Game Theory-based Defense Mechanisms against DDoS Attacks on TCP/TCP-friendly Flows

Harkeerat Singh Bedi, Sankardas Roy, Sajjan Shiva
Department of Computer Science
University of Memphis
Memphis, TN, USA

Abstract— **While there are significant advances in information technology and infrastructure which offer new opportunities, cyberspace is still far from completely secured. In many cases, the employed security solutions are ad hoc and lack a quantitative decision framework. To this end, game theory poses huge potential in building a defense architecture based on a solid analytical setting. In this paper, we explore the applicability of game theoretic approaches to the cyber security problem while keeping the focus on active bandwidth depletion attacks on TCP/TCP-friendly flows. We model the interaction between the attacker and the defender as a game in two attack scenarios: (i) one single attacking node for Denial of Service (DoS) and (ii) multiple attacking nodes for Distributed DoS (DDoS). The defender's challenge is to determine optimal firewall settings to block rogue traffic while allowing legitimate ones. Our analysis considers the worst-case scenario where the attacker also attempts to find the most effective sending rate or botnet size. In either case, we build a static game model to compute the Nash equilibrium that represents the best strategy for the defender. We validate the effectiveness of our game theoretic defense mechanisms via extensive simulation.**

Keywords – Denial of Service (DoS); Distributed DoS; TCP-friendly flows; Game Theory; Simulation.

I. INTRODUCTION

The Nation's economic progress and social well-being are becoming increasingly dependent on cyberspace. At the same time, the growing inter-connectivity and the increasing availability of the computational power for the attacker is providing for distributed and sophisticated attacks [3]. The research and practicing community have paid attention to the cyber security problem for more than two decades. However, the problem remains mostly unsolved. The core security breaches occur in terms of confidentiality, integrity, and availability.

The main limitation of the current cyber security practice is that the security approach is largely heuristic, increasingly cumbersome, and is struggling to keep pace with rapidly evolving threats. Most of the current security approaches lack a quantitative decision framework. As game theory deals with problems in which multiple players with contradictory objectives compete with each other, it can provide a mathematical framework for modeling and analyzing network security problems. As an example, a system administrator (defender) and an attacker can be viewed as two competing players participating in a game. Recently, researchers have started exploring the applicability of game theory to address this problem [11].

In this paper, we focus on bandwidth depletion attacks for Denial of Service (DoS) or Distributed DoS (DDoS) where a single attacking node or multiple attacking nodes attempt to break down one or more network links by exhausting limited bandwidth. We consider the interaction between the attacker[1] and the defender (network administrator) as a two player game and apply game theory-based countermeasures. For each of DoS and DDoS cases, we design a game model. The attacker attempts to find the most effective sending rate or botnet size while the defender's challenge is to determine optimal firewall settings to either block or redirect rogue traffic while allowing legitimate ones. We study the existence of the Nash equilibrium, which represents the best strategy of each player. We show the benefit of using game theoretic defense mechanisms for the network administrator.

We focus on a widely practiced type of denial of service attacks which are launched by creating congestion on TCP or "TCP-friendly" flows. *TCP-friendly* flows are the type of flows which adhere to the TCP ideology of "fair" bandwidth sharing of a congested bottleneck link. One definition of "fair" is that of TCP "friendliness" [9, 5] – if a non-TCP connection shares a bottleneck link with TCP connections, traveling over the same network path, then the non-TCP connection should receive the same share of bandwidth (i.e., achieve the same throughput) as a TCP connection. In the rest of the paper, for ease of exposition, 'TCP-friendly' flows refer to TCP/TCP-friendly flows.

II. RELATED WORK

The research community has been actively studying the Denial of Service (DoS) attacks over the last two decades. There are many significant contributions made to mitigate a variety of these attacks [6]. The key of DoS/DDoS defense approaches is to identify malicious nodes and restrict their packet injection from the source or drop unwanted packets at intermediate routers before they reach the destination. Lau et al. [4] experimented with various queuing algorithms to determine which queuing method in the target router could provide better management of the bandwidth during a DDoS attack.

Chertov et al. [10] emphasized that DoS is not only caused by flooding but also by exploiting the congestion window of TCP protocol used in the communication between the server and the client. The experiments were based on the assumption

[1]We assume that one single attacker controls all of the attacking nodes present in a botnet setup for DDoS.

that length of the attack pulse controls the tradeoff between attack damage and attack stealthiness. During the congestion avoidance phase, when packet losses occur TCP halves its congestion window, which is the property exploited. In our model we take into account the rate of data sent by the attacker as the degree of maliciousness of the attacker. Moreover, our model is not limited to only TCP flows. Our model also works with non-TCP flows as long as they TCP-friendly in nature.

Andersen [2] proposed a proactive protection against DDoS attacks, by imposing overhead on all transactions to actively prevent attacks from reaching the server. Their architecture generalizes the Secure Overlay Services (SOS) to choose a particular overlay routing. The set of overlay nodes are used to distinguish legitimate traffic from the attack traffic.

Yaar et al. [1] proposed a flow based mitigation filter for DDoS flooding attacks. Stateless Internet Flow Filter (SIFF) based approach uses a per-flow state, where the flows are classified into two categories privileged flows, and unprivileged flows with the goal of protecting privileged packets from unprivileged packet flows.

Recently, the research community has designed efficient honeypot systems to better understand the novel attack techniques and attacker's strategies such as Potemkin [8] and Collapsar [12].

