Executive Summary

Cyber warfare is constantly increasing the last years with the infiltration of internet getting larger in more countries. The increasing number of users leads to more people who will probably try to perform an attack against someone or to more computers that will be used as slaves to perform an attack by an attacker. Thus it becomes a necessity to create better defence mechanisms to decrease the costs and damages caused.

One of the most common attacks that are being performed are Denial of Service attacks. Last month their number reached 28% out of the total amount of cyber attacks, ranking them the second most commonly performed attacks. Because of their nature, it often becomes difficult to defend against them properly.

In our research, we used a mathematical framework capable of creating models for rational participants, known as Game Theory. Game Theory gives us the means to model interactions between rational decision makers. We present a Game Theory model regarding Denial of Service attacks and a defence mechanism against them trying to find the optimum defence configurations for each attacker's configuration.

Apart from our model though it is necessary to have some data with which we could evaluate our theoretical approach. In this point we introduce Network Simulator 2, which we used to conduct our simulations to extract data for our theoretical study verification.

The basic points of our work are the following:

- Creation of a Game Theoretical model to study Denial of Service attacks, see pages 30-32.
- Creation of an Intrusion Prevention System-like mechanism for Network Simulator 2 to help us in our experiments, see pages 21-21 and 63-65.
- Verification of the proposed model by performing and studying simulations using Network Simulator 2, see pages 43-55.

Contents

1 Aims and objectives

Availability of services is one of the main concerns in computer networks. This is the main reason it has attracted so much attention in the last few year. Along with the increase of digital information, an increase in provided services has been noticed. The main source of those services and information is Internet, which has grown in number of peers massively the last few years. That increase in both services and peers though, can have several security dangers. One of those dangers is the proliferative Denial of Service attacks that take place. Denial of Service attacks have become one of the most common issues of networking since they make the target unavailable to its legitimate users, decreasing the quality of service provided.

Such attacks can have huge impacts, not only in terms of availability. A recovery from such an attack has high financial costs. The same also applies for the time that a service is unavailable. Especially if the target of the attack is a key node in a network, the impact can be vast. Lets take a router in a small company network as an example. Once it becomes unavailable, all the nodes that connect to the network through that router also loose their networking capabilities. Imagine that a well prepared attack can stop a whole company's network from working even in a few seconds.

In order to avoid such issues, we try to create defence mechanisms that are able to identify and stop attacks or even better prevent them from even taking place. The area of network security is of great interest and many researches have been made to create various defence mechanisms to reduce the amount of Denial of Service attacks.

In order for such defences to be created though, one must first be able to model and simulate attack procedures to test their effectiveness. Because we want to model smart attackers in order to make our defences better, we need a framework that can provide efficient decision making mechanisms. This is where game theory is introduced as a very efficient tool for that job. Game theory is a set of tools that help us model rational interactions between two or more parties. In our environment those parties are the attackers and defenders of a network and the interaction between them is the moves they make.

Many researchers have used game theory so far to model and test their proposed defence mechanisms. In our work we studied attacks where the network is modelled as a dumbbell topology and the attacker tries to cause a denial of service attack by spawning zombie hosts that send bogus data to the server. The defender has as a defence mechanism an Intrusion Prevention System in which he can set a threshold so that every data flow that exceeds that, will be immediately dropped. Moreover what makes the scenario interesting is that while the legitimate nodes' flow rates derive from a normal distribution, the attacking nodes' flow rates can derive from either a normal, an exponential or a Poisson distribution, whichever the attacker chooses.

The rest of the paper is organised as follows. In Section 2, we give an introduction to DoS attacks. Section 3 presents the basic notions of game theory.

Section 4 explains the importance of simulations and how we used them in our project and in Section 5, we present the related work to our topic. In Section 6 we give the definition of the problem we are approaching and our Game Theoretical model. Section 7 includes the analysis of our experiments and studies both for our model and simulations and in Section 8 we give an evaluation of our work. Lastly in Section 9 we present our goals for further research based on the model we present.

2 Denial of Service Attacks

A Denial of Service (DoS) attack is an attempt to make one or more computer resources unavailable to their legitimate users by sending bogus requests over the network to them by a singular attacking point. Distributed Denial of Service (DDoS) attacks are DoS attacks in which the number of attacking points is more than one. Hence we can say that DoS attacks are a subgroup of DDoS attacks where we have only one attacking point.

