On Modeling and Simulation of Game Theory-based Defense Mechanisms against DoS and DDoS Attacks

Qishi Wu, Sajjan Shiva, Sankardas Roy, Charles Ellis, Vivek Datla
Department of Computer Science
University of Memphis
Memphis, TN 38152, USA
{qishiwu, sshiva, sroy5, ceellis, vvdatla}@memphis.edu

Keywords: DoS, DDoS, game theory, static games, dynamic games

Abstract

As cyber attacks continue to grow in number, scope, and severity, the cyber security problem has become increasingly important and challenging to both academic researchers and industry practitioners. We explore the applicability of game theoretic approaches to the cyber security problem with focus on active bandwidth depletion attacks. We model the interaction between the attacker and the defender as a two-player non-zero-sum 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 traffics 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 both static and dynamic game models to compute the Nash equilibrium that represents the best strategy of the defender. We validate the effectiveness of our game theoretic defense mechanisms via extensive simulation-based experiments using NS-3.

1. INTRODUCTION

Today's national security, economic progress, and social well-being largely depend on the use of cyberspace. Despite considerable efforts made by the research and practicing communities over the last two decades, the cyber security problem is far from being completely solved. Particularly, the rapid growth in both inter-connectivity and computational power provides attackers with the potential to launch attacks of unprecedented scale and complexity [5]. Recent incidents [4, 12, 13] indicate that cyber attacks can cause severe damages to governments, enterprises, and the general public in terms of money, data confidentiality, and reputation.

Researchers have recently started exploring the applicability of game theory to address the cyber security problem [10]. The weakness of traditional network security solutions is that they lack a quantitative decision framework. As game theory deals with problems where multiple players with contradictory objectives compete with each other, it can provide us with a mathematical framework for modeling and analyzing cyber security problems.

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 static game, which is a one-shot game where no player is allowed to change the strategy. 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 block rogue traffics while allowing legitimate ones. We study the existence of the Nash equilibrium, which represents the best strategy of each player. We also show the benefit of using the game-theoretic defense mechanisms to the network administrator. Furthermore, we present a dynamic game model that allows each player to change the strategy during the game.

We validate our analytical results through extensive simulation-based experiments in NS-3. Our simulations provide performance measurements from situations involving a single attacking node and multiple ones. We develop a new module in NS-3, NetHook, which enables an application or a module to have direct access to packets as it traverses the network stack. The addition of this module satisfies the requirements of our packet filtering specifications. More importantly, NetHook facilitates packet inspection at any arbitrary level of the NS-3 network stack and can be used to implement any of the myriad of filtering applications to provide features such as firewall, network address translation, and intrusion detection system. We also develop two additional modules based on NetHook in NS-3: NetHookFlowMon, a layer-2 flow monitoring module that provides a per-flow association of packet flow information, and NetHookFilter, a layer-3 module that implements our game-inspired filtering approaches.

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

2. RELATED WORK

Bandwidth depletion in the form of DoS or DDoS is one of the most common attacks in cyberspace and various defense mechanisms have been proposed to mitigate the effect of such attacks [8]. We provide below a survey of related efforts.

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. PATRICIA [14] allows edge networks to cooperate to prevent misbehaving sources from flooding traffic. Lau *et. al* [6] conducted simulation-based analysis on various queuing algorithms including DropTail, Fair Queuing, Stochastic Fair Queuing, Deficit Round Robin, Random Early Detection, and Class-based queuing to determine the best queuing strategy in the target router during a DDoS attack. Chertov *et. al* [3] pointed out that DoS can be caused not only by flooding but also by exploiting the congestion window of TCP in the communication between the server and the client. Their experiments were based on the assumption that the length of the attack pulse controls the trade-off between attack damage and attack stealthiness. During the congestion avoidance phase, when packet losses occur, TCP halves its congestion window, which is exploited for attack.

Andersen *et. al* [1] proposed a proactive protection scheme against DDoS attacks by imposing an 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 and the set of overlay nodes are used to distinguish legitimate traffic from the attack traffic. Yaar *et. al* [17] proposed a Stateless Internet Flow Filter (SIFF) to mitigate DDoS flooding attacks based on per-flow states by protecting privileged flows from unprivileged ones. They used a handshake mechanism to establish a privileged flow that consists of marked packets with the "capability" obtained by the handshake. Wu *et. al* [15] constructed an adaptive cyber security monitoring system that integrates a number of component techniques such as decision fusion-based intrusion detection, correlation computation of event indicators, random matrix theory-based network representation of security events, and event identification through network similarity measurements.