Game theory has been applied in various application domains and is attracting more attention from network researchers for cyber security. Xu et al. [7] proposed a game-theoretic model to defend a web service under DoS attack. They used a single bottleneck link to simulate the attacks. The metrics used for the performance of their system are total throughput of the attackers and their legitimate clients, legitimate client's average amount of time to download a web page, number of concurrent attackers and clients, and packet drop probability of the attackers and the clients.

Wu et al. [13] perform similar research where they primarily focus on DoS/DDoS attacks launched using UDP based traffic. In their model, the actions possible by the defender are either to allow or drop incoming traffic. Our model extends this by also providing defender with the ability to redirect traffic to honeypot to learn more about the attacker.

Our work focuses on mitigating DoS and DDoS attacks for TCP-friendly flows using a game theoretic approach.

III. PRELIMINARIES:

A. Network Topology

The real world scenario under consideration consists of legitimate nodes attempting to transfer data to the target server. The target server is intended to process incoming requests and accept the uploaded data. Our network topology broadly consists of attack nodes, legitimate nodes, a target server and a honeypot. The attacker nodes are malicious in nature and intend to send unreasonable uploads to the target server in order to seize most of the available bandwidth, thus resulting in a DoS/DDoS attack for the legitimate user uploads. One kind of unreasonable request can include uploading the same file multiple times and simultaneously.

For convenience, we list all the notations and abbreviations used in this paper in Table I.

TABLE I. NOTATIONS USED THROUGHOUT THIS PAPER

Symbol	Definition
TS	Target Server
PR	Perimeter Router
FW	Firewall
GW	Gateway
HP	Honeypot
P_1	Network pipe end connected to interface i_2
P_2	Network pipe end connected to target server TS
B	Total bandwidth of the pipe (P_1P_2) between i_2 and TS.
n	Number of legitimate nodes
m	Number of attacking nodes
u	Number of flows per attacking node
t	Total number of nodes
S	Total number of legitimate flows.
s_i	Number of flows created by i^{th} legitimate user.
s_l	Mean value of s_i
σ_1	Standard deviation of s_i
d	Bandwidth consumed per node, which is B/t.
r_l	Bit rate of a legitimate flow
σ_2	Standard deviation of a legitimate flow rate
r_a	Bit rate of an attack flow
γ	Minimum bit rate for a flow to be considered alive
γ_l	Mean value of a legitimate flow's threshold

The network topology which we focus on for analyzing DoS/DDoS attacks and its countermeasures is shown in Fig. 1. Target Server TS is accessible to the Internet through a Gateway machine GW which runs our proposed Game Inspired Defense Architecture (GIDA) Module. This figure also illustrates the flow of execution of our network model. The traffic flow coming from the Internet destined for the target server TS first interacts with the perimeter router PR of the network, and then enters the Gateway (which runs the GIDA Module) at interface i_1.

The Gateway is an isolated machine which consists of three interfaces where i_1 is connected with the Internet via the perimeter router PR and i_2 and i_3 are connected with the Target Server TS and Honeypot HP respectively. The interface i_1 is monitored by our GIDA Module which implements our proposed defense architecture. Based on the decisions computed by the GIDA Module, the traffic can be allowed to go through interface i_2 and reach the target server TS, or redirected to a honeypot via i_3 or dropped entirely. The honeypot HP is primarily used for analyzing traffic and learning further information from the attacker.

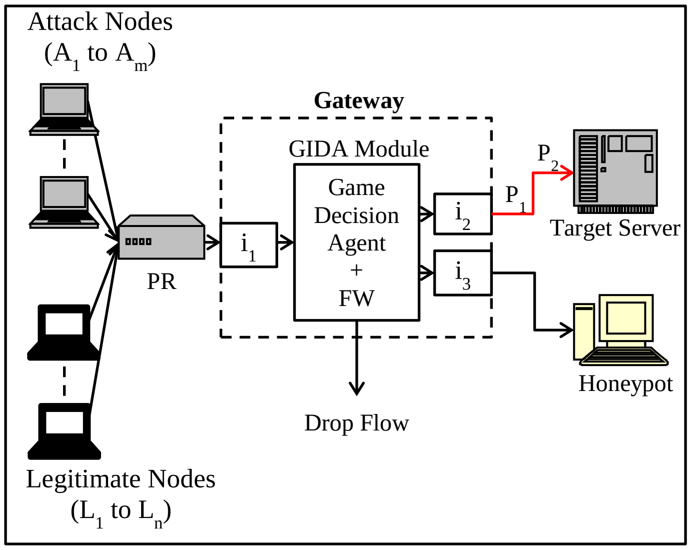

Figure 1.The Network Topology

B. Problem Statement

There are n legitimate nodes that need to communicate with the server TS, and also, there is one attacker A who is interested in launching a denial of service attack over the network pipe (P_1, P_2) which connects the gateway GW with the target server TS. The attacker aims in do so by consuming most of the bandwidth of the pipe (P_1, P_2). The attacker A controls m attacking nodes that can perform arbitrary uploads to exploit the limited bandwidth. It can be noted that DoS attack is a special case of DDoS attack when $m = 1$.

Nodes labeled L_1 to L_n are legitimate nodes which send request to the target server TS and upload the requested data. Nodes labeled A_1 to A_m are the attack nodes which simultaneously send a large number of requests to the target server TS for uploading data and limiting network bandwidth for the legitimate nodes. High usage of limited network bandwidth over the pipe (P_1, P_2) by the attacker nodes results in a denial of service scenario for the legitimate nodes. Hence, securing this pipe (P_1, P_2) against such attacks is our goal. The defender's controls are present in the GIDA Module which consists of the Game Decision Agent and the firewall.