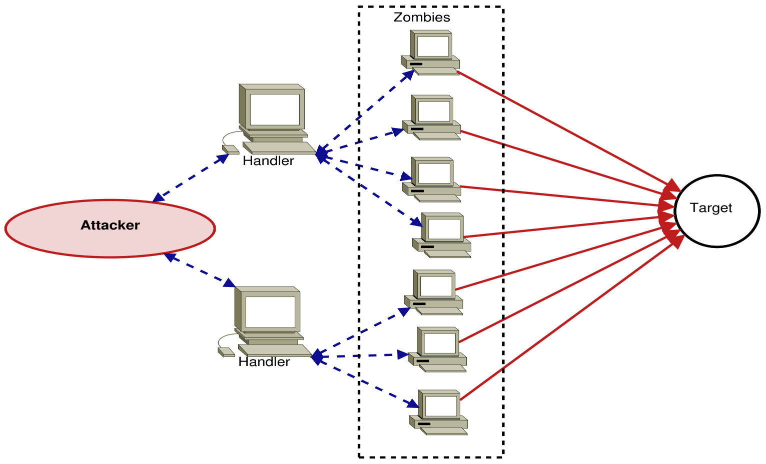


Figure 2.1: Distributed Denial of Service Attack

DDoS attacks can be performed by a single attacker that can take control over a number of systems and use them as attacking points or by many different attackers. When we have a single attacker then the process that is performed in an abstract description is as follows. First the attacker tries to find vulnerabilities on various systems in order to install attack tools on them so that he has control over them. After that the attacker sends command messages to the compromised systems, referred to as zombies, using them to attack the corresponding targets. The attacker is also able to use another type of attacking agents called handlers. Handlers act as an intermediate between the attacker and the zombie nodes and they are the ones sending the commands to the zombies to perform the attack. In Figure 2.1 we can see how the DDoS attack structure is. It is easy to understand that DDoS attacks are more powerful and more difficult to be protected from.

Nowadays the number of DoS and DDoS attacks has been increased and keeps on increasing. This has to do with the usage of new network protocols and the discovery of new vulnerabilities. Another factor is that today it is much easier to perform a DoS attack since several tools exists (e.g. LOIC (Batishchev 2009)) that make all the work for the attacker without much effort (such tools are supposed to be used only for legitimate usage).

DoS attacks have various forms and aim different targets and/or services. CERT Coordination Center defines three basic types of attacks (CERT/CC 1997):

1. consumption of scarce, limited, or non-renewable resources.

2. destruction or alteration of configuration information.
3. physical destruction or alteration of network components.

2.1 Flooding and Vulnerability based DoS Attacks

DoS attacks can be broadly classified into flooding (also called brute-force) and vulnerability based (also called semantic).

Flooding attacks try to make a resource unavailable by invoking vast amounts of bogus requests and trying to exhaust a key resource of the target. For example in a User Datagram Protocol (UDP) flooding attack the attacker sends vast amounts of requests to random ports in the target, trying to consume all the available bandwidth and making the target inaccessible by its legitimate users.

Vulnerability based attacks on the other hand try to make the target inaccessible by exploiting a bug in the installed software or flaws in the policy that is applied. This way the attacker, by sending a small number of carefully crafted requests is able to consume vast amount of the target's resources. One example is the Ping-Of-Death (POD) attack in which the attacker is able to make certain operating systems to reboot or crash by sending fragmented oversized Internet Control Message Protocol (ICMP) datagrams.

2.2 DoS targets

DoS attacks can have various targets according to the attack's scope and the desired outcomes. The target can be an end system (a computer implementing all the OSI model levels (Handley & Rescorla 2006)), a router, an ongoing communication, a link, an infrastructure or any combination or variant of these (Braden 1989). We now present some DoS targets in more detail and with examples (Abliz 2011).