Game theory has been widely applied in various application domains and is attracting more attention from network researchers for cyber security. Xu *et. al* [16] proposed a game-theoretic model to protect a web service from DoS attack. Network attacks [17, 11, 9, 7] have been extensively studied via simulations, which often require realistic parameters of simulated components. Our work focuses on mitigating DoS/DDoS attacks using a game theoretic approach and validating the game models in NS-3. Different from other simulation efforts, we develop several new modules of NS-3 for gathering packet statistics and mitigating malicious flows.

3. NETWORK TOPOLOGY

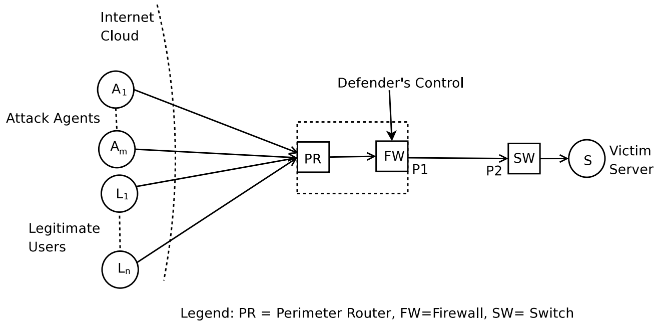

Figure 1. A generic network topology for DoS/DDoS attack.

We consider a generic network topology for DoS/DDoS attacks as shown in Figure 1, where the server S is connected to the Internet cloud via an edge switch (SW), a firewall (FW), and a perimeter router (PR). The bandwidth of the pipe (P1, P2) between the FW and the SW is limited and is subject to a DoS/DDoS attack. The defender's control is present at the FW. There are n legitimate users who need to communicate with the server S, and also, there is one attacker A who attempts to launch a denial of service attack by consuming most of the bandwidth of the pipe (P_1, P_2). The attacker A controls m attacking nodes that can send bogus packets. Note that DoS attack is a special case of DDoS attack when $m = 1$.

We would like to point out that our models and simulation-based experiments are not network-specific and are readily applicable to any DoS/DDoS scenarios in an arbitrary network topology with the following assumptions on network settings:

- A single attacker controls all of the attacking nodes, each of which sends a flow of bogus packets to the server S.
- There is an infinitely high bandwidth available on the channel between the PR and the FW, which is able to process all of the incoming packets.
- The defender has no knowledge of whether the flow is coming from the attacker or a legitimate user.
- The FW's belief on the legitimacy of the flow decreases with the increase of the flow rate.
- Some packets of a flow f are dropped in one of the two places: (i) at the FW; and (ii) at point P1 when the total incoming flow rate T is more than the available bandwidth B.
- 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.

For convenience, we tabulate all the notations and abbreviations used in our mathematical models in Table 1.

Symbol	Meaning
S	the victim server
PR	the perimeter router
FW	the firewall
SW	the switch
P_1	the outgoing point from FW
P_2	the incoming point to SW
B	the bandwidth of the pipe (P_1P_2) between the firewall FW and the switch SW
n	the number of legitimate users
m	the number of attacking nodes
r_l	the expected bit rate of a legitimate flow
σ_l	the standard deviation of a legitimate flow rate
r_A	the bit rate of an attack flow
γ	the minimum bit rate for a flow to be considered alive

Table 1. Notations and abbreviations used in the models.

4. GAME MODELS

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 an equilibrium in these games and also show the benefit of using the game-theoretic defense mechanisms.

The attacker attempts to find the most effective packet sending rate or botnet size, and the defender's challenge is to determine the best firewall settings to block rogue traffics while allowing legitimate ones. We first discuss some basic concepts of game theory and the profile of legitimate users, and then construct our game models.

4.1. 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. A 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.

4.2. Legitimate User Profile

We consider the presence of n legitimate users interested in communicating with the server S. The sending rate of a legitimate user is considered to be a random variable. In particular, we model the sending rate of legitimate users by picking n samples from a Normal Distribution, i.e. $X_i \sim \mathcal{N}(r_l, \sigma_l^2)$, $i = 1, 2, ..., n$ where X_i represents the sending rate of the i-th user, r_l is the mean value of a legitimate user's sending rate, and σ_l is the standard deviation. Therefore, the total incoming flow rate with no attack is $T^{na} = X_1 + X_2 .... + X_n$. By basic laws of probability, we have $T^{na} \sim \mathcal{N}(n \cdot r_l, n \cdot \sigma_l^2)$. We assume that the pipe bandwidth B is chosen such that $T^{na} < B$ with a high probability.