C. Assumptions

We would like to point out that our model is not network-specific and is readily applicable to any DoS/DDoS scenarios in a general network topology with the following assumptions:

- A single attacker controls all of the attacking nodes, each of which sends arbitrary packets and file upload requests to the server TS in order to misuse limited network bandwidth.
- There is an infinitely high bandwidth available on the channel between PR, the interface i_1, and GIDA Module is able to process all of the incoming packets.
- The attacker does not spoof a unique source address for each packet in a single flow. Such spoofing would be extremely difficult and is highly unlikely to occur. Note that when the spoofed source address is the same for the entire flow, the filtering mechanism would act the same as if there were no spoofing.
- Furthermore, it is conservatively assumed that the attacker has clever tools to detect whether his flows are being redirected to the honeypot.
- We assume that all flows from all nodes (legitimate and attack) are "TCP-friendly" in nature.

D. Main Idea

We envision GIDA as a security model which aims to provide protection for target systems against attacks by computing and performing preventive and defensive strategies using game theoretic concepts. We consider the interaction between the attacker and the defender (network administrator) as a game played among them and aim to apply game theory-based countermeasures.

The GIDA Module is the decision module which analyzes the incoming flow and restricts or provides access to the target server based on its computed decisions which in turn are based on certain properties of the incoming flow. We envision that this GIDA Module can be integrated as part of the target system to be protected or can be an implemented as a standalone arrangement. In this work, we consider the latter. The GIDA Module primarily consists of two major components which are a Game Decision Agent and a firewall. The Game Decision Agent performs the game theoretic analysis on incoming flows and computes the appropriate defensive decisions which are then implemented using the firewall. Decisions taken by the GIDA Module on incoming flow encompass the actions possible by the defender to prevent attacks and protect the target server TS.

In our present implementation, the GIDA Module decisions:

- Allow traffic to flow to the target server TS as normal.
- Drop traffic at the firewall to prevent it from reaching the target server TS.
- Redirect traffic to the honeypot HP for continuation of the attack as intended by the attacker. This step is crucial for further analysis, which can be helpful for learning more about the kind of attack and the intention of the attacker.

The actions possible by the attacker consist of the following:

- Deploy an arbitrary number of attacker nodes that send file upload requests the target server TS and consume the limited available bandwidth resources by performing multiple uploads simultaneously.
- Adjust the rate of file uploads from each attack node during an attack instance.

Any of the above actions performed by the attacker yields a certain amount cost and benefit to it. Use of less number of nodes but with high request frequency yields a lower cost to the attacker as he has to employ fewer number of nodes. However this increases the probability of those attack flows being redirected or dropped because now the attacker is required to send more traffic from each attack node to consume a significant amount of bandwidth of the target link. Use of additional attack nodes with a lower frequency increases the cost of the attacker due to the high number of nodes the attacker has to employ, however the benefit also increases as since the frequency of requests is moderate, the attack becomes less obvious.

IV. Game Model

In this section, we present our game models for DoS/DDoS attacks and their possible countermeasures. We consider the interaction between the attacker and the defender (network administrator) as a two-player game. We study the existence of equilibrium in these games and also show the benefit of using the game-theoretic defense mechanisms.

The attacker attempts to find the most effective request sending rate or botnet size to maximize his utilization of the limited network bandwidth, and the defender's challenge is to determine the best firewall settings to block and redirect rogue traffic while allowing legitimate ones. We first discuss some basic concepts of game theory and the profile of legitimate nodes, and then construct our game models.

A. Basic Concepts of Game Theory

In a *game*, each player chooses actions that result in the best possible rewards for self, while anticipating the rational actions from other players. A *strategy* for a player is a complete plan of actions in all possible situations throughout the game. Nash equilibrium is a solution concept that describes a steady state condition of the game; no player would prefer to change his/her strategy as that would lower his/her payoffs given that all other players are adhering to the prescribed strategy. A *static game* is a one-shot game in which each player chooses his/her plan of actions and all players' decisions are made simultaneously. A *dynamic game* is a game with multiple stages in which each player can consider his/her plan of actions not only at the beginning of the game but also at any point of time in which they have to make a decision.

B. Legitimate User Profile

We consider the presence of n legitimate nodes interested to communicate with the target server TS. Each user uses one or more TCP-friendly flows to do his/her work. We also consider that the number of flows used by a user (which originate from the same physical machine) follow a normal distribution: $s_i\colon \mathcal{N}(s_l, \sigma_1^2)$, $i = 1,2,\ldots,n$; where s_i represents the number of flows of the i^{th} user, s_l is the mean value of a legitimate user's number of flows , and σ_1 is the standard deviation. If one has information that s_i follows a different distribution, then the above normal distribution can be replaced by this different distribution and our analysis will still remain the same for the rest. For example, if one were required to speculate the number of flows originating from a user over a course of time, then following a Poisson distribution may be more appropriate.

The available bandwidth of a legitimate flow at the absence of any attack is $r^{na} = \frac{\mathrm{B}}{\mathrm{S}}$, where $\mathrm{S} = \mathrm{s}_1 + \mathrm{s}_2 \ldots + \mathrm{s}_n$. This observation for the bandwidth available per flow holds true because we assume that all flows passing through the pipe (P_1, P_2) are TCP-friendly in nature. By basic laws of probability, we get $\mathrm{S}\colon \mathcal{N}(n \cdot \mathrm{s}_l, n \cdot \sigma_1^2)$, where B is the bandwidth of the pipe $(P_1 P_2)$ between the interface i_2 and the target server TS.

