Countermeasure against Distributed Denial of Service Attack

Gaoping Hu

ABSTRACT

The threat of Distributed Denial of Service (DDoS) attack now has become a major issue in network security not only in commercial sectors but also government infra-structures. Although a lot of research has been done in this field, these attacks remain one of the most common threats affecting network performance. In this paper, the authors experimentally verify the validity of the analysis performed by running simulations using the SSFNet network simulator. A DDoS attack is simulated by flooding the mincut arcs in the network. The results indicate that the minimum number of zombie processors required to disable a set of arcs, the minimum attack traffic volume required to disable the arcs and our proposed technique will be part of an effective DDoS countermeasure.

INTRODUCTION

Denial of Service (DoS) Attacks uses multiple systems to attack one or more victim systems with the intent of denying service to legitimate users of the victim systems. The degree of automation in attack tools enables a single attacker to install their tools and control tens of thousands of compromised systems for use in attacks. Intruders often search address blocks known to contain high concentrations of vulnerable systems with high-speed connections. DoS attacks are effective because the Internet is comprised of limited and consumable resources, and Internet security is highly interdependent. Once DoS occurred in more than two places, it is called as Distributed Denial of Service (DDoS) [1] [2]. Although a lot of research has been done in this field, these attacks remain one of the most common threats affecting network performance. One defense against DDoS attacks is to make attacks infeasible for an attacker, by increasing either the amount of attack traffic needed to disable a link or the number of attackers needed to disable the network.

Theoretical work has been done previously which focused on quantifying the attack traffic required to disable a set of mincut arcs in a network. In this paper, we experimentally verify the validity of the analysis performed by running simulations using the Scalable Simulation framework (SSF) network simulator[3]. A DDoS attack is simulated by flooding the mincut arcs in the network. The results show that the minimum number of zombie processors required to disable a set of arcs and the minimum attack traffic volume required to disable the arcs.

Gaoping Hu
Information Engineering Department Communication University of China, Beijing, China
*Corresponding author: hugaoping@aliyun.com
DISTRIBUTED DENIAL OF SERVICE COUNTERMEASURES

Defense Mechanisms

DDoS defense may be regarded as a resource allocation problem in which the server resources are fairly allocated to clients to prevent attackers from consuming an excessive amount of resources. DDoS attacks can also be thwarted by filtering or rate limiting attack packets. An attack detection module is used to extract the characteristics of the attack packets and once the characteristics have been summarized, packet filtering modules are used to filter malicious packets. [4]

Some detection techniques use attack source traceback and identification as a response to a DDoS attack. The routers record information about the packets they have seen for later traceback requests or they send additional information about the packets they have seen to the packet’s destination. However, traceback is ineffective in DDoS attacks in which the attack traffic comes from legitimate sources. [5]

Activity profiling monitors the average packet rate for a network flow, which consists of consecutive packets with similar packet fields. The total network can be measured as the sum over the average packets rates of all inbound and outbound flows. An attack can be detected if an increase is observed in the network flows. [6]

In the backscatter analysis, the researchers monitor a wide IP address space for incoming backscatter packets. The backscatter packet’s source address is that of the victim, but the packet’s destination address is randomly spoofed. An attack that uses uniformly distributed address spoofing leads to a finite probability that any monitored address space will receive backscatter packets. The packets are clustered based on the unique victim source address. To detect attacks, the researchers analyze a cluster’s destination address distribution uniformity. [6]

The authors in [7] classify the DDoS defense mechanisms as being reactive and preventive. In reactive measures, the attack sources are identified as early as possible and are prevented from executing further attacks. The countermeasures here may be attack specific, when the attack is consuming fewer resources than available. The preventive measures focus on eliminating the possibility of performing a DDoS attack. This mechanism is not 100% effective but does ensure a decrease in the frequency and strength of DDoS attacks by making a host resilient to the attacks which includes identifying loopholes in the system and eliminating the vulnerabilities or removing application bugs to prevent intrusions.

Bandwidth Limited Coordination of Games

Combinatorial Game Theory

The authors in [8] set up a game between the attacker and multiple distributed applications of an enterprise. The attacker might not have sufficient resources to disrupt all the processes of an enterprise. It will try to maximize the number of processes it can disable. In reaction, an enterprise can shift to another configuration that has not been attacked. Both the players have to determine the best process to make a move in. Since the problem is P-Space complete, the authors analyze it using Combinatorial Game Theory and Thermographs. Reconfiguration strategies are provided for distributed applications using Thermostrat strategy.
The work presents an example which consists of 3 distributed applications. For determining the process in which a move has to be made, the authors make use of the Thermostrat strategy. Figure 1 shows the three distributed applications.

