computer. The main objectives are:

- Monitoring network traffic and detecting ongoing TCP-based DDoS attacks. It has to notify the user by raising an alert.
- Detecting the two types of TCP-based attacks mentioned in section 3.1, namely connection attacks and bandwidth attacks. An extra feature of DDoSniffer is IP spoofing detection, which is mostly employed in reflection attacks.
- Keeping processing time per packet as short as possible to be able to handle all packets, even during heavy traffic load.

Note that the main objective of DDoSniffer is to raise an alarm in order to warn the user when his/her computer is used to perform a denial of service attack. There is no reaction module at this time. As a consequence, the user has to take the appropriate action.

4.2 Principle and architecture

DDoSniffer is a passive packet sniffer, in other words, our tool does not affect the computer's network traffic. DDoSniffer monitors TCP traffic and maintains two tables, the *new-connection table* and the *connection table*. In the first table we keep the connections that might be considered as connection attacks; in the second table we keep the connections that might be flagged as bandwidth attacks.

As attack traffic resembles legitimate traffic, a given packet cannot be classified as part of an attack upon its arrival at the DDoSniffer computer (i.e. the computer that runs DDoSniffer); instead it has to be put into the tables for a later classification. This traffic classification is done over time in order to be more accurate in attack detection. Another reason to avoid classifying packets immediately is to prevent missing out packets due to the time spent processing each packet.

Three parallel modules compose the architecture of DDoSniffer: the *capturing module*, the *new-connection module*, and the *bandwidth module*. The new connection module is dedicated to the connection attacks detection and is based on the new-connection table whereas the bandwidth module is dedicated to the bandwidth attacks detection and is based on the connection table. Figure 2 illustrates a high level diagram of DDoSniffer's architecture. The three modules run in parallel, sharing access to the new-connection table and the connection table.

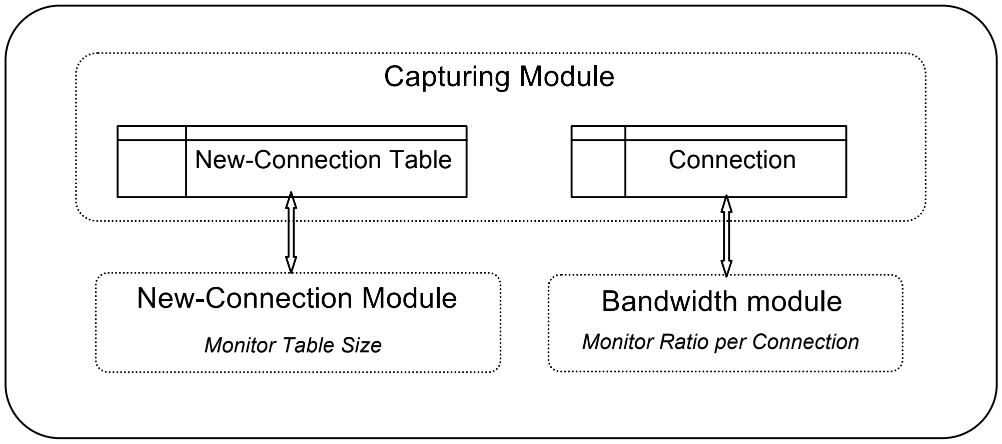

Figure 2: Architecture of DDoSniffer.

4.3 Capturing Module

New-connection table and connection table. The capturing module monitors network traffic, filters out all but TCP traffic, and builds the new-connection table and the connection table. The capturing module does not affect the network traffic in any manner since it captures the packets from the TCP/IP stack. All TCP packets are analyzed. The tables are implemented as hash-tables. A hash-table provides fast access to any of its record data provided the table index is chosen effectively.

The two tables are built in the same manner as presented in table 2. Each table record corresponds to a connection identified and indexed by the pair of sockets (IP address + TCP ports) participating in the connection, that is, by the first four rows of table 2. Connections are indexed based on the first connection packet. For example, if the first packet of a given connection is an outgoing packet, the source IP address field contains the IP address of the computer DDoSniffer is running on, the destination address field takes the destination address of the packet, and so forth.

Rows 5 and 6 of table 2 are used to count the number of packets per connection (connection attacks), and are also used to compute the connection ratio in row 7 (bandwidth attack). Three other fields are stored in each record of both tables because some extra information is deemed necessary in order to base detection on overall changes in packet behaviour instead of on an individual per packet basis.

Table 2: List of the fields of a connection record in both connection tables.

	Field	**Explanation**
1	Source IP address	
2	Destination IP address	
3	Source port	
4	Destination port	
5	Counter of incoming packets	
6	Counter of outgoing packets	
7	Connection ratio	Number of outgoing packets out of the number of incoming packets (Equation 2-1). This field is computed only for connections with more than four packets.
8	Current status	*Suspicious*: if the number of packets is less or equal to four. (default value when the record is created). *Normal*: if the connection ratio is less than R. *Attack*: if the connection ratio is greater than R.
9	Time stamp	To signal the time of the last operation.
10	Last status	Same possible values as the field Current status.

Capturing traffic. The following diagram presents the process of capturing and filling the connection tables.