DoS on applications

In application DoS attacks the attacker tries to make an application exhaust the host's resources (e.g. CPU) by exploiting a bug of the software. An example can be the SQL Wildcards attack in which the attacker takes advantage of the wildcards used by SQL to consume CPU resources. For example if an attacker uses the following search term in a search form that has no protection against this type of attacks, he will achieve to consume 90% of the host's CPU for around 6 seconds with a database of 2600 records (Mavituna 2008).

```
_[^!_%/%a?F%_D)_(F%)_%([)({}%){()}$&N%_)$*()$*R"_)][%](%[x])%a][$*"$-9]_
```

Usually operating systems set some boundaries on the resources that are available to an application in order to prevent such type of DoS attacks. Those boundaries should be correctly set though according to the type of the host (Web Server, Database Server etc.) or else the main (important) applications of the system become vulnerable to application DoS attacks.

DoS on operating systems

Operating system DoS attacks have the same principles as the application attacks with the difference that they might be more catastrophic for the system. One of the most well known operating system DoS attacks is the Transmission Control Protocol (TCP) SYN flooding (CERT/CC 1996). In this attack, an attacker sends many TCP SYN requests to the target without completing the TCP handshake, exhausting the target's state memory.

DoS on routers

IP routers are candidate targets for DoS attacks as well, exploiting routing protocols to attack a router or a network of routers (Handley & Rescorla 2006). One possible attack on a router is to overload its routing table with a vast amount of routes so that the router runs out of memory or has not enough CPU power to process all the routes (Chang, Govindan & Heidemann 2002).

DoS on ongoing communications

An attacker can also target an ongoing communication. Considered that the attacker can observe a TCP connection, then it is relatively easy to spoof TCP packets in order to reset or de-synchronise the communication to prevent further progress (Joncheray 1995). Even if the attacker is not able to observe a TCP connection he can still guess the port and the sequence numbers of the TCP communication and send the reset packets.

DoS on links

Dos on links is probably the most common type of DoS attacks. The simplest form of link DoS attack is to send enough non-congestion controlled traffic so that the link becomes extremely congested and legitimate users suffer from high packet loss (Handley & Rescorla 2006). In another type of link DoS attack, the attacker is able to deny access to legitimate users to a link by causing the router to generate sufficient monitoring or report traffic such that the link is filled. A possible candidate for such an attack are Simple Network Management Protocol (SNMP) traps (Rose 1991) since they are not normally congestion controlled.

DoS on infrastructure

Computer networks rely a lot on some main infrastructure that provides access to the rest of the network. This applies both in big networks (e.g. Internet) and local networks (e.g. home networks). If an attacker targets such an infrastructure it will result in mass problems for the legitimate users of the network as they will not be able to perform easily (or maybe at all) networking operations. One example of such infrastructure are Domain Name Systems (DNS) that act as translators in networks, translating human-friendly hostnames to actual IPs that represent a network node. If an attacker performs a DoS attack on a DNS, it means that the legitimate users that connect to the rest of the network though that DNS will not have access to Web, email and any other network service. The 13 DNS servers that are the root DNS servers of Internet, were subjected to a DoS attack in 2002 (Vixie, Sneeringer & Schleifer 2002) meaning that some of them were unreachable from some parts of the Internet. Another target for such attacks can be routers that are used in home networks. An attacker might be able to exhaust the IP pool allocated by a Dynamic Host Configuration Protocol (DHCP) of the router and preventing legitimate users to connect to the network (Handley & Rescorla 2006).

DoS on firewalls and Intrusion Detection Systems (IDS)

Firewalls are intended to defend the systems behind them against outside threats by restricting data communication traffic to and from the protected systems (Shirey 2007). While firewalls are a tool used to prevent DoS attacks they can become targets for DoS attacks as well. Firewalls can be classified as stateful or stateless. Stateful are the ones that hold state for the active flows that traverse them. Stateless are the quite opposite.

Attacks on stateless firewalls try to exhaust their processing resources. Attacks on stateful firewalls can be the same as in stateless and also such that the attacker sends traffic to the firewall in order to make it hold excessive state or state of pathological structure (Handley & Rescorla 2006). Since most firewalls are fail-disconnected this will cause a denial of service for the underlying network protected by the firewall. In the case of excessive state, the firewall may run out of memory and may be no longer able to process legitimate flows. When it comes to pathological structure the attacker sends data to the firewall in order to make its data structures work under their worst-case behaviour.

IDS suffer from the same types of attacks that firewalls do. The difference is that when an IDS is subjected to a DoS attack it will not deny service to the protected systems but it will be unable to protect them against subsequent attacks.