The authors in [9] define surreal numbers as an extension of real numbers with a tangible concept of infinity and infinitesimals. They describe surreal numbers as a pair of sets (Left and Right) of previously created surreal numbers such that no member of the right set maybe less than or equal to any member of the left set. By definition Left wins the game if the final score is greater than zero, or if the final score equals zero and it is Right’s turn to play when the game ends [10]. If every element of the left set is not less than every element of the Right set, then it results in an ill-formed surreal number, also called as a game. Every surreal number is a game, but not all games are surreal numbers.

A combinatorial game involves two players – Left and Right. A game tree has a root node which represents the initial position. The root node has zero or more branches going downwards to the left (representing moves for the left player) and downwards to the right (representing moves for the right player). At each point, the player considers the options he has and chooses the one which will maximize his payoff value. Game trees can be typically represented as shown in equation (1).

\[
\{L_1, L_2, \ldots, L_n | R_1, R_2, \ldots, R_m \} 
\]  

(1)

The options for left are represented as \(L_1, L_2, \ldots, L_n\) and the options for right are options from \(R_1, R_2, \ldots, R_m\). The equation has a numeric value if

\[ \forall L \forall R_j : L < R_j \]  

(2)

The value of a surreal number where equation (2) holds is the “simplest” number between the greatest L value (Lmax) and the smallest R value (Rmin) [11]. If equation (2) is not satisfied, then the number is ill formed and it is a game.

\[ \exists L \exists R_j : L \geq R_j \]  

(3)

The value of the game then depends on the sequence of moves taken. Figure 2 shows a diagram of a game tree which can be represented by equation (4)

\[
G = \{15, 25\} \{10\} \{-5\} 
\]  

(4)
Combinatorial Game Theory

A combinatorial game involves two players – Left and Right. These are perfect information games in which all players know all the moves that have taken place. Combinatorial game theory does not study games of chance. In our example scenario, there are multiple networks which want to coordinate and communicate over the network links. The Blue player needs to prioritize data and send the most important information. This effectively translates into a Sum of Games problem, where the Blue player is engaged in multiple games with Red and the aim is to maximize the overall payoff function. This sum of games is represented by

\[ G = \sum_{j=1}^{n} G_{i,j} \] (5)

Yedwab’s theorems in [10] state that a truly optimal strategy for a sum of games is only found by an exhaustive search of alternatives which requires exponential time. Instead of finding the best possible solution, it is possible to find a solution within a constant offset of the optimal.

We introduce a concept called thermographs which could be used for chilling the games and finding the optimal strategies for the sum of games. We use the concept of thermographs in calculating the value of a game. Thermographs are plotted on graphs in which the co-ordinate system used has the tax on the y-axis and the game value on the x-axis. The values on the x-axis are plotted in decreasing order to keep the Left player’s options to the left side of the graph. As tax \( t \) increases, both sides reach a common value which is called as the ‘mean value’ of the game. The smallest tax needed to reach the game’s mean value is called as the temperature of the game.

Berlekamp’s Strategies

Choosing a strategy to play the sum of games problem would help to make a decision. In [12], Berlekamp’s Hotstrat strategy recommends play in a game which has the highest temperature. In other words, the Hotstrat strategy when applied to a sum of games problem would choose a game with the highest variability. Since the variability directly relates to the payoff values, this strategy correctly reflects the most important component game.

The Hotstrat strategy [13] when applied to a sum of games problem would choose a game with the highest temperature and will correspondingly choose that game. Since the temperature of the game signifies the importance and variability of the game, the higher the variability of the game, the higher is the payoff that can be obtained by playing that game. In this example scenario, the result would give us the most important message that needs to be transmitted. This ensures that
communication is maintained till the affected links are restored back to their normal state.

Example Game

We have multiple departments which need to coordinate in order to maintain communication. If the links between the departments experience a DDoS attack, there would be a heavy constraint on the bandwidth that can be assigned to the players. This limits the number of messages that could be transmitted. At such times, it would be of paramount importance to prioritize the messages that need to be sent. We model this as a two player game with the two players being Red and Blue. The aim of the Blue player is to maintain communication within its departments and the aim of the Red player is to try and disrupt it. If the attacker is able to disrupt the communication between the departments then the Red player wins.