We first present our static game model where one single attacker controls all of the attacking nodes. Note that there is only one attacking node in a DoS attack, while there are multiple attacking nodes in a DDoS attack. Our discussion considers m attacking nodes and is generic with respect to DoS or DDoS attacks. When m is set to be 1, we get the DoS attack scenario. We further discuss the dynamic game model highlighting its difference from the static game.

4.3. A Static Game

In a static game, once a player decides his/her strategy, he/she does not have a second chance to change it. We consider that the attacker's reward is not necessarily equivalent in value to the defender's cost, i.e. it could be a zero-sum or non-zero sum game. The only actions available to the attacker are to set the sending rate and to choose the number m of attacking nodes. We assume that the sending rate is the same for all of the attacking flows, which is represented by r_A. In an attack situation, the total flow rate $T = (X_1 + X_2 + ... + X_n) + m \cdot r_A$. If $T > B$, then the denial of service occurs due to a congestion condition in the pipe (P_1, P_2).

4.3.1. Impact of the Attack with no Defense

When there is no defence mechanism in place, all the packets of each flow pass the firewall. However, if $T > B$, only a fraction of each flow can pass through the pipe (P_1, P_2). Let α denote this fraction, which is the same for each flow. We know that $(1 - \alpha)$ fraction of each flow will be dropped at P_1: if the bit rate of a flow is r, only αr bit rate will reach the server or destination. We further assume that the bandwidth resource is shared in a fair and equitable manner, and we have $\alpha = \frac{B}{T}$. Let γ be the minimum bit rate for a flow to be considered as a flow, which depends on the specific communication protocol used, and let n_g be the average number of legitimate flows, which are able to reach the server. We get $n_g = n \cdot P[X_i > \frac{\gamma}{\alpha}]$, where n is the total number of legitimate flows and $P[X > x]$ represents the probability that the value of the random variable X is higher than x. Similarly, α fraction of each attack flow will also be dropped at P_1. So, we have the average bandwidth consumption (by the attacker) ratio calculated as:

$$v_b^{nd} = \frac{m \cdot \alpha \cdot r_a}{B} = \frac{m \cdot r_A}{n \cdot r_l + m \cdot r_A}. \quad (1)$$

and the ratio of lost users to the total number of users on average calculated as:

$$\begin{aligned} v_n^{nd} &= \tfrac{n-n_g}{n} \\ &= P[X_i < \tfrac{\gamma}{\alpha}] \\ &= P[X_i < \tfrac{\gamma(n \cdot r_l + m \cdot r_A)}{B}]. \end{aligned} \quad (2)$$

The attacker's objective is to increase v_b^{nd} and v_n^{nd}, which are considered as the rewards. Also, we assume that the attacker has to incur some cost to get control of an attacking node. We assume that the attacker's total 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 defined as:

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

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

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

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

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

4.3.2. Impact of the Attack in the presence of Firewall

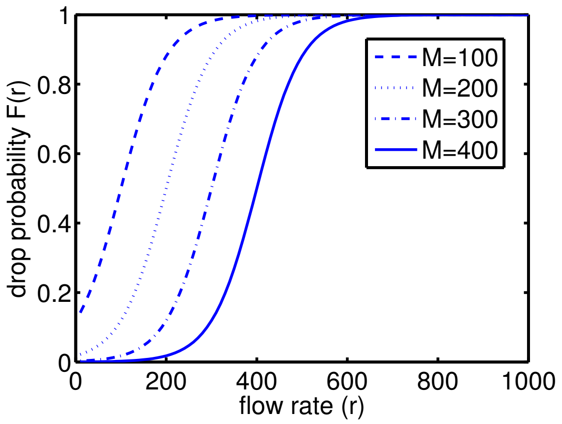

Figure 2. Plots of several sample S curves. Dropping rate of a flow at the firewall is modeled by an S curve. The X axis is the flow rate and the Y axis is the drop probability. The parameter M represents the flow rate for which the drop rate is 0.5.

The firewall is the defense agent of the network administrator: it drops the packets of an incoming flow with a probability depending on the flow rate. The dropping rate is modeled by a sigmoid function as follows:

$$F(x) = \frac{1}{(1 + e^{-\beta \frac{(x-M)}{B}})}, \quad (5)$$

where the parameter M represents the flow rate for which the drop rate is 0.5 and β is a scaling parameter. Figure 2 illustrates several sample sigmoid functions where $B = 1000$ units and $\beta = 20$. The firewall drops the packets of a flow of rate r with a probability $F(r)$. It is worth pointing out that some packets of a legitimate flow might also get dropped at the firewall. We consider that the defender controls the value of M, which is the only defense action.