2.3 DoS at different protocol layers

Except the previous categorization, DoS attacks can also be classified according to the layer of the TCP/IP stack that the target service or protocol belongs to. The TCP/IP model specifies five different layers which are named Hardware, Network Interface, Internet, (Host-to-Host) Transport and Application layer and they map to the seven OSI layers (Stallings 1987) as shown in Figure 2.2. Since end systems implement all seven OSI layers, any of them can be possibly subjected to DoS attack. If we take routers as an example and compare them with end systems, then since routers only implement from network layer (Data Link in OSI model) and below, not all possible DoS attacks of end systems are applicable to routers. From now on we will use the OSI model do classify DoS attacks on the different layers (Compton & Hornat 2007).

[1] `http://www.tcpipguide.com/free/t_TCPIPArchitectureandtheTCPIPModel-2.htm`

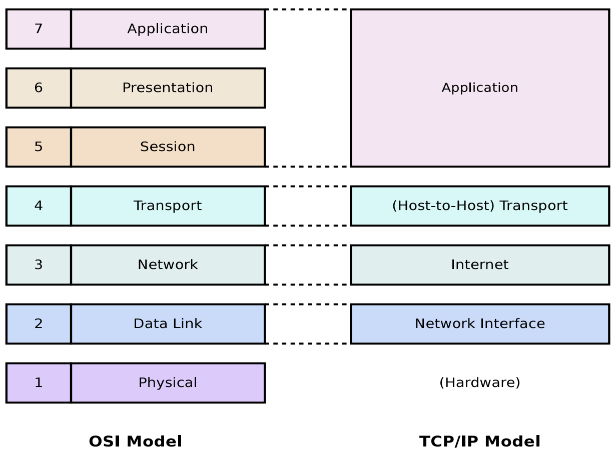


Figure 2.2: OSI Reference Model and TCP/IP Model Layers (From[1])

DoS attacks on Physical layer

Because of the Physical layer's nature, in order for an attacker to launch a DoS attack, he must be close to the network's infrastructure or inside it. Networks that rely on a shared medium are possible targets for DoS attacks, since if the shared medium is under attack then all the users that connect to it will experience bad networking behaviour. For example an attacker that has in his possession a laptop (or any computer) with a high output wireless client card and a high gain antenna is able to attack a wireless network by generating RF noise that reduces the signal-to-noise ratio by saturating the 802.11 frequency bands making the network unusable.

DoS attacks on Media Access Control (MAC) layer

MAC layer is subject to DoS attacks since it has no means to distinguish between legitimate and illegitimate frames, meaning all the frames are being processed. An attacker can spoof his MAC address and transmit spoofed management frames to harm the network. For example on the Authentication/Association flooding attack, the attacker spoofs his MAC address and sends authentication requests to the access point exhausting its IP address pool or its memory making the network unavailable or with bad quality for the legitimate users (Handley & Rescorla 2006).

DoS attacks on Network layer

Network layer DoS attacks attempt to jam communication links making the legitimate users experience increased dropped traffic that stalls their interaction. The most common attack on Network layer is the Smurf attack (daemon9, route & infinity 1996) in which the attacker sends many ICMP ECHO REQUEST packets to the network with a source address of the target victim (spoofed address). The nodes in the network respond to the request flooding the victim with large amount of messages. This attack consumes the network's bandwidth and also the victims resources making it unusable by its legitimate users.

DoS attacks on Transport layer

Transport layer DoS attacks mostly target operating systems by sending many connection requests to a host and exhausting its resources. They are very effective and hard to trace back because attackers use IP spoofing techniques to perform them. An example of Transport layer DoS attack is the Transmission Control Protocol (TCP) SYN flooding (CERT/CC 1996) that was described earlier.

DoS attacks on Application layer

DoS attacks on the Application layer focus on exploiting weaknesses and/or bugs of software. In other words an attacker can send massive amount of requests to an application in order to make it consume the host's resources. An example can be the HTTP flood attack in which the attacker sends a SYN packet and the target responds with a SYN ACK. The attacker then completes the three way handshake with an ACK packet and then sends a HTTP GET request for some common page on the target system. This process can cause very high computational load on the target system and possibly make it unavailable to its legitimate users.