There is threshold bandwidth associated with a flow such that if at any point of time the bandwidth available to a flow is less than its threshold then that flow is considered as dead. This threshold for each flow depends on the specific application the flow is intended for, and is not related to the TCP protocol or its optional keepalive feature. We assume a flow as being dead if it does not meet this minimum threshold bandwith which is dependent on the application.

We consider the threshold bandwidth of a legitimate flow as a random variable. In particular, we model the threshold bandwidth via a normal distribution. That means $\Gamma_i\colon \mathcal{N}(\gamma_l, \sigma_2^2)$, $i = 1, 2, \ldots, S$; where Γ_i represents the threshold bandwidth of the i^{th} flow, γ_l is the mean value of a legitimate flow's threshold, and σ_2 is the standard deviation.

The probability that i^{th} legitimate flow will terminate in the absence of any attack is $P[r^{na} < \gamma_i]$; where γ_i represents the instantaneous value of Γ_i and $P[x < X]$ represents the probability that the value of the random variable X is greater than x. We assume that the pipe bandwidth B is chosen such that $P[r^{na} < \gamma_i]$ is negligible in attack-free scenarios. We also assume that no two nodes connect to the server from the same physical machine. However, a user can send more than one flows from the same physical machine.

We now present our static game model. We assume that one single attacker controls all of the attacking nodes. There is only one attacking node in a DoS attack, while there are multiple attacking nodes in a DDoS attack. Our discussion is generic with respect to DoS or DDoS attacks, which considers that number of attacking nodes is m. If we replace m by 1, we get the DoS scenario.

C. A Static Game

A static game is a one-shot game, i.e, once a player decides his strategy he does not have a second chance to change it. We consider the attacker's reward is not necessarily the defender's cost, i.e. it could be a zero-sum or non-zero sum game. The actions available to the attacker are to set the number of attack flows u from a single machine and to select the number of attacking nodes, m.

Each flow in our model is identified by a pair of source and destination IP addresses. We do not take the source and destination ports into consideration for identifying a flow, since, an attacker can open multiple connections with the target server TS and they all contribute to the amount of bandwidth used by the attacker in total. Therefore in our model, each user flow is characterized by the amount of bandwidth used between one IP source/destination address pair.

We assume that the attacker only uses TCP-friendly flows to send its traffic and the attacker cannot cheat with TCP protocol parameters such as the congestion window size, packet sequence or acknowledgement numbers.

It is assumed that TCP protocol will ensure that each flow gets an equal share of the bandwidth of the pipe (P_1, P_2), i.e., the bit rate is same for all of the flows (legitimate or attack flows), which is represented by r. We also assume that the number of attack flows, u is same for all the attack nodes. In an attack situation each flow rate, $r = \frac{B}{S + m \cdot u}$. If r is small, then we consider that the denial of service occurs due to congestion in pipe (P_1, P_2). In particular, severe situation happens when $r < \gamma_i$, and some of the legitimate flows die out, where γ_i be the minimum bit rate for the i^{th} flow to be considered as active.

1) Impact of the Attack with no Defense Mechanism:

In *no defence* situation, all the flows pass through the firewall to the target server TS. However, if r is small, then we consider that the denial of service occurs due to congestion in pipe (P_1, P_2). That means, the bandwidth available to a legitimate flow is reduced which results in more latency for legitimate data transmission. In particular, severe situation happens when $r < \gamma_l$, and some of the legitimate flows are exhausted.

Let S_g be the average number of legitimate flows which are able to reach the server and whose rate is greater than γ_l. We get $S_g = S \cdot P[\gamma_l < \mathrm{r}]$. Hence, we get the following results.

Average bandwidth consumption (by the attacker) ratio:

$$v_b^{nd} = \frac{m \cdot u}{S + m \cdot u} \qquad (1)$$

As there are S legitimate flows and $m \cdot u$ attack flows which equally divide the bandwidth we get equation (1).

Ratio of lost legitimate flows to the total number of legitimate flows on average:

$$\begin{aligned} v_n^{nd} &= \frac{S - S_g}{S} \\ &= 1 - P[\gamma_l < \mathrm{r}] \\ &= P[\gamma_l > \mathrm{r}] \end{aligned} \qquad (2)$$

The attacker's objective is to increase v_b^{nd} and v_n^{nd}, which he considers as his rewards. On the other hand, we assume that the attacker has to incur some cost to get control of an attacking node. We assume that the attacker's cost (v_c) is proportional to the number of attacking nodes employed and $v_c = m$. We model the attacker's net payoff as a weighted sum of the above three quantities given as

$$V^a = w_b^a \cdot v_b^{nd} + w_n^a \cdot v_n^{nd} - w_c^a \cdot v_c \qquad (3)$$

where w_b^a, w_n^a, and w_c^a are the attacker's corresponding weight parameters.

On the other hand, we model the defender's net payoff as a weighted sum given as

$$V^d = -w_b^d \cdot v_b^{nd} - w_n^d \cdot v_n^{nd} + w_c^d \cdot v_c \qquad (4)$$

where w_b^d, w_n^d and w_c^d are the defender's weight parameters.

2) Defending Attacks with GIDA Module:

As discussed previously, the actions possible by the defender consists of allowing the traffic to the target server TS, redirecting them to honeypot HP or dropping the same.

The defender selects two thresholds E_1 and E_2 for deciding his actions on an incoming flow. He begins by computing the total flow rate $r \cdot u$ for a source node, where u is the number of flows for that node and r is the bit-rate per flow. If the defender observes that from a particular source node K, the total flow rate is $r \cdot u < E_2$, then the firewall allows these set of flows to reach the target server TS. On the other hand, if $r \cdot u > E_2$ and $r \cdot u < E_1$ then all the flows from this source node K are redirected to the honeypot HP. Finally, if $r \cdot u > E_1$ then all the flows from this source node K are dropped by the firewall. It should be noted that these decisions are probabilistic in nature, which signifies that even if $r \cdot u < E_2$, there is a minute likelihood that some flows from source node K may be dropped. This is the same for all other cases as well.