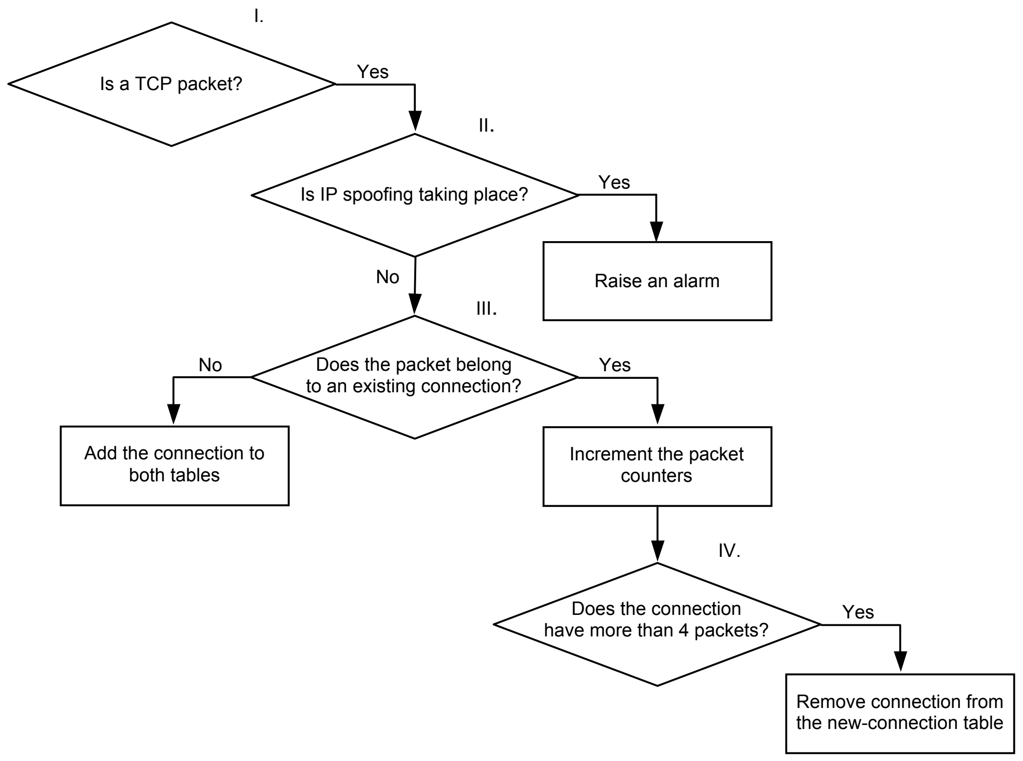
I.
Is a TCP packet?
Yes
II.
Is IP spoofing taking place?
Yes
Raise an alarm
No
III.
No
Does the packet belong to an existing connection?
Yes
Add the connection to both tables
Increment the packet counters
IV.
Does the connection have more than 4 packets?
Yes
Remove connection from the new-connection table

Figure 3: Flow of a packet in the capturing module.

(I) The capturing module first filters out non-TCP traffic. (II) Then, it checks whether the packet corresponds to an IP spoofing attack and notifies the user in case of an attack. (III) If not, the following step consists in checking if the packet falls within an existing connection. This is done by comparing the IP addresses and ports of the packet with the indexes of existing connections. If this packet corresponds to a new connection, it is added to both tables since any new connection can be part of a connection attack or a bandwidth attack. Otherwise the respective connection packet counter (incoming or outgoing) is incremented. (IV) When a counter is updated it is necessary to verify the total number of packets (sum of the two counters) of the corresponding connection: if the connection has more than four packets then the connection record is removed from the new-connection table and its connection ratio is computed and saved on the connection table record. Indeed, with more than four packets the connection is no longer suspicious with regards to the connection attack detection, as explained in section 3.2. If the connection has less than four packets, no action is taken.

Regarding the current status field, when a connection is first added to the tables, its status is set to *Suspicious* because a new connection should be considered suspicious until its number of packets is greater than four. Once the connection has more than four packets, it is no longer in the new-connection table. Its status field is then set to *Normal* or *Attack* by the bandwidth module according to the traffic ratio of the connection.

As described in the diagram, the capturing module enables to detect IP spoofing attacks. Since, we monitor the traffic at the host level, each packet going in or out must have the host IP address as the source address or the destination address. Otherwise, an IP spoofing alert is raised.

4.4 New-Connection Module

This module is designed with the objective of detecting connection attacks. It relies on the new-connection table. As we have just shown in the previous section, the new-connection table keeps record of current open connections that have four or fewer packets.

Under normal circumstances, the size of the new-connection table should be near zero because legitimate connections complete their three-way handshake in fractions of a second; but during a DDoS flooding attack, the size of the new-connection table increases within a few seconds.

If the size of this table is greater than *NEWCONN*, an alarm of an ongoing attack should be raised to the end-user. *NEWCONN* is an upper limit to the ideal size of the new-connections table: only *NEWCONN* connections are allowed to be in the table. *NEWCONN* is a system parameter that is empirically adjusted as explained in section 5.2. DDoSniffer attack detection highly depends on the adjustment of the *NEWCONN* parameter.

This module's operation is simplified due to the fact that connections with four packets or less are kept in a separate table, the new-connection table. Instead of having to read every record to count the number of connections that has four packets or fewer, we just have to check the size of the new-connection table.

The flow chart of this module is as follows: (I) First, the module obtains the size of the new-connection table. (II) If this size is lower than *NEWCONN*, an alarm is raised to the end-user and the user can take an appropriate decision on what to do next.

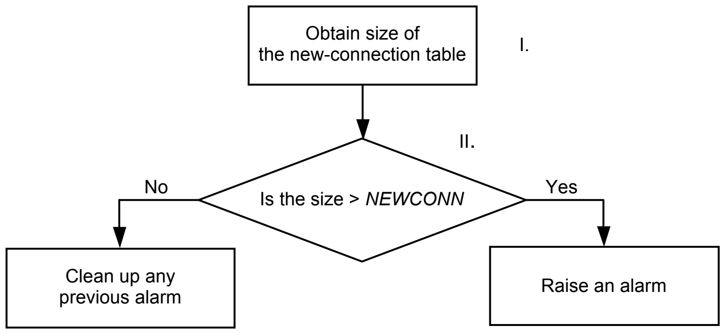
Obtain size of the new-connection table
I.
II.
No
Is the size > NEWCONN
Yes
Clean up any previous alarm
Raise an alarm

Figure 4: Flow of the new-connection module.