Recall that r_l represents the expected rate of a legitimate flow. Let the average rate of legitimate flows passing through the firewall be r'_l. We have $r'_l = r_l \cdot (1 - F(r_l))$. On the other hand, the average rate of attacking flows passing through the firewall is $r'_A = r_A \cdot (1 - F(r_A))$. If we replace r_A by r'_A and r_l by r'_l in Equations (1) and (2), we obtain the following results: the ratio of average bandwidth consumption by the attacker is

$$v_b = \frac{m \cdot r'_A}{n \cdot r'_l + m \cdot r'_A}, \quad (6)$$

and the ratio of lost users to the total number of users on average is

$$v_n = P[X_i < \frac{\gamma(n \cdot r'_l + m \cdot r'_A)}{B}]. \quad (7)$$

Note that the right hand side of Equation (7) considers the losses due to both the firewall and the congestion. We can compute the attacker's and defender's payoffs V^a and V^d from Equations (3) and (4), respectively, by replacing v_b^{nd} by v_b and v_n^{nd} by v_n.

We use the notion of Nash equilibrium to determine the best strategy profile of these two players. Each player has the goal to maximize his/her payoff. The attacker needs to choose optimal values for m and r_A, and the defender needs to choose the best value for M in the sigmoid function to be used by the firewall. The Nash equilibrium of this game is defined to be a pair of strategies $(r_A^*, m^*,\ M^*)$, which simultaneously satisfy the following two relations:

$$V^a_{(r_A^*,\ m^*,\ M^*)} \geq V^a_{(r_A,\ m,\ M^*)} \quad \forall\ r_A,\ m$$

$$V^d_{(r_A^*,\ m^*,\ M^*)} \geq V^d_{(r_A^*,\ m^*,\ M)} \quad \forall\ M$$

We can analytically compute the Nash equilibrium strategy profile $(r_A^*, m^*,\ M^*)$, which could also be obtained through numerical computation for a particular game setting. 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 remains constant, i.e., $m \cdot r_A$

does not change, which means that the attacker only needs to set the value of m. In our future work, we will extend this analysis to a more general case. As an example, let us consider the scenario where the attacker's and the defender's weight coefficients are the same ($w_b^a = w_b^d$, $w_n^a = w_n^d$, and $w_c^a = w_c^d$), i.e., $V^a = -V^d$ (in a zero-sum game). Figure 3 illustrates the attacker's payoff V^a for different numbers m of attack flows, and different values of M with $w_b^a = 1000$, $w_n^a = 1000$, $w_c^a = 10$, $B = 2000$, $n = 20$, $r_l = 60$, $\sigma_l = 20$, $\gamma = 10$, and $m \cdot r_A = 5000$. We observe a saddle point at $m^* = 22$, $M^* = 225$, which represents the Nash equilibrium. The attack flow rate $r_A^* = 227.27$ corresponding to $m^* = 22$.

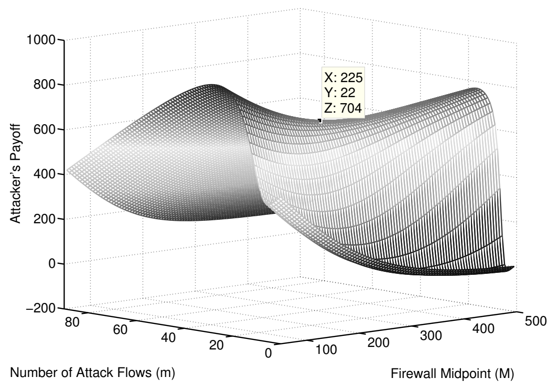

Figure 3. The attacker's payoff V^a for different numbers m of attack flows and different values of M (the firewall midpoint). We observe a saddle point at $m^* = 22$, $M^* = 225$, which represents the Nash equilibrium.

4.4. A Dynamic Game

In the static game model discussed above, no player has the chance to modify his/her strategy. Once the attacker sets the value for the flow rate r_A and the number m of attacking nodes, they remain fixed throughout the game. Similarly, the defender is not allowed to change the value of M, i.e. the firewall midpoint. The dynamic game model allows the players to change their strategies. This property may shift the game equilibrium point, i.e, the strategy profile (r_A, m, M) may change during the game.