Multiple departments share a limited communications channel and more than one department can simultaneously detect changes in the network. The player needs to prioritize messages before deciding which is the most important. This problem was solved by Virtenen in [14] by considering Maximax, Maximin and central values prioritization schemes. We modify the problem to compare game trees instead of comparing range of values. We represent our set of messages as different branches of a game tree with payoff values assigned to each branch. In essence, m departments are simultaneously deciding which of the n attackers to engage (one attacker might target multiple links). Thus the message prioritization problem is changed from a team decision problem to a Sum of Games problem from combinatorial game theory.

Playing in a game with one of the options masked

Since more than one department can simultaneously detect changes in the network, a subset of the game changes, however because of bandwidth limitations, the players can not accurately know the details of all the games in the set. They need to choose the games which are more important. So the players end up playing in a sum of games problem where they are ignorant about the payoffs in a subset of the games.

Incorporating chance moves

In order to deal with combinatorial game theory, we need to modify Conway’s surreal number approach to include chance moves. Surreal number representations of the game assume perfect information. Unfortunately, the underlying nature of this problem is probabilistic in nature. Figure 3 shows an extensive form representation of a chance move. Extensive form is a tree structure with each interior node of the tree representing a decision point. Leaves are associated with payoffs. At the root node, Blue wants to maximize the payoff. If Blue chooses the alternative on the left, two choices exist on the left with probability 0.4 and right with probability 0.6. After those chance moves are nodes that represent Red’s choices. Since Red wants to minimize, the left (right) node has value 5 (14). We state a theorem which helps us solve this problem.
Figure 3. Extensive form representation of a chance move.

**Theorem 1.** Given a probability distribution function \( \{p_1, p_2, ..., p_i\} \) where \( p_k \) is the probability of surreal number \( \{L_R | R_k\} \), the expected value is a surreal number.

Since addition and multiplication of well formed surreal numbers is a surreal number \([9]\), for all elements in a game tree,
\[
p_k \cdot \text{surreal number} = \text{surreal number}
\] (6)

Blue uses the expected value of its left node \(0.4 \times 5 + 0.6 \times 14 = 10.4\) as its expected payoff in calculating which alternative to take. It can also be viewed as compressing the two Red moves into a single information set where Blue cannot know in advance which node in the information set it chooses. In extensive form, each player’s possible moves are expressed in alternation with chance moves inserted as necessary. By replacing chance moves in a game tree, we convert an imperfect information game into an equivalent perfect knowledge game. The next section talks about the algorithm to prioritize the messages.

**Message Prioritization Algorithm**

1. Each network monitors the state of its links.
2. Each network constructs game trees based on the monitored data.
3. Surreal numbers are constructed for each engagement.
4. Thermographs are constructed from each surreal number and the freezing point is noted.
5. The data is prioritized using the temperature of its associated surreal number.
6. An alarm is set proportional to the inverse of the temperature.
7. As soon as the alarm expires, if the bandwidth is not occupied then data is transmitted.

**SIMULATION OF GAME SCENARIO**

The simulation game scenario developed using Python helps explain the example game. We also show that the Hotstrat strategy dominates the strategies used in \([14]\) for game theory problems. Both the players – Red and Blue start the game with a common operating view. The common operating view is a set of three randomly generated games \((G_1, G_2, G_3)\). The two players compete by playing a sum of games problem on this set. On monitoring their links, the players determine that the games \(G_2\) and \(G_3\) are replaced by games \(G_4\) and \(G_5\). Since the players have bandwidth enough to transmit information of only one game, they have to choose between game \(G_4\) and \(G_5\). The decision about which player makes the first move is made randomly with both players having an equal probability. Both players are now playing a sum of games problem which consists of \(G_1, G_4,\) and \(G_5\). However, the players are forced to choose between the information sets of \(\{G_1, G_2, G_3\}\) and \(\{G_1, G_4, G_3\}\). The players choose a strategy from Maximax, Maximin, Central
values and Hotstrat to help them decide which game to play in and make a move in that game. This procedure is followed until payoff values are obtained for all three games. The payoff values corresponding to the real scenario are summed to give the payoff for the sum of games. If the sum is greater than or equal to half the maximum payoff possible, then Blue player wins, else Red wins.

The five randomly generated games are

\[ G_1 = \{12|32|33|32|48|9|43|9\} \]  
(7)

\[ G_2 = \{4|9|49|34|24|1|9|4\} \]  
(8)

\[ G_3 = \{6|37|49|2|10|34|21|18\} \]  
(9)

\[ G_4 = \{10|2|4|45|32|39|32|34\} \]  
(10)

\[ G_5 = \{15|5|14|43|13|27|3|27\} \]  
(11)

The steps taken to choose the game which it prefers to see are listed in Table 1. The following conventions are used to describe the simulation example.