4.5 Bandwidth module

This module is designed to detect bandwidth attacks. It bases its operation on the connection table built by the capturing module. However, the traffic ratio of each connection is updated by the capturing module and not by the bandwidth module. In this manner, no extra time is unnecessarily spent computing traffic ratios for connections that have not completed the three-way handshake.

The bandwidth module classifies connections as *Normal* or *Attack* based on its connection ratio. An alarm is raised if at least one connection has its connection ratio greater than *R*.

As in the case of connection attack detection, the sensitivity of the DDoSniffer for detecting bandwidth attacks depends on a system parameter, in this case: the ratio *R*. The value of *R* is set in section 5.2.

By keeping two different tables for recording connections, we avoid the overhead of verifying the number of packets each connection has. At the time of an attack, this number could be very consuming in time due to the large size of the connection table.

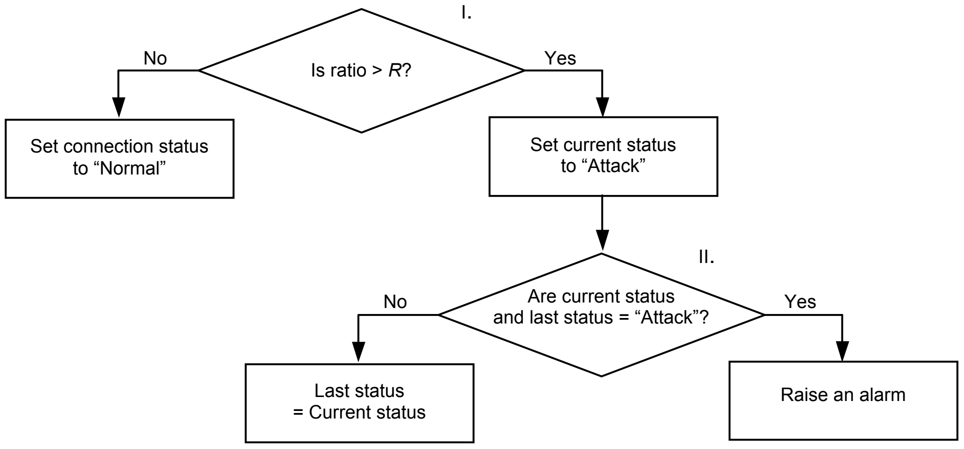
I.
Is ratio > R?
No
Yes
Set connection status to "Normal"
Set current status to "Attack"
II.
Are current status and last status = "Attack"?
No
Yes
Last status = Current status
Raise an alarm

Figure 5: Flow of the bandwidth module.

Figure 5 presents the process in the bandwidth module: (I) The module checks the traffic ratio of the connection. If it is lower or equal to *R*, the current status remains *Normal*. (II) If not, the connection status is set to *Attack*. (II) If the current and the last status are the same, this means that the connection ratio is greater than *R*, and as a consequence an alarm is raised. Otherwise the last status is set to *Attack*. The use of the current and last status enables us to create a sort of confirmation technique since it is tolerated that the traffic ratio of a connection be greater than *R* once.

5. Tests and results

Now that we described the detection methodology and the architecture of DDoSniffer, we are in a position to tackle the experimental work. First, we describe the experimental setup. Then we explain how we obtain the two key factors *NEWCONN* and *R*. At this point we will be able to expose the test results for both connection attacks and bandwidth attacks. Finally, we compare our results with those obtained with the other approaches described in section 2.

5.1 Experimental setup

The experimental platform consists of three modules we have developed: DDoSniffer, a background traffic generator and an attack tool.

DDoSniffer. In a real-life network, DDoSniffer should be deployed on each computer. However, in order to test DDoSniffer, the number of agent forming part of the attack is irrelevant. As a consequence we carried out the test on a single computer.

DDoSniffer is implemented using the Microsoft Visual C++ .NET development platform and is intended for use on Windows XP machines.

Background Traffic Tool. This tool simulates legitimate traffic during the experiments. It uses three weeks of traffic data provided by Cistech Limited to simulate a real-world network activity. We checked, by reading carefully the traffic traces, that no DoS/DDoS attack occurred during the period the data was taken. Those three weeks of data was filtered out to obtain two usable traffic traces as described below:

- TR01: Three weeks traffic of the ten most active users (in term of traffic generated on their computer) was used. The average packet rate per minute is estimated to 426. With a customized application that randomly opens connections, we generated 10 runs from this trace, using 10 random seeds.
- TR02: Three weeks traffic of the top user was used. The average packet rate per minute is estimated to 4,200. Using the same method as for TR01, we generated 40 runs.

Attack tool. This tool generates DDoS attacks to test DDoSniffer under diverse circumstances. To evaluate DDoSniffer's performance with regards to connection attacks, it is unnecessary to test several flooding attacks. Instead, by testing it with an HTTP-GET type of attack, other flooding attacks are implicitly tested as well. The attack tool opens one new connection every second for the duration of the attack, and each connection sends one GET packet over HTTP. In this manner, we are testing the smallest possible rate, that is, 1 connection per second: an attack with a lower rate (e.g. 1 connection every 2 seconds) would be considered as a pulsing attack, as described in section 5.5. Usually DDoS attacks generate much more connections per second; and the higher the rate is, the easier the detection is. Therefore, attempting to detect attacks that open only 1 connection per second is a challenge and a good way to evaluate our tool. Furthermore, if DDoSniffer can detect such attacks, it also can detect attack with higher rates.