The entire game duration is considered as a sequence of time steps. As an example, the attacker A can think that if he/she sets r_A low and m high during the first few time steps, the defender D will set M to a low value, and then A can exploit it by setting r_A high and m low in the next few time steps assuming that D does not change M. On the other hand, the defender can also decide a strategy based on his/her anticipation of the attacker's behavior.

In general, it is harder to determine the Nash equilibrium for a dynamic game compared to the static game. Due to the space constraint, we do not present its formal analysis in this paper. Let us consider that the game lasts for h time steps in total. When h is infinitely large, the game is said to have an infinite horizon, otherwise it is with a finite horizon.

We first extend the notations used in the static game. Let r_{A_t}, m_t, and M_t denote the corresponding quantities at the t-th time step. We represent the attacker's and defender's payoffs at the t-th time step by V_t^a and V_t^d, respectively, which are determined by the strategy profile ($r_{A_t}, m_t,\ M_t$) at that step. Similarly, the attacker's and the defender's total payoffs are denoted by V^a and V^d, respectively.

We compute the total payoff of a player by adding his/her time serial payoffs over the entire game, i.e. $V^a = \sum_{t=1}^h V_t^a$ and $V^d = \sum_{t=1}^h V_t^d$. The attacker can construct his/her strategy by deciding r_{A_t}, m_t at the t-th step $\forall t = 1, \ldots, h$. Similarly, the defender can construct his/her strategy by deciding M_t at the t-th step $\forall t = 1, \ldots, h$. The strategy profile ($r_{A_t}^*, m_t^*,\ M_t^*,\ t = 1, \ldots, h$) leads to the Nash equilibrium if it simultaneously satisfies the following two relations:

$$V^a_{(r^*_{A_t},\ m^*_t,\ M^*_t,\ t=1,\ldots,h)} \geq V^a_{(r_{A_t},\ m_t,\ M^*_t,\ t=1,\ldots,h)} \quad \forall\ r_{A_t},\ m_t$$

$$V^d_{(r^*_{A_t},\ m^*_t,\ M^*_t,\ t=1,\ldots,h)} \geq V^d_{(r^*_{A_t},\ m^*_t,\ M_t,\ t=1,\ldots,h)} \quad \forall\ M_t$$

5. SIMULATION

NS-3 is an advanced simulator tool written completely in C++ with optional binding for experiment scripts written in Python. There have been many recent developments with numerous research teams contributing their research as different modules for the simulator. FlowMonitor [2] is one such model which has inspired us to develop our own application to monitor packet flows. Unfortunately, FlowMonitor was not applicable in our experiment situation as it depends entirely upon the traced output of packet data, rather than inspecting these packets as they traverse NS-3's protocol stack. In our module, we need to develop a packet filtering module based on the game theory model and collect statistics on that module. For this packet-filtering module, we implement a unique network hook, which is used to observe packet flow information as they actually pass through the stack rather than at the end of the simulation.

5.1. Development of New Modules in NS-3

The NetHookFilter module we developed provides a means to manipulate the standard packet handling routines in NS-3. This concept has been widely used in Linux kernel for packet filtering, mangling, NAT (network address translation) and queuing packets for user-land inspection. Linux's NetFilter makes connection tracking possible through the use of various hooks in the kernel's network code. These hooks are places that kernel code, either statically built or in the form of a loadable module, can register functions to be called for specific network events at pre-defined locations within the protocol stack.

NetFilter is a useful component of modern networked systems for addressing various issues regarding packet inspection and manipulation. Traditionally NetFilter implements hooks during a packet's traversal through the protocol stack at the following locations: pre-routing, local deliver, forward, and post-routing. Each hook corresponds to locations in which one might be interested in viewing/manipulating a packet as it traverses the stack. Unfortunately, this component does not yet exist within NS-3. In order to overcome this limitation, we have developed a new NS-3 module called NetHook, which can be aggregated to any node with a network protocol stack and enables a developer to integrate their own inspection module. This new module provides the capability of manipulating packets at any location within the protocol stack.