2.4 DoS attacks taxonomies

In an effort to classify the different types of DoS/DDoS attacks many taxonomies have been proposed by various researchers according to their point of view. Kargl, Maier & Weber (2001) presented a taxonomy that focuses mainly in the actual attack phase, classifying DoS attacks according to the the type of the target, a resource that the attack consumes and the exploited vulnerability. Douligeris & Mitrokotsa (2004) presented a taxonomy that classifies DDoS attacks under two levels, as shown in Figure 2.3. The first level classifies attacks according to their degree of automation, exploited vulnerability, attack rate dynamics and their impact. In the second level the classification has to do with specific characteristics of each first level category.

Figure 2.4 shows the taxonomy presented by Mirkovic & Reiher (2004) for DDoS attacks looking at the complete attack mechanism, highlighting features that are specific to DDoS attacks. As we can see it shares some categories with the one in Figure 2.3, extends them and adds new ones. It is a far more analytical taxonomy that from our point of view can be used to distinguish DDoS attacks in several smaller groups that help in the creation of more targeted and efficient defence mechanisms.

2.5 DoS defence mechanisms

As networking protocols evolve and become more complex and advanced, the number of ways a DoS attack can be performed increase as it is mentioned earlier. It has become important to create adequate defence mechanisms to protect against

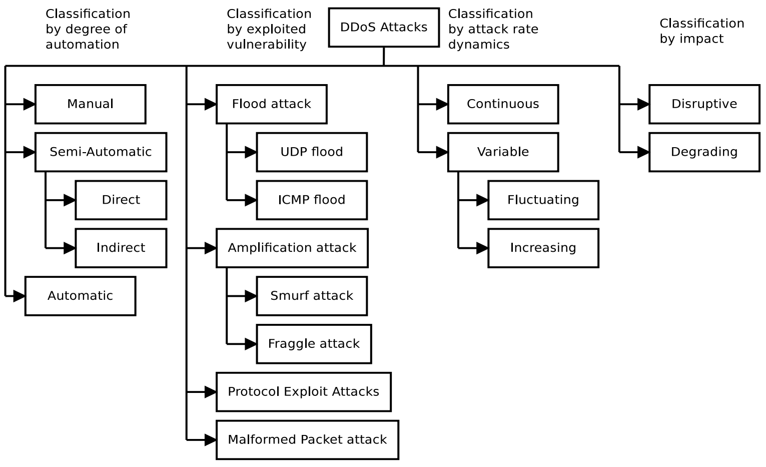


Figure 2.3: Taxonomy of DDoS Attacks (From Douligeris & Mitrokotsa (2004))