Two connection attacks, attack 1 and attack 2, are defined from this method with the only difference that attack 1 lasts one minute and attack 2 lasts 10 seconds.

Regarding bandwidth attacks, the methodology is much simpler: the attack tool opens a TCP connection. Then one packet per second is continuously sent over the same connection for the duration of the attack. As in the case of connection attacks, two attacks lengths were tested: attack 3 lasts 1 minute and attack 3 lasts 10 seconds.

Table 3: HTTP-GET attacks set up.

Attack type	Attack number	Attack rate	Attack duration
Connection attacks	Attack 1	1 connection per second	1 minute
	Attack 2	1 connection per second	10 seconds
Bandwidth attacks	Attack 3	1 packet per second	1 minute
	Attack 4	1 packet per second	10 seconds

In all cases, the attack tool launches the attack 2 minutes 30 seconds after the legitimate traffic starts. The legitimate traffic lasts 5 minutes.

5.2 Defining NEWCONN and ratio R

As discussed in sections 3.2 and 3.3, *NEWCONN* and *R* are two key factors that can only be determined experimentally.

NEWCONN. We used the attack tool to generate a one-minute long HTTP-GET attack that opens one connection per second. Values from 1 to 10 were tested in order to select the optimum value for *NEWCONN*. For each value, the experiment was repeated five times with a different random seed used in the background traffic generator.

Two metrics were measured: average detection time and average number of false detection. It seems rather intuitive to consider that the optimum value of *NEWCONN* would be obtained when we detect attacks in a reasonable time, and at the same time, when we have a short number of false detections. As expected, with the smallest value of *NEWCONN*, the attack was detected with the shortest average time (4.4 seconds), but also with the largest average number of false detection (6.8). The shortest *NEWCONN* value obtained for which the average number of false positives is zero (no false alarm occurred) was 3. Consequently, *NEWCONN* was set to **3**.

Ratio *R*. As explained in section 3.3, the traffic ratio of a connection depends on several factors including the type of traffic. From the three weeks of traffic data provided by Cistech Limited, we calculated for each user the traffic ratio. The average maximum ratio was 2.8240, and the largest value was 3.3366. As a consequence, if we set the ratio *R* to 4, we have a very good threshold to detect bandwidth attacks: traffic with a connection ratio higher than 4 can be considered as illicit traffic. Although a smaller value of *R* could have been chosen, there is at least one case of a legitimate connection with a traffic ratio equal to four: if a connection is established and then terminated with a RST packet immediately after sending a GET packet (outgoing packets: SYN, ACK, GET, RST, incoming packets: SYN-ACK).

Table 4 shows the values for the system's parameters; both parameters affect DDoSniffer's performance.

Table 4: System configuration parameters.

Parameter	Value
NEWCONN	3
Ratio *R*	4

5.3 Connection attack test results

Three metrics are relevant to measure the performance of DDoSniffer: the lowest detectable attack rate, the shortest attack length and the detection rate (inverse of false negative rate).

Figure 6 depicts the results of detection of attack 1 over trace TR01 (2,130 legitimate packets, 10 runs). The x-axis shows the run number and the y-axis shows the detection time. The average detection time is 6.7 seconds, and the detection rate is 100%. This high percentage of attack detection is attributable to the low-rate of legitimate traffic as well as the duration of the attack.

Figure 7 presents the results of detection of attack 1 over trace TR02 (21,000 legitimate packets, 40 runs). The average detection time is 8.05 seconds and the detection rate is 100%. However, there is an average of 1.42% of packet loss in 35% of the runs. Packet loss in this context means some packets could not be captured and thus could not be added to the tables. Additionally, 50% of the runs during which packets were lost experienced misclassification of connections. Since some packets could not be captured, the number of packet per connection, as well as the traffic ratio, is processed based on an incomplete representation of the traffic. This is the reason of the misclassifications.

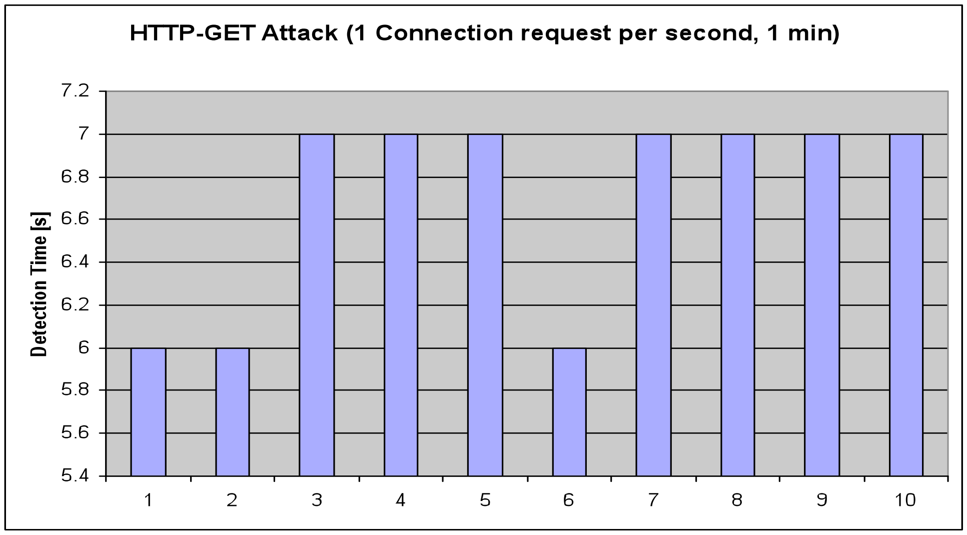

Figure 6: Detection time for TR01and Attack 1.