These thresholds E_1 and E_2 are used for creating two sigmoid filters, namely F_1 and F_2, which model the allowing, dropping and redirecting probabilities of flows per source node. These filter are designed as:

$$F_1(x) = \left(1 + e^{-\beta\left(\frac{x - E_1}{d}\right)}\right)^{-1} \qquad (5)$$

$$F_2(x) = \left(1 + e^{-\beta\left(\frac{x - E_2}{d}\right)}\right)^{-1} \qquad (6)$$

Here E_1 and E_2 represent the flow rate for which the probability of dropping and redirecting a flow is 0.5 respectively. β is a scaling parameter. The variable d represents the bandwidth consumed per node, which is B/t.

Fig. 2 illustrates one sample sigmoid curve for each filter. This figure corresponds to the setting where $B = 1000$ units, $t = 2$ and $\beta = 20$. The firewall drops a flow of rate x with a probability $F_1(x)$ and redirects with a probability of $F_2(x)$. Here x represents the sum of all flows per user. It is worth noting that some of the legitimate flows might get dropped at the firewall. We consider that the defender decides the value of E_1 and E_2, which are the only control parameters for these filters. In Fig. 2, we consider the number of nodes sharing a network pipe of limited bandwidth as two and hence the legitimate share per user should not exceed half of the original pipe bandwidth. This share per user can also be represented using d, which in this case becomes 500 $(d = B/t = 1000/2 = 500)$. To make the analysis simpler, we correlate the two thresholds as $E_2 = d = 500$ and $E_1 = d * 1.25 = 625$.

To make a graceful decision, the defender designs three probabilistic functions $F_d(E_1, E_2, x)$, $F_r(E_1, E_2, x)$, and $F_a(E_1, E_2, x)$, which represent the probabilities with which the flows from a source will be dropped, redirected or allowed respectively. The 3rd argument of these functions, namely x, denotes the total bandwidth accessed by a particular source node. Note that the summation of these three probabilistic functions is always 1, and $x = r \cdot u$; where r is the bit-rate of each flow and u is the number of flows.

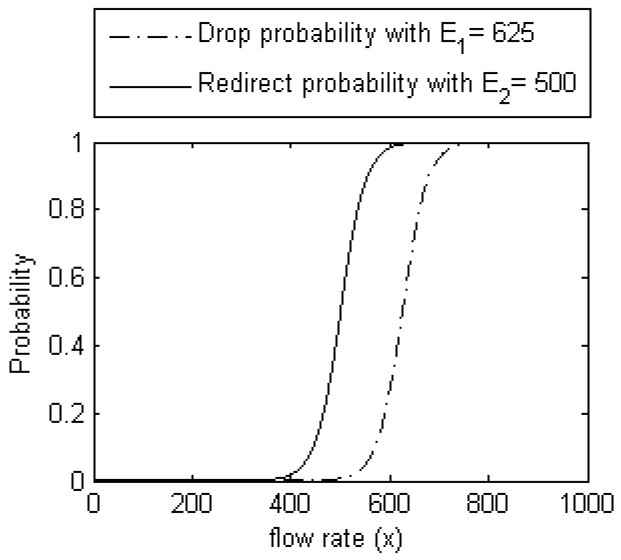

Figure 2. Plot of sample $\boldsymbol{S}$ curves: Drop or redirect rate of a flow at the firewall is modeled by a $\boldsymbol{S}$ curve. The X axis is the flow rate and the Y axis is the drop and redirect probabilities computed by the filters. The parameters $\boldsymbol{E_1}$ and $\boldsymbol{E_2}$ represent the flow rate for which the drop and redirect probability is 0.5.

Fig. 3 illustrates the arrangement and working of these two filters F_1 and F_2 together for creating the three probabilistic functions F_a, F_r and F_d, which are used for computing the probabilities for flows from a node that should be allowed to reach the target server, redirected to the honeypot or dropped by the firewall. The variable x which denotes the total bit-rate from each source is the input for the first filter F_1. This filter decides the probability whether flows from a node should be dropped or not. The probability for dropping flows from a user is directly obtained from this first filter and is denoted by F_d.

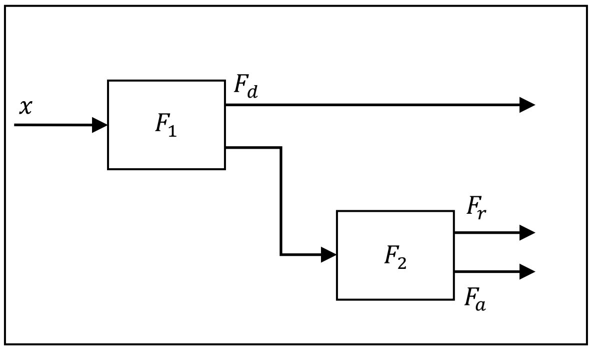

Figure 3. Filter arrangement: Filters $\boldsymbol{F_1}$ and $\boldsymbol{F_2}$ represent the defender's defense controls. These are used to compute the probabilities of allowing ($\boldsymbol{F_a}$), dropping ($\boldsymbol{F_d}$) or redirecting ($\boldsymbol{F_r}$) incoming flows.