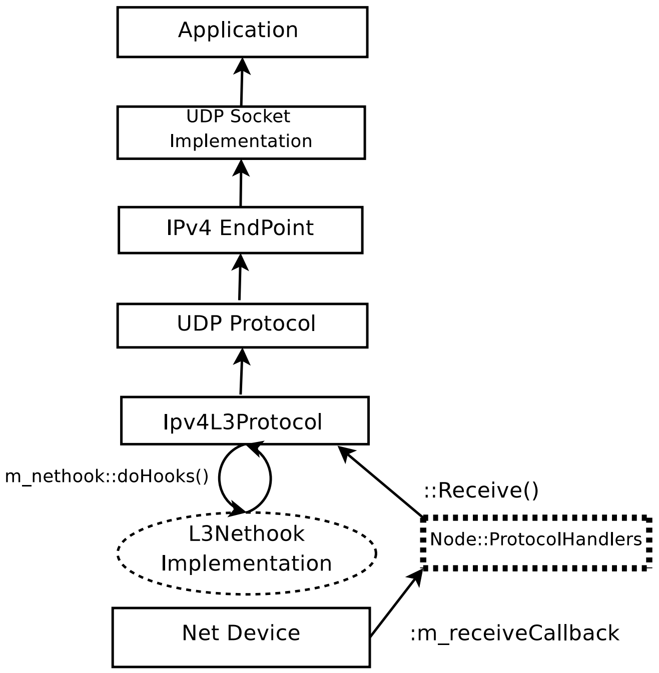

Figure 4. Implementation of NetHook. The function DoHook() enables the NetHook, which returns a boolean value that determines whether or not the packet needs to be dropped.

As shown in Figure 4, NS-3 NetHook is implemented via an ordered list of callbacks associated by callback type and a given priority value. The NetHook callback list is then initiated via a call from within the existing NS-3 code via the method DoHooks(), and is capable of running any arbitrary number of hooks given the appropriate hook type. NetHook is not limited to the traditional NetFilter inspection points, pre_routing, post_routing, local_in, local_out, and forward, rather it offers the flexibility for an NS-3 developer to implement an inspection callback at *any* location they desire within the NS-3 network system. The developer is left with the choice on the hook point for NetHook. They only need to implement the functor call and aggregate the NetHook object to the appropriate node within the topology.

5.2. Experimental Setup

We simulate our game-theoretic defense mechanisms in NS-3 to understand what aspects of networking would place constraints on the applicability of our model when applied to a real-world scenario. We wish to observe how control traffic would be affected, whether our model can be applied to data-intensive operations such as packet filtering, or even if the model could be applied at all. We adopt an attack model for raw bandwidth consumption where the attack nodes utilize UDP as the transport protocol in order to avoid using a modified TCP stack and avoid retransmission storms and their effect upon the simulation results. Figure 5 shows the relationship of the core infrastructure (perimeter router, firewall, and edge switch) and the packet filtering functionality.

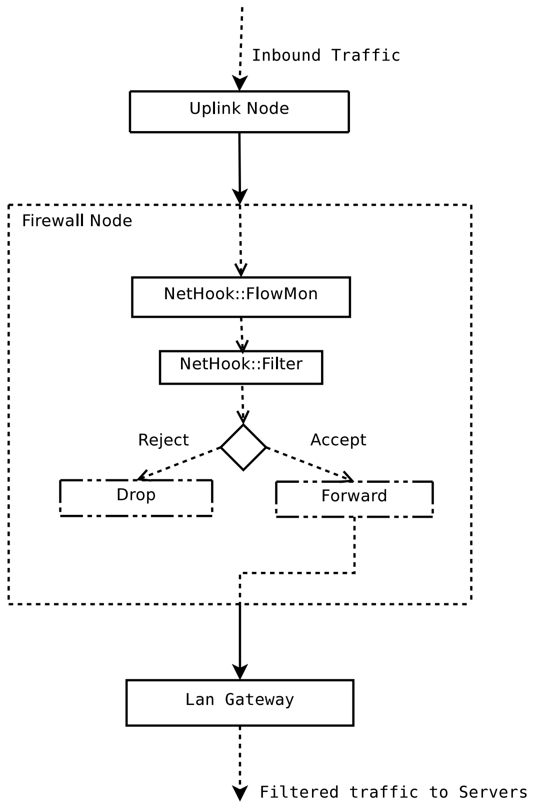

Figure 5. NetHook::Filter integration into experimental network topology.

We use the traditional dumbbell network topology for our experiments, as shown in Figure 1, which consists of three nodes where the leftmost node is the uplink node to which all legitimate and attack nodes are connected. The middle node of the dumbbell core is where we implement the packet filter. The rightmost node represents the local area network (LAN) side of the topology and provides connectivity for the server nodes. We use Point-to-Point channels to simplify the setup of the simulation topology. The left side of the topology has 1 Gbps of bandwidth while the rightmost side has 1 Mbps of bandwidth available with the bottleneck at the firewall node. The client nodes, either malicious or legitimate, are configured via the command line with arbitrary arguments for the number of nodes, packet size, and sending bit rates in order to support multiple runs with different settings. We use a constant bit rate generator available in NS-3, OnOffApplication, to generate packets destined to a server.