As expected, the detection time is longer for TR02 than TR01, because of the higher rate of legitimate traffic (4,200 packets per minute vs. 426 packets per minute respectively). This is due to the fact that access to the new-connections table and the connection table is shared among the three modules of DDoSniffer.

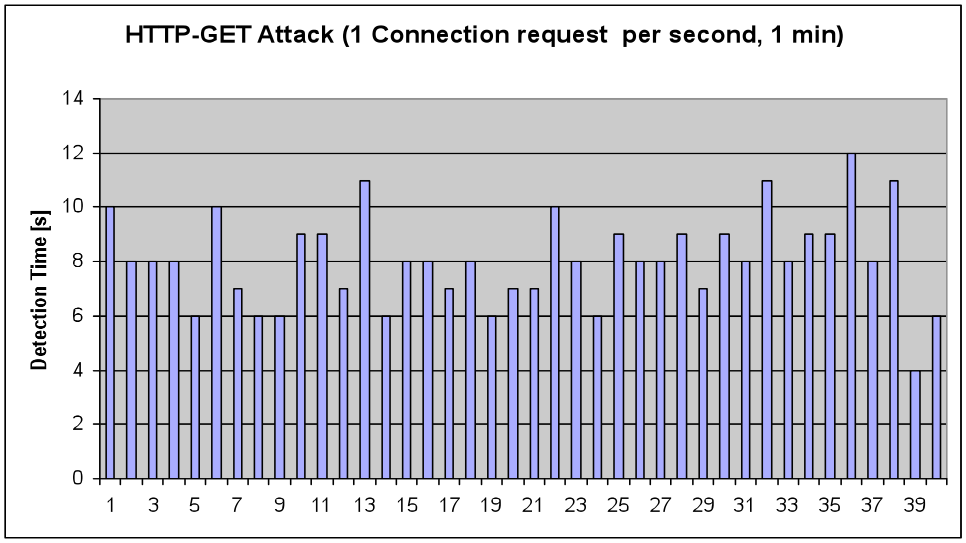

Figure 7: Detection time for TR02 and Attack 1.

Figure 8 shows the results for TR02 and attack 2. The average detection time increases to 8.225 seconds. The detection rate is 73%. Actually all attacks are detected but 11 of them are detected after the ten 10 seconds duration of the attack, and thus could not have been stopped by the user. However, the percentage of runs experiencing packet loss decreases considerably from 35% (TR01- Attack 1) to 2.5%, merely because the attack is shorter, and as a consequence there is less access to both tables. The 2.5% actually corresponds to only one run. This run experiences packet loss of 0.12% (which also occurs during the attack) and no misclassification of connections occurs.

Although attack 2 is shorter, the detection time is slightly longer than the previous attack. As the percentage of packets loss is considerably lower for attack 2, the total number of packets ends up being larger and this explains the longer detection time.

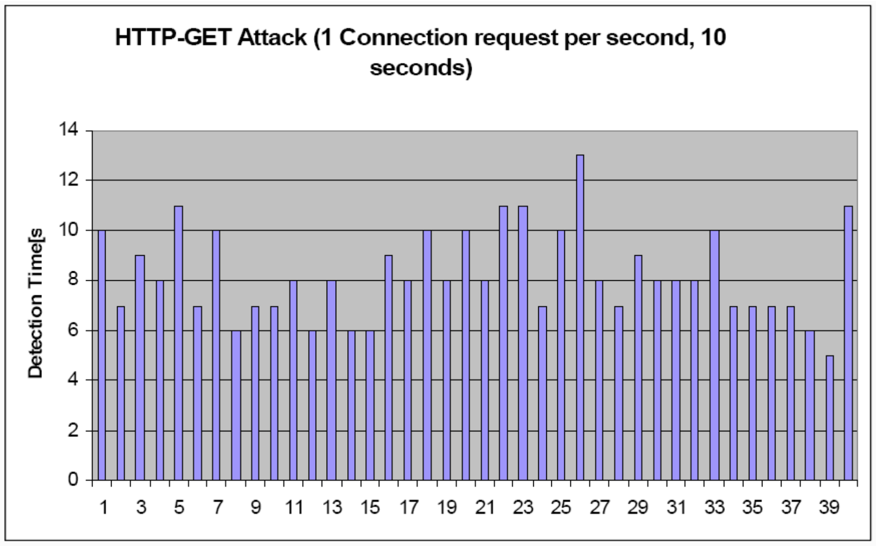

Figure 8: Detection time for TR02 and Attack 2.

5.4 Bandwidth Attacks

The same metrics apply for bandwidth attack detection. Figure 9 depicts the results for TR01 and attack 3. The average detection time is 3.5 seconds and the detection rate is 100%.