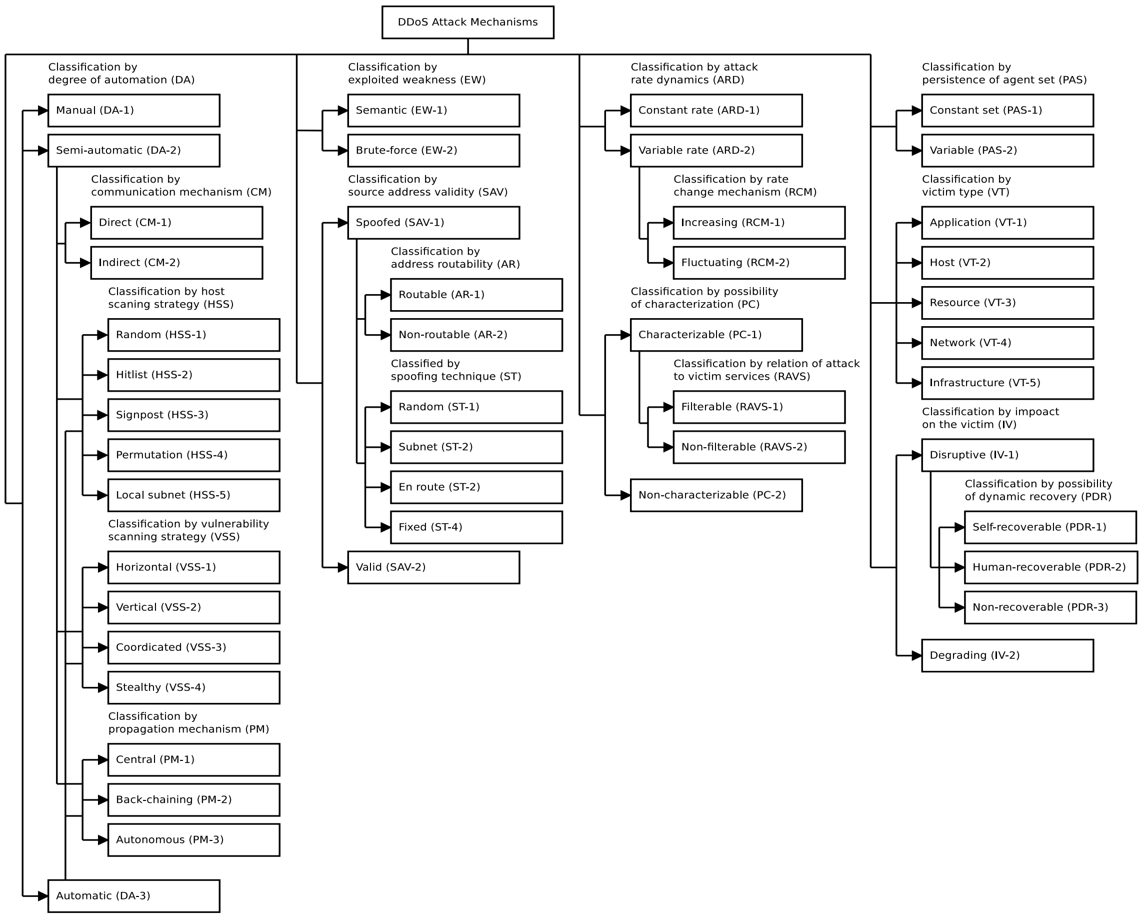

DDoS Attack Mechanisms
Classification by degree of automation (DA)
Manual (DA-1)
Semi-automatic (DA-2)
Classification by communication mechanism (CM)
Direct (CM-1)
Indirect (CM-2)
Classification by host scaning strategy (HSS)
Random (HSS-1)
Hitlist (HSS-2)
Signpost (HSS-3)
Permutation (HSS-4)
Local subnet (HSS-5)
Classification by vulnerability scanning strategy (VSS)
Horizontal (VSS-1)
Vertical (VSS-2)
Coordicated (VSS-3)
Stealthy (VSS-4)
Classification by propagation mechanism (PM)
Central (PM-1)
Back-chaining (PM-2)
Autonomous (PM-3)
Automatic (DA-3)
Classification by exploited weakness (EW)
Semantic (EW-1)
Brute-force (EW-2)
Classification by source address validity (SAV)
Spoofed (SAV-1)
Classification by address routability (AR)
Routable (AR-1)
Non-routable (AR-2)
Classified by spoofing technique (ST)
Random (ST-1)
Subnet (ST-2)
En route (ST-2)
Fixed (ST-4)
Valid (SAV-2)
Classification by attack rate dynamics (ARD)
Constant rate (ARD-1)
Variable rate (ARD-2)
Classification by rate change mechanism (RCM)
Increasing (RCM-1)
Fluctuating (RCM-2)
Classification by possibility of characterization (PC)
Characterizable (PC-1)
Classification by relation of attack to victim services (RAVS)
Filterable (RAVS-1)
Non-filterable (RAVS-2)
Non-characterizable (PC-2)
Classification by persistence of agent set (PAS)
Constant set (PAS-1)
Variable (PAS-2)
Classification by victim type (VT)
Application (VT-1)
Host (VT-2)
Resource (VT-3)
Network (VT-4)
Infrastructure (VT-5)
Classification by impact on the victim (IV)
Disruptive (IV-1)
Classification by possibility of dynamic recovery (PDR)
Self-recoverable (PDR-1)
Human-recoverable (PDR-2)
Non-recoverable (PDR-3)
Degrading (IV-2)

Figure 2.4: Taxonomy of DDoS Attack Mechanisms (From Mirkovic & Reiher (2004))

each possible threat. Because of the diversity of attacks though we require many different defences. In order to do so we must distinguish the possible defences and create a taxonomy to summarize and categorise them. In Figure 2.5 is the taxonomy presented by Mirkovic & Reiher (2004) for the different defence mechanisms that can be applied against the various DDoS attack mechanisms.