The probabilities of redirecting flows from a source node to the honeypot (F_r) or allowing it to reach the target server (F_a) are obtained by using these two filters in combination. These probabilities for dropping, redirecting and allowing flows are defined as $F_d = F_1$; $F_r = F_2 \cdot (1 - F_1)$ and $F_a = (1 - F_1) \cdot (1 - F_2)$.

Based on the above decision factors and probability functions, if the attacker sends u flows from each attack node, we derive the following analytical results represented in equations (7) and (8). These are derived using the same logic as in equations (1) and (2) while considering the attack condition.

Average bandwidth consumption (by the attacker) ratio:

$$v_b^d = \frac{m \cdot u \cdot F_a(E_1, E_2, r_a \cdot u)}{S \cdot F_a(E_1, E_2, r_l \cdot s_l) + m \cdot u \cdot F_a(E_1, E_2, r_a \cdot u)} \qquad (7)$$

Ratio of lost legitimate flows to the total number of legitimate flows on average:

$$v_n^d = P\left[\gamma_l > \frac{B}{S \cdot F_a(E_1, E_2, r_l \cdot s_l) + m \cdot u \cdot F_a(E_1, E_2, r_a \cdot u)}\right] \qquad (8)$$

Here $S = s_l \cdot n$; where s_l is the mean value of the total number of flows for one legitimate node and n represents the total number of legitimate nodes.

The purpose of honeypot redirection is to learn more about the attacker before the defense architecture begins dropping their flows. This step also reduces the load on the target server TS by offloading the attacker's traffic. Data collected using honeypots can be used in various ways. The methods in which defense architectures can use the data collected using a honeypot is outside the scope of this work. Hence we assume the amount of cost incurred by the defender for using a honeypot equals the amount of information gained from the attacker by using the same.

We assume that the defender instantiates a honeypot from an active honeynet [8, 13] when he decides to learn more about an attacker. In our case, this decision is made based on the bit-rate used by an attacker over the pipe (P_1, P_2). As the attacker's total bit-rate exceeds the firewall threshold E_2, his flows are likely to be redirected to the honeypot.

Instantiation of a honeypot requires a cost to the defender as valuable and limited resources are required for this process. We assume that each learning process per attacker requires an instantiation of a new honeypot for that respective attacker. Hence, one honeypot for every attacker is instantiated as they exceed E_2 and are below E_1. However, once instantiated we assume the cost per attacker is nominal.

The attacker wants to reduce the redirection of his flows to the honeypot as it does not assist him in accomplishing his goal of utilizing the pipe (P_1, P_2)'s bandwidth and his resources are wasted. We assign to this a weight factor for the attacker w_h^a.

The defender on the other hand is interested in redirecting the flows from an attacker for learning more about the attacker. However, instantiating a honeypot requires a cost to the defender. We assign to this a weight factor for the defender w_h^d.

We consider the amount of flows redirected to the honeypot as the defender's benefit of using the same as follows:

$$v_h^d = m \cdot u \cdot F_r(E_1, E_2, r \cdot u) \qquad (9)$$

We can compute the attacker and defender's payoffs (V^a and V^d) from expression (3) and (4), respectively by replacing v_b^{nd} by v_b^d and v_n^{nd} by v_n^d and extending it to include v_h^d.

Therefore the new payoff functions become:

$$V^a = w_b^a \cdot v_b^d + w_n^a \cdot v_n^d - w_c^a \cdot v_c - w_h^a \cdot v_h^d \qquad (10)$$

where w_b^a, w_n^a, w_c^a and w_h^a are the attacker's corresponding weight parameters.

On the other hand, we model the defender's net payoff as a weighted sum given as

$$V^d = -w_b^d \cdot v_b^d - w_n^d \cdot v_n^d + w_c^d \cdot v_c + w_h^d \cdot v_h^d \quad (11)$$

where w_b^d , w_n^d , w_c^d and w_h^d are the defender's weight parameters.

We use the notion of Nash equilibrium to determine the best strategy profile of these two players. Each player has the goal to maximize their payoff. The attacker needs to choose an optimum m and u. The defender needs to choose an optimum E_1 and E_2. The Nash equilibrium of this game is defined to be a pair of strategies $(\mathrm{m}^*, u^*, \mathrm{E}_1^*, \mathrm{E}_2^*)$ which simultaneously satisfy the following two relations:

$$V^a_{(m^*,u^*,E_1^*,E_2^*)} \geq V^a_{(m^*,\,u^*,E_1,E_2)} \forall E_1, E_2 \quad (12)$$

$$V^d_{(m^*,\,u^*,E_1^*,E_2^*)} \geq V^d_{(m,u\,,E_1^*,E_2^*)} \forall m, u \quad (13)$$

We can analytically compute the Nash equilibrium strategy profile (m^*, u^*, E_1^*, E_2^*), which could also be obtained through numerical computation for a particular game setting.

V. Simulation Results

We use MATLAB as the platform for numerical computation. The following analysis shows an interesting case in which the total bytes sent by the attacker remain constant, which means that the attacker only needs to set the value of m. In our future work, we aim to extend this analysis to a more general case.

We begin our simulation by first developing the firewall filters which are controlled by manipulating their mid-points E_1 and E_2. These filters are used for generating the probabilities for dropping, redirecting or allowing incoming flows. These probabilities are then used for building the various components based on equations (7, 8, 9) which together yield to the payoff for the attacker. Arbitrary weights for each of the components are then used to model this simulation as a real world scenario.

To represent our findings in a three-dimensional figure, we introduce a relation of between the two mid-points of the filters E_1 and E_2 such that $E_1 = E_2 * 1.25$. Hence the defender is only required to adjust one value, which is E_2 in our case.