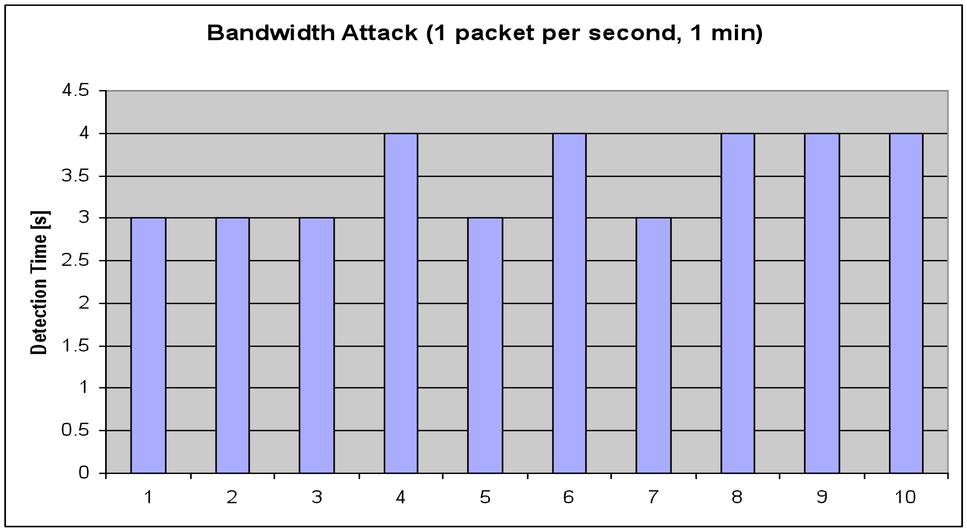

Figure 9: Detection time for TR01 and Attack 3.

Figure 9 shows the results for TR02 and attack 3. The average detection time is 9.8 seconds and the detection rate is 100%. Nevertheless, in 25% of the runs, packet loss occurred, at an average rate of 1.7%. As in the case of connection attacks, and for the same reasons, the detection time for TR02 is longer than TR01.

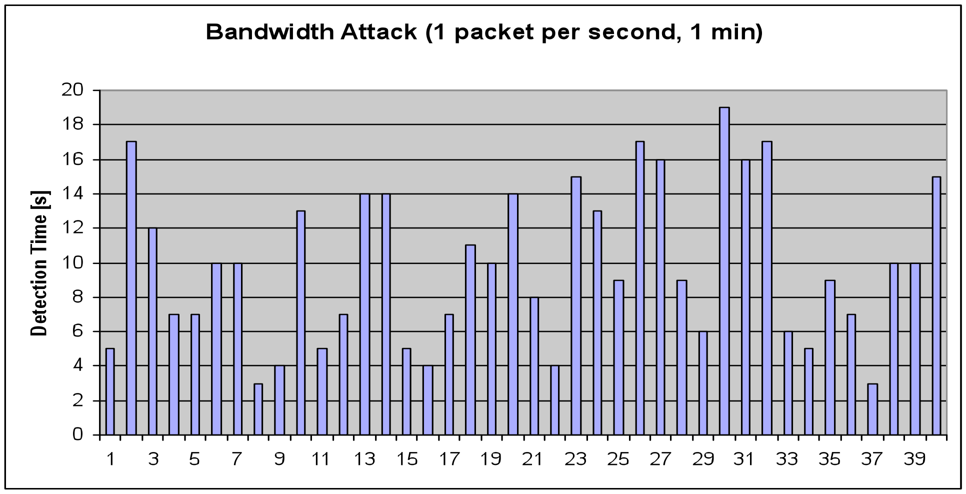

Figure 10: Detection time for TR02 and Attack 3.

Figure 11 illustrates the results obtained for TR02 and attack 4. The average detection time falls to 7.9 seconds and the detection rate to 87%. The packet loss occurrence drops to 2.5%. In the run which suffers packet loss, only 0.43% of the packets are lost and no misclassification occurs.

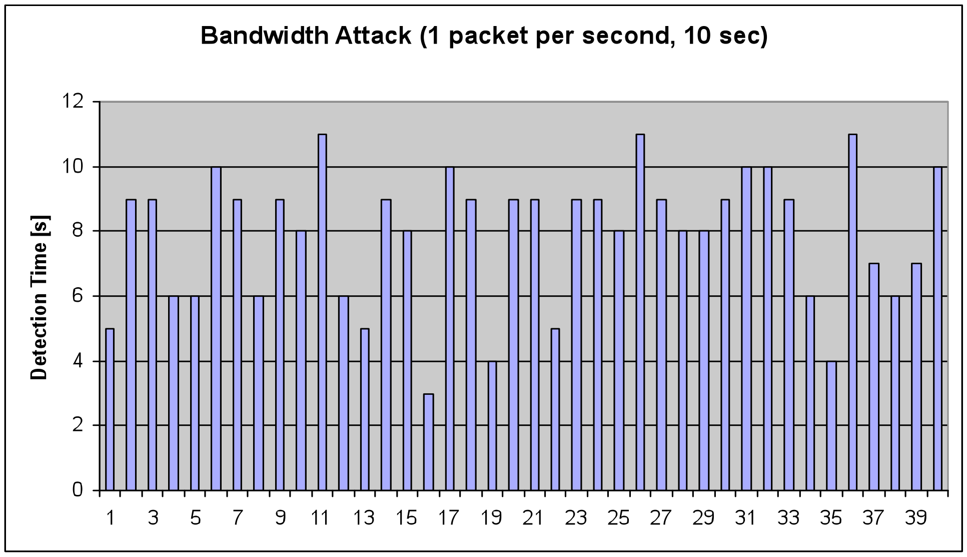

Figure 11: Detection time for TR02 and Attack 4.

Table 5 summarizes the experiment results discussed in sections 5.1 and 5.2. Both connection and bandwidth attacks have a rate of 1 packet per second. Experiments showed that DDoSniffer detected TCP-based attacks in 7.36 seconds (average time).

Table 5: Results summary.

Trace	Attack type	Attack duration	Detection Time [sec]	Detection Rate	Packet loss Occurrence	Packet loss
TR01	Connection	1 min	6.7	100%	0%	0%
	Bandwidth	1 min	3.5	100%	0%	0%
TR02	Connection	10 sec	8.225	73%	2.5%	0.12%
	Connection	1 min	8.05	100%	35%	1.42%
	Bandwidth	10 sec	7.9	87%	2.5%	0.43%
	Bandwidth	1 min	9.8	100%	25%	1.7%

5.5 Implementation discussion

Although efficient as it is, some features of DDoSniffer must be improved in order be fully effective. Especially, three important factors must be addressed to handle long runs of DDoSniffer without compromising its performance in detecting DDoS attacks.