The experiments are run in 10 cycles, where there are 50 legitimate nodes whose packet size is 512 bytes and sends at a rate of 15Kbps. The first cycle has 5 attack nodes that send at a total of 5Mbps that is divided evenly between each attack node, and the number of attack nodes increases by 5 for each cycle. Within each cycle, we change the filter midpoint setting three times at 250Kbps, 500Kbps, and 700Kbps, respectively. Each cycle consists of 90 runs in total, 30 runs for each midpoint settings, in which there is no change in the simulation's number of attack nodes. Each run lasts for 600 seconds in length where the legitimate nodes send at a constant rate and the attack nodes begin at 30 seconds and last for 300 seconds in total. The exact same settings are used for cases without packet filtering in order to provide a baseline performance comparison. All simulations are run on an Intel Core-2 Duo 3Ghz machine with 4Gb of RAM and running Ubuntu 9.04 with Linux kernel version 2.6.28. Each run typically takes 3-10 minutes to complete depending on the number of nodes involved. The seed value of the random number generator in each run is incremented in order to ensure independent replication of the simulation results.

5.3. Results

The players' payoffs depend upon three components as discussed in Section 4. Our simulation focuses on the first component, which is the percentage of bandwidth consumed by the legitimate and attacking nodes. The second component, the fraction of active legitimate nodes, will be considered in our future work when the legitimate nodes send at different bit rates. The last component, the attacker's payoff falls outside the scope of this simulation. Figure 6 displays the effectiveness of our game theoretic defense mechanism against a DoS/DDoS attack. Figure 7 illustrates that there exists an optimal setting for the attacker, while Figure 8 shows the effectiveness of the attack can be reduced by selecting an appropriate midpoint setting. All experimental results indicate conclusively that the attacker can increase the number of attacking nodes, while decreasing the per-node bit rate, in order to bypass the filter. Conversely, the defender should select an appropriate *S*-curve midpoint in order to allow a majority of legitimate traffic while denying the attack traffic. If the *S*-curve midpoint is too high, then a large portion of the attack traffic will pass. These facts are consistent with the results from Figure 3 where we clearly see that there exists an optimal setting for both the attacker and the defender.

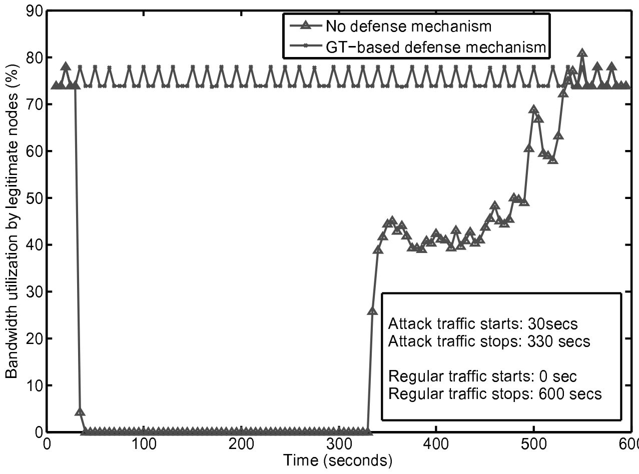

Figure 6. Impact of DDoS attack on legitimate bandwidth consumption: 5 attacking nodes transmit at 1Mbps each (total 5Mbps), 50 legitimate nodes transmit at 15Kbps each (total 750Kbps), and the *S*-curve midpoint is set at 500Kbps.

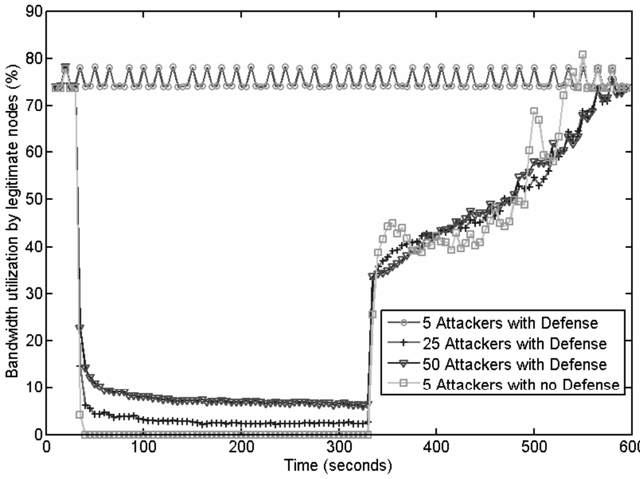

Figure 7. Bandwidth consumed by legitimate nodes when varying the number of attack nodes. The total attack bit rate remains at 5Mbps.