- \( S \): Set of games which the player sees and applies the strategy to.
- \( I_g \): Game which is inconsistent with the real scenario.
- \( c_g \): Game chosen to be modified after applying \( x \).

**Table 1.** Procedure to decide the starting scenario.

| Time step | Action performed by Blue | Action performed by Red |
|-----------|--------------------------|-------------------------|
| 1         | \( S = \{G_1, G_2, G_3\} \) \( x = \text{Maximax} \) | \( S = \{G_1, G_2, G_3\} \) \( x = \text{Maximax} \) |
| 2         | Apply \( x \) to \( G_2 \) and \( G_3 \) \( c_g = G_3 \) | Apply \( x \) to \( G_2 \) and \( G_3 \) \( c_g = G_3 \) |

\( I_g = G_5 \)

\( S = \{G_1, G_2, G_3\} \)

Table 2 details the steps followed after choosing the inconsistent game. In our simulation run, Blue player starts the game.

**Table 2.** Steps to play the game.

| Time step | In action | Scenario before \( S \) | Action | Scenario after \( S \) |
|-----------|-----------|-------------------------|--------|------------------------|
| 1 Blue    | \( G_1 = \{12|32|33|32|48|9|43|9\} \) \( G_2 = \{10|2|4|45|32|39|32|34\} \) \( G_3 = \{6|37|49|2|10|34|21|18\} \) \( c_g = G_j \) | Apply \( x \) to \( S \) \( G_1 = \{12|32|33|32|48|9|43|9\} \) \( G_2 = \{10|2|4|45|32|39|32|34\} \) \( G_3 = \{6|37|49|2\} \) |
| 2 Red     | \( G_4 = \{12|32|33|32|48|9|43|9\} \) \( G_2 = \{10|2|4|45|32|39|32|34\} \) \( G_3 = \{6|37|49|2\} \) | Apply \( x \) to \( S \) \( c_g = G_d \) | \( G_1 = \{12|32|33|32|48|9|43|9\} \) \( G_2 = \{32|39|32|34\} \) \( G_3 = \{6|37|49|2\} \) |
| 3 Blue    | \( G_4 = \{12|32|33|32|48|9|43|9\} \) \( G_2 = \{32|39|32|34\} \) \( G_3 = \{6|37|49|2\} \) | Apply \( x \) to \( S \) \( c_g = G_j \) | \( G_1 = \{12|32|33|32|48|9|43|9\} \) \( G_2 = \{32|39|32|34\} \) \( G_3 = \{6|37\} \) |
At time step 4, a final payoff value is obtained for $G_3$. The procedure is continued until payoff values are obtained for all the games. Since the payoff values corresponding to the real scenario are considered, the payoff value in this case is the summation of the payoff values for games $G_1$, $G_4$, and $G_5$ which is 69 (32 for $G_1$ + 32 for $G_4$ + 5 for $G_5$).

The simulations are run 500 times for each pair of strategies. The percentage wins for Blue player are recorded. The rows represent the strategies chosen by Blue and the columns represent the strategies chosen by Red.

**Table 3.** Recorded percentage wins for Blue.

|        | MAX   | MIN   | Central Values | Hotstrat |
|--------|-------|-------|----------------|----------|
| MAX    | 0.53  | 0.518 | 0.55           | 0.27     |
| MIN    | 0.492 | 0.548 | 0.488          | 0.354    |
| Central Values | 0.564 | 0.538 | 0.534          | 0.362    |
| Hotstrat | 0.75  | 0.71  | 0.718          | 0.542    |

The test for statistical significance between binomial distributions is used to verify that the values in Table 3 are significantly different.

$$\log \left( \frac{p_1(1-p_2)}{p_2(1-p_1)} \right) < 0.41$$  \hspace{1cm} (12)

Row wise and column wise comparisons are performed to determine the most optimal strategy for Red and Blue.

For a Red strategy, the most optimal strategy for Blue can be determined by comparing values within columns. This is shown in Table 4. Sub columns are created within columns to show which strategies are being compared. We note that Hotstrat’s performance is significantly better than the other three, no matter which strategy was chosen by Red. So Hotstrat is marked as + and the others are marked as -. When there is no significant difference between the strategies, then they are marked as $\approx$.

On comparing within rows, we obtain the strategy that performs best for Red against a given Blue strategy. Similarly, Hotstrat causes Blue to win fewer games than other strategies. This is shown in Table 5. Thus the Hotstrat provides an effective strategy for determining the priority of the games when competing for bandwidth.