First, the size of both tables, the connection table and the new connection table, must be carefully set. Since higher rate traffic introduces delay in attack detection, an attacker wishing to circumvent DDoSniffer can open a large number of legitimate connections to overwhelm the system. To overcome this limitation, the two tables can be bounded. Once this upper limit is reached, some connections are to be removed from the tables. The criteria for connection removal need to be investigated in the future because it cannot be only based on time.

Second, due to the length of the experiments, it was not necessary to clean up the tables. However, in the long run, inactive connections must be removed (regardless of the size of the tables). The timestamp field was included in the tables for this reason. In the case of the new-connection tables, any connection that has been longer than 75 seconds should be removed. 75 seconds is the duration a half-open connection is kept in the queue waiting to be completed, as indicated in the TCP specification [13]. In the case of the connection table, which connections must be removed is an open research aspect that needs to be further investigated.

Third, the size of the log file must also be limited. Although, originally raw data was recorded to provide insights into DDoSniffer operations (specifically when DDoS fails to capture all traffic), this log file can also be used by an advanced user in order to determine whether or not an alarm was due to a real attack or some other reasons. If the log file is not limited in size, the storage capacity of DDoSniffer computers can be exhausted.

As far as we know, the only type of TCP-based attacks that might defeat DDoSniffer are pulsing attacks. Instead of constantly sending packets, a pulsing attack is active for a given period of time (henceforth referred to as the "on time" T_{on}) and inactive for another period of time (henceforth referred to as the "off time" T_{off}). During T_{on}, the attacker transmits packets, and during T_{off} no action is taken. Frequently, pulsing attacks are used in a synchronized manner in such a way that sets of agents are *on* during the T_{off} period of other sets of agents. An attacker seeking to evade DDoSniffer by using a pulsing attack needs to send fewer packets than *NEWCONN* during T_{on} that has to be less than 7 seconds to avoid detection. This approach is not impossible, but it puts severe time constrains on the attacker. Synchronizing agents under such short time periods is a challenge the attacker needs to solve to achieve a successful denial of service attack.

5.6 Performance comparison

To gain a deeper understanding of DDoSniffer's performance, our results are summarized and compared to other approaches as presented in Table 6. However, it should be emphasized that DDoSniffer supplements existing defence mechanisms rather than competes against them. For example, an ISP can benefit from adopting any of these solutions and providing DDoSniffer as a detection tool for the end-hosts.

As seen in the above table:

- Attack rate: DDoSniffer detects the lowest attack rate. Only D-WARD equals this result **Error! Reference source not found.**.
- Attack variety: DDoSniffer can detect a large variety of TCP-based attacks, and can also detect IP spoofing. Similar features are offered by D-WARD **Error! Reference source not**

found..

- Detection time: Average detection time is 7.36 seconds, which is slightly longer than D-WARD's detection time but shorter than the rest of the solutions.

Notice that all of the above solutions are implemented at the border of networks that incur hardware costs. This is not the case for DDoSniffer.

Table 6: Performance comparison.

Defence Mechanism	**Attack type**	**Detection Time**	**Attack rate** [pps]	**Attack duration**	**Detection rate**
DDoSniffer	TCP	8.225 sec	1	10 sec	73%
	TCP	8.05 sec	1	60 sec	100%
	TCP BW	7.9 sec	1	10 sec	87%
	TCP BW	9.8 sec	1	60 sec	100%
D-WARD Error! Reference source not found.	TCP	7 sec	1	100 sec	N/A
Stub-Domain Error! Reference source not found.	TCP	10.14 sec	100	8 min	100%
	TCP	13.35 sec	50	8 min	100%
	TCP	27.88 sec	20	8 min	100%
	TCP	106.25 sec	10	8 min	99%
FDS Error! Reference source not found.	TCP SYN	40 sec	80	10 min	N/A
	TCP SYN	6 min	35	10 min	N/A

6. Conclusion and future work

We studied the DDoS threat from a general perspective to narrow down the roots of attackers' success: computers participation in DDoS attacks. We described in detail the methods employed by attackers to successfully mount a DDoS attack. Although attack strategies are very diverse, this diversity can be classified into two categories: attacks that generate connections with four packets or fewer, and attacks that generate connections with traffic ratios larger than usual. We called these two types of attacks *connection attacks* and *bandwidth attacks*.

Based on the decomposition of the strategies, we designed and developed a software tool, DDoSniffer. It is capable of detecting a broad range of attacks, as well as IP spoofing attacks. Even though it is a prototype, it already showed promising results. Using it, we proved it is possible to detect ongoing attacks at the agent computers within seconds.

Some improvements are still to be done. Among others, further research must be carried out to determine the most accurate table sizes. We also plan to extend our tool by developing a component to detect UDP-based attacks.

The distributed and complex nature of DDoS attacks requires several defence mechanisms interacting together. As a consequence, an extension to DDoSniffer is to trigger antivirus software and personal firewall installed on the DDoSniffer computer to complement our functionalities.

Finally, we envisage developing a reaction module that would filter the traffic. The reaction module would block the attacks much faster than a user receiving an alarm would do. Second, by blocking only the suspicious connections, the user would not have to block all the outgoing traffic.

Acknowledgement

This work was partially supported by Ontario Research Network on E-Commerce with collaboration with Cistech.