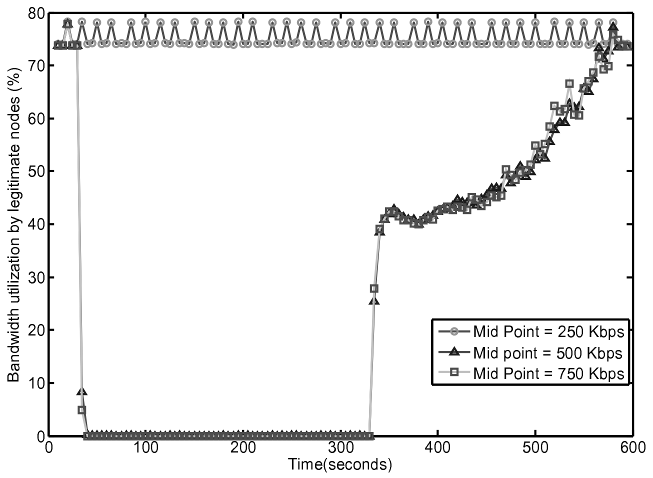

Figure 8. Bandwidth consumed by legitimate nodes when varying the *S*-curve midpoint. There are 15 attacking nodes whose aggregate rate is 5Mbps.

6. CONCLUSION AND FUTURE WORK

We presented a game theoretic model as a defense mechanism against a classic bandwidth consuming DoS/DDoS attack. Validation of our analytical results was performed utilizing the NS-3 network simulation tool.

In our future work, we will consider the existence of multiple equilibria in some scenarios. We plan to extend our simulation to incorporate a normal distribution to select the sending rate of a legitimate flow. In addition, we plan to investigate the applicability of our game-theoretic defense mechanisms in scenarios where the attacker is interested in exploiting specific protocol mechanisms to create attacking conditions. The TCP congestion window is one example of such possibilities. Furthermore, we plan to simulate a dynamic game where both the attacker and the defender can alter their strategies during the attack event. We also plan to contribute our NetHook module to the NS-3 codebase in order to make it available to other researchers interested in packet manipulation within the simulator.

ACKNOWLEDGMENT

This research is sponsored by the Office of Naval Research under Award No. N00014-09-1-0752 with University of Memphis.

REFERENCES

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

[2] G. Carneiro, P. Fortuna, and M. Ricardo. Flowmonitor-a network monitoring framework for the network simulator 3 (ns-3). In *NSTOOLS*, Pisa, Italy, Oct. 19 2009.

[3] 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.

[4] Security Focus. http://www.securityfocus.com/archive/1. *Security Focus Bugtraq Vulnerability Notification Database*, 2009.

[5] B. Gourley. Cloud computing and cyber defense. *Crucial Point LLC*, March 2009.

[6] 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.

[7] M. Liljenstam, J. Liu, D. Nicol, Y. Yuan, G. Yan, and C. Grier. Rinse: the real-time immersive network simulation environment for network security exercises. In *Workshop on Principles of Advanced and Distributed Simulation*, pages 119–128, 2005.

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

[9] D. Nicol, W. Sanders, and K. Trived. Model-based evaluation: From dependability to security. *IEEE Transactions on Dependable and Secure Computing*, 1(1):48–65, 2004.

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

[11] C. Sarraute, F. Miranda, and J.L. Orlicki. Simulation of Computer Network Attacks. In *Argentine Symposium on Computing Technology*, Aug. 30 2007.

[12] Packet Storm. http://packetstormsecurity.org/. *Packet Storm Vulnerability Database*, 2009.

[13] US-CERT. http://www.us-cert.gov/. *United States Computer Emergency Readiness Team*, 2009.

[14] L. Wang, Q. Wu, and Y. Liu. Design and Validation of PATRICIA for the Mitigation of Network Flooding Attacks. In *Proceedings of the 2009 International Conference on Computational Science and Engineering-Volume 02*, pages 651–658. IEEE Computer Society, 2009.

[15] Q. Wu, D. Ferebee, Y. Lin, and D. Dasgupta. Monitoring security events using integrated correlation-based techniques. In *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, page 47, 2009.

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

[17] 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.