**Table 4.** Choosing an optimal strategy for Blue.

|        | MAX   | MIN   | Central Values | Hotstrat |
|--------|-------|-------|----------------|----------|
| MAX    | - $\approx$ | - $\approx$ | - $\approx$ | - $\approx$ |
| MIN    | - $\approx$ | - $\approx$ | - $\approx$ | - $\approx$ |
| Central Values | - $\approx$ | - $\approx$ | - $\approx$ | + |
| Hotstrat | +  | +  | +  | +  |
Table 5. Choosing an optimal strategy for Red.

|        | MAX | MIN | Central Values | Hotstrat |
|--------|-----|-----|----------------|----------|
| MAX    | +   | +   | +             | -        |
| ≈      | ≈   | ≈   | ≈             |          |
| MIN    | +   | +   | +             | -        |
| ≈      | ≈   | ≈   | ≈             |          |
| Central Values | +   | +   | +             | -        |
| ≈      | ≈   | ≈   | ≈             |          |
| Hotstrat | +   | +   | +             | -        |
| ≈      | ≈   | ≈   | ≈             |          |

CONCLUSIONS

In this paper, we verified, by performing simulations, the work in [8] to quantify the number of resources that an attacker would need to disable a network. Performing a DDoS on a large scale network is more reasonable than a DDoS on a small scale network. We choose the SSFNet simulator over its competitors as it is capable of handling large networks. To simplify the tedious and error prone process of writing script for large networks, we automate the network generation process.

We develop an alternative application of combinatorial game theory in which we allocate bandwidth between processes. We present an example scenario by setting up a game between an attacker and multiple distributed applications of an enterprise. The enterprise coordinates between its different networks by maintaining communication over bandwidth communication links. The limited bandwidth links make it necessary to determine the most important message that an enterprise needs to transmit.

In order to account for the probabilistic nature of the problem, we convert a game with imperfect information into perfect information games. We compare four strategies – Maximin, Maximax, Central value and Hotstrat to determine the priority of the messages and conclude that Hotstrat gives us the best possible results. We verify our understanding by running simulations. The results indicate that our proposed technique will be part of an effective DDoS countermeasure.

REFERENCES

[1] T. Pardoe and G. Snyder, Network Security, Thomson Delmar Learning. 2005, pp.xiv.
[2] J. Mirkovic, J. Martin, and P. Reiher, “A taxonomy of DDoS attacks and DDoS defense mechanisms,” in ACM CCR, April 2004.
[3] J. Guo, W. Xiang, S. Wang, Reinforce Networking Theory with OPNET Simulation, Journal of Information technology Education, Volume 6, 2007.
[4] [R. Chen, J. Park, and R. Marchany, “A Divide-and-Conquer Strategy for Thwarting Distributed Denial-of-Service Attacks,” IEEE Transactions on Parallel and Distributed Systems, vol 18, issue 5, May 2007, pp 577-588D.
[5] R. Chang, “Defending against flooding-based, distributed denial-of-service attacks: a tutorial,” IEEE Communications Magazine, 40(10), 2002.
[6] G. Carl, G. Kesidis, R. Brooks and S. Rai, “Denial-of-Service Attack-Detection Techniques,” IEEE Internet Computing, vol. 10, nos. 1, pp. 82-89, Jan. 2006.
[7] Kevin, J., Weaver, G. M., Long, N., and Thomas, R, “Trends in denial of service attack technology,” Technical report, CERT Coordination Center, Carnegie Mellon University, October, 2001.
[8] C. Dingankar, Enterprise Security Analysis Including Denial of Service Countermeasures, Clemson University, August 2007.
[9] C. Tondering, “Surreal Numbers - An Introduction,” Version 1.5, January 2005 (online). Available: http://www.tondering.dk/claus/sur15.pdf. Last accessed 12/01/2009
[10] L. J. Yedwab, “On playing well in a sum of games,” M.S. Thesis, MIT, 1985, MIT/LCS/TR-348.
[11] J. H. Conway, On Numbers and Games, AK Peters, LTD, 2000, Ch 2.
[12] E. R. Berlekamp, J. H. Conway, and R. K. Guy, Winning Ways for your mathematical plays Volume 1: Games in General, Academic Press, New York, 1982, pp.383-391.
[13] B. C. A. Milvang-Jensen, “Combinatorial Games, Theory and Applications,” Thesis, IT University of Copenhagen, 2000.
[14] K. Virtanen, R. P. Hamalainen, and V. Mattila, “Team Optimal Signaling Strategies in Air Combat,” IEEE Transactions on Systems, Man, and Cybernetics, Part A, vol. 36, no. 4, pp. 643-660, July, 2006.