REFERENCES

[1] C. Duffy Marsan, and C. Garretson, “Net Security Gets Root-Level Boost”, *Network World Fusion*, October 2003; http://www.networkworld.com/news/2003/1027ddos.html

[2] Yao Chen, Shantanu Das, Pulak Dhar, Abdulmotaleb El Saddik, and Amiya Nayak, “Detecting and Preventing IP-spoofed Distributed DoS Attacks”, International Journal of Network Security, Vol.7, No.1, PP.70–81, July 2008.

[3] C. Kommareddy, “Detecting DDoS Attacks in Stub-Domain”, doctoral dissertation, University of Maryland, 2006.

[4] BBC NEWS, “Google recovers after virus hits”, July 2004; http://news.bbc.co.uk/ 2/hi/technology/3927963.stm

[5] Symantec, “Symantec Internet Security Threat Report, Trends for July 06 – December 06”, March 2007.

[6] H. Wang, D. Zhang, and K. G. Shin, "Detecting SYN flooding attacks," *Proceedings of the IEEE INFOCOM’02*, 2002.

[7] R.A. Paulson and J.E. Weber, “Cyberextortion: An Overview of Distributed Denial of Service”, *Journal oOf Computer Information Systems*, Vol. VII, N°.2, 2006.

[8] J. Mirkovic, P. Reiher, “D-WARD: A Source-End Defense against Flooding Denial-of-Service Attacks*”, IEEE Transactions on Dependable and Secure Computing*, Vo.2, Issue 3, July 2005.

[9] J. Mirkovic, and P. Reiher, “A Taxonomy of DDoS Attack and DDoS Defense Mechanisms,” *ACM Special Interest Group on Data Communications (SIGCOMM) Computer Communications Review*, Vol.34 N°.2, April 2004

[10] B. Acohido and J. Swartz, “Unprotected PCs can be hijacked in minutes”, *USA Today*, 29 November 2004; http://www.usatoday.com/money/industries/technology/2004-11-29-honeypot_x.htm

[11] V. Jacobson, R. Braden, and D. Borman, “TCP Extensions for High Performance”, *IETF RFC1323*, May 1992; http://www.ietf.org/rfc/rfc1323.txt

[12] Information Sciences Institute, "Transmission Control Protocol", *IETF RFC793*, September 1981; http://www.ietf.org/rfc/rfc0793.txt

[13] W.R. Stevens, "TCP/IP Illustrated, Volume 1: The Protocols, 1st ed.", Addison Wesley, 1994.

[14] Cisco Systems Inc., "Cisco Security Advisory: Multiple Vulnerabilities in CBOS," 7 August 2001; http://www.cisco.com/en/US/products/products_security_advisory09186a00800b13cb.shtml

[15] J. Nagle, "Congestion Control in IP/TCP Internetworks", *IETF RFC896*, January 1984; http://www.ietf.org/rfc/rfc0896.txt

[16] C. Douligeris, and A. Mitrokotsa, "DDoS attacks and defense mechanisms: classification and state-of-the-art", *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 44, (5) pp. 643-666, 2004.

[17] T. M. Gil, and M. Poletto, "MULTOPS: A Data-Structure for bandwidth attack detection", *Proceeding of 10th Usenix Security Symposium*, pp 23-38, August 2001.

[18] S. Jin, and D. Yeung, "A covariance analysis model for DDoS attack detection", *IEEE International Conference on Communications* (ICC'2004), Paris, France, 20-24 June 2004.

Authors:

Vicky Laurens holds a Master's degree in Science, a Graduate Certificate in Internet Technologies, both from the University of Ottawa (Canada), and a B.S. Electronics Engineering from Simon Bolivar University (Venezuela). During her graduate studies her major research area was Network and Internet Security with special focus on the Distributed Denial of Service Attack. Currently, she is on the industry as a Vulnerability Research Specialist.

Alexandre Miège received his computer science engineer diploma from the EFREI engineering school (France) in 2000 and his M.S. from Télécom Paris in the field of IT security in 2001. He completed his Ph.D. at Télécom Paris in 2005 under supervision of Prof. Frédéric Cuppens. His research focused on access control models. He was a postdoctoral research fellow at the MCRLab of the University of Ottawa, Canada, until 2007, where he worked on network security and ID theft. Dr. Alexandre Miège is now working as an IT security expert at Savoir-faire Linux in Ottawa, Canada.

Abdulmotaleb El Saddik Professor and University of Ottawa Research Chair and Professor, and recipient of, among others, the 2008 Professional of the Year Award, the Friedrich Wilhelm-Bessel Research Award from Germany's Alexander von Humboldt Foundation (2007) the Premier's Research Excellence Award (PREA 2004). He is leading researcher in haptics aurdio visual environments, collaborative environments and ambient interactive multimedia and communications.He has authored and co-authored three books and more than 170 publications. His research has been selected for the BEST Paper Award at the "Virtual Concepts 2006" and "IEEE COPS 2007". He is an IEEE Distinguished Lecturer.

Pulak Dhar received his B.Tech degree in Electronics and Electrical Communication Engineering from the Indian Institute of Technology in 1969. He was an engineer in the Semiconductors Division of Bharat Electronics Ltd., Bangalore, India, from 1969 till 1982 working in the area of development, production and quality assurance of semiconductor devices, integrated circuits and hybrid microcircuits. In January 1983 he joined Northern Telecom Ltd in Canada as a component quality and reliability assurance engineer and was later given the responsibility as a senior engineer for software quality and reliability assurance. Since 2001 he is with Cistel Technology Inc as R&D Manager, managing research and development projects on software engineering, wireless communication and security systems for the information technology sector.