As an example, let us consider one scenario where the attacker’s and the defender’s weight coefficients are the same ($w_b^a = w_b^d$, $w_n^a = w_n^d$, $w_c^a = w_c^d$ and $w_h^a = w_h^d$), i.e., $V^a = -V^d$ (as a zero-sum game). Fig. 5 illustrates the attacker’s payoff V^a for different number of attack flows u, and different values of threshold E_2 with $w_b^a = 200, w_n^a = 200, w_c^a = 2, w_h^d = 2, B = 40000, n = 50, m = 13, r_l = 30, s_l = 10, \sigma_1 = 15, \gamma = 10, m \cdot u \cdot r_a = 4000$. We observe a set of saddle points in Fig. 4 which represents the Nash equilibrium. This relates to Nash equilbirum since either player which tries to deviate from the same, receives a lower or equal payoff.

One such point we observe is $(450, 25, 3.8)$ which is a Nash equilibirum point. This point signifies the optimal value of the firewall mid-point: $E_2 = 450$, the number of flows per attacker: $u = 25$ and the payoff obtained by the attacker: $V^a = 3.8$. It can be observed (by definition of a Nash equilibrium) that either player which deviates from the above mentioned strategy receives a lower or equal payoff. We verify the existence of such saddle points using a contour plot which is shown in Fig. 4 under the mesh surface. A contour plot is illustrated with contour lines. We know that a contour line of a function of two variables is a line along which the function remains constant. Fig. 4 shows the contour plot for the variables u and E_2 beneath the mesh.

The region between the pair of lines closest to the circular contour on its right represent the Nash equilibrium region. As the number of contour lines used to represent the mesh are increased, these two lines get closer to a value of $V^a = 3.8$.

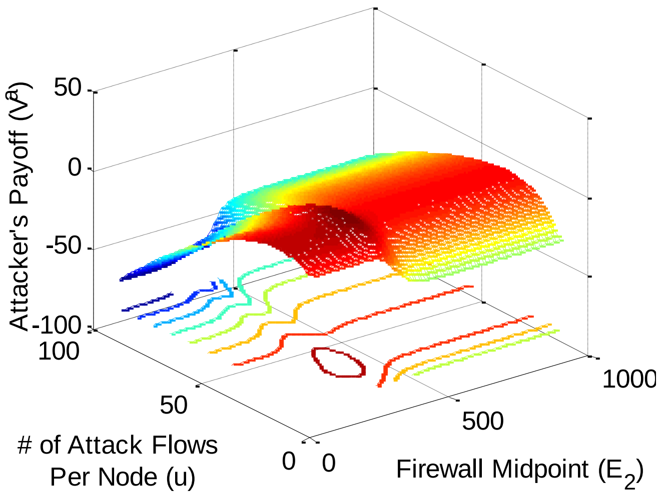

Figure 4. Attacker's payoff $\boldsymbol{V^a}$ for different number of attack flows $\boldsymbol{u}$ per node and different values of threshold $\boldsymbol{E_2}$. One of the saddle points which also represents the Nash equilibrium is observed at $(\mathbf{450}, \mathbf{25}, \mathbf{3.8})$.

VI. Future Work

This section provides a brief overview of our defense architecture which is currently in progress. The network topology as illustrated in Fig. 1 is setup using DETERlab.

Fig. 5 below shows our GIDA Module in detail. This implementation of the GIDA Module primarily consists of the following components: a Game Decision Agent, an Intrusion Detection System (IDS), a firewall and a routing module. As nodes upload TCP data streams to the Intended Destination, which is the Target Server, their flows are analyzed by the IDS. We use BRO which is an open source Network Intrusion Detection System (NIDS) for this purpose.

As an incoming flow is analyzed, we extract the following information: source and destination IP addresses/ports, starting time and duration of the flow, and amount of bits transferred so far. The bit-rate of the flow is then computed at regular intervals by sampling the bits transferred over these intervals. This information is then used by the Game Decision Agent to compute the thresholds for the permissible bit-rates for each flow. Decision to drop, redirect or allow the flows are determined by comparing the current bit-rate of flows with the permissible threshold which is computed by the Game

Decision Agent. Redirection to honeypot is achieved using the Routing Module. We implement this redirection using Click Modular Router.

With respect to the game decision analysis, we intend to consider the existence of multiple equilibria in various scenarios. We plan to extend our simulation to incorporate a normal distribution for selecting the sending rate of a legitimate flow. We aim to utilize our prior work in analyzing imperfect information games [14] to study the impact of imperfectness in the sensors of an intrusion detection system. Furthermore, we plan to simulate a dynamic game where both the attacker and the defender can alter their strategies during the attack event.

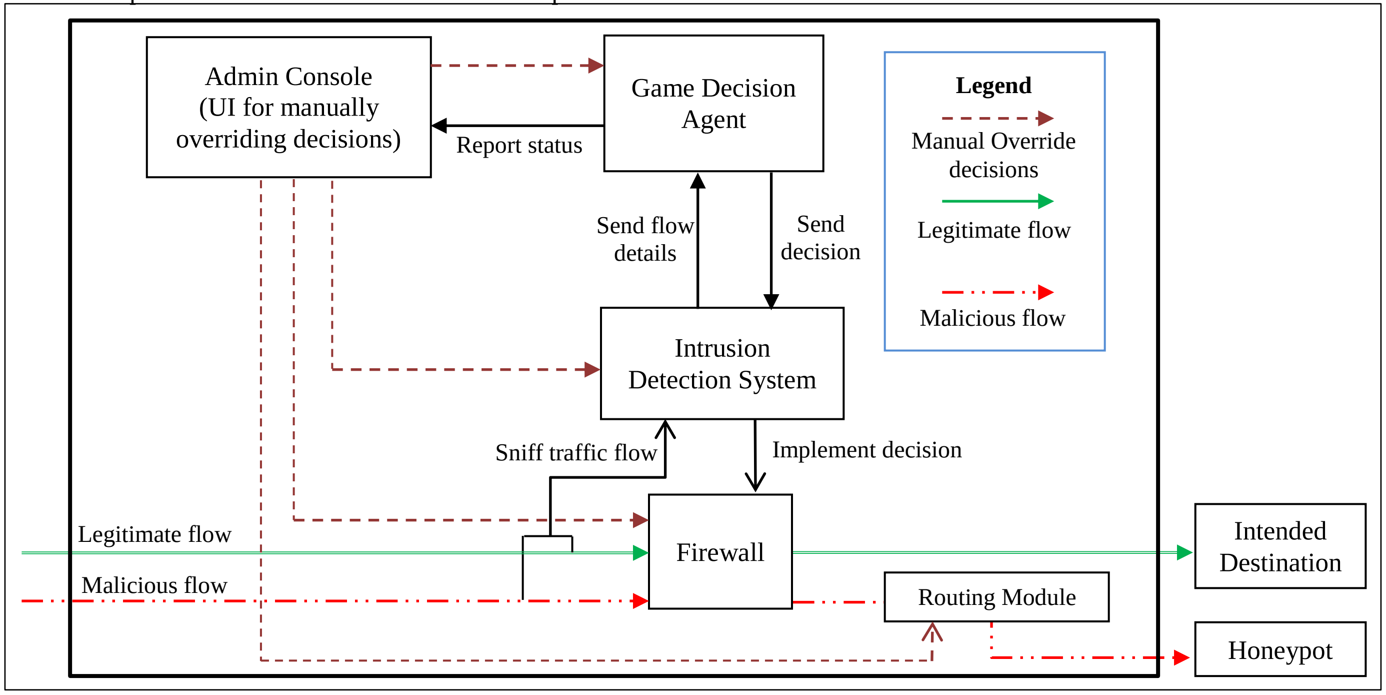

Figure 5. GIDA Module Architecture: The solid and dashed-dot lines represents path followed by legitimate and malicious flows across GIDA Module respectively. Legitimate flows are allowed to reach the intended destination, whereas malicious flows are either dropped or redirected to a honeypot. The dashed lines represent the administrator's ability to override the actions performed by the GIDA Module.

Conclusion

We observe that the domain of game theory provides a huge potential for addressing cyber security related problems as it can be leveraged for building a defense architecture which is placed on a solid analytical setting. We present a game theoretic model as a defense mechanism against the classic bandwidth consuming DoS/DDoS attacks on TCP-friendly flows. Validation of our game theoretic results was performed via MATLAB simulation.

References

[1] A. Yaar, A. Perrig, and D. Song. Siff: A stateless internet flow filter to mitigate ddos flooding attacks. In In Proc of IEEE Symposium on Security and Privacy, pages 130–143, 2004.

[2] D. Andersen. Mayday: Distributed filtering for internet services. In Proc. of the 4th Usenix Symposium on Internet Technologies and Systems, March 2003.

[3] Director of National Intelligence. "Annual threat assessment of the intelligence community for the senate armed services committee." *Statement for the Record*. March 2009.

[4] F. Lau, S. Rubin, M. Smith, and L. Trajkovic. Distributed denial of service attacks. In IEEE International Conference on Systems, Man, and Cybernetics, volume 3, 2000.

[5] J. Mahdavi and S. Floyd. TCP-friendly unicast rate-based flow control. Note sent to end2end-interest mailing list, Jan 1997.

[6] J. Mirkovic. A taxonomy of DDoS attack and DDoS defense mechanisms. ACM SIGCOMM Computer Communication Review, 34(2):39–53, 2004.

[7] J. Xu and W. Lee. Sustaining availability of web services under distributed denial of service attacks. IEEE Transactions on Computers, pages 195–208, 2003.

[8] M. Vrable, et al. "Scalability, fidelity, and containment in the potemkin virtual honeyfarm." *ACM SIGOPS Operating Systems Review* (ACM New York, NY, USA) 39 (2005): 148--162.

[9] J. Padhye, J. Kurose, D. Towsley, and R. Koodli. "A model based TCP-friendly rate control protocol." 1999.

[10] R. Chertov, S. Fahmy, and N. Shroff. Emulation versus simulation: A case study of TCP-targeted denial of service attacks. In Proc. of the 2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, page 10, 2006.

[11] S. Roy, C. Ellis, S. Shiva, D. Dasgupta, V. Shandilya, and Q. Wu. A survey of game theory as applied to network security. The 43rd Hawaii International Conference on System Sciences, 2010.

[12] X. Jiang, and D Xu. "Collapsar: A VM-based architecture for network attack detention center." *Proceedings of the 13th USENIX Security Symposium.* 2004. 15--28.

[13] Q. Wu, S. Shiva, S. Roy, C. Ellis, V. Datla, and D. Dasgupta. On Modeling and Simulation of Game Theory-based Defense Mechanisms against DoS and DDoS Attacks. 43rd Annual Simulation Symposium (ANSS10), part of the 2010 Spring Simulation MultiConference, April 11-15, 2010.

[14] Shiva, S., Roy, S., Bedi, H., Dasgupta, D., and Wu, Q. "A Stochastic Game Model with Imperfect Information in Cyber Security", The 5th International Conference on i-Warfare and Security, April 8-9, 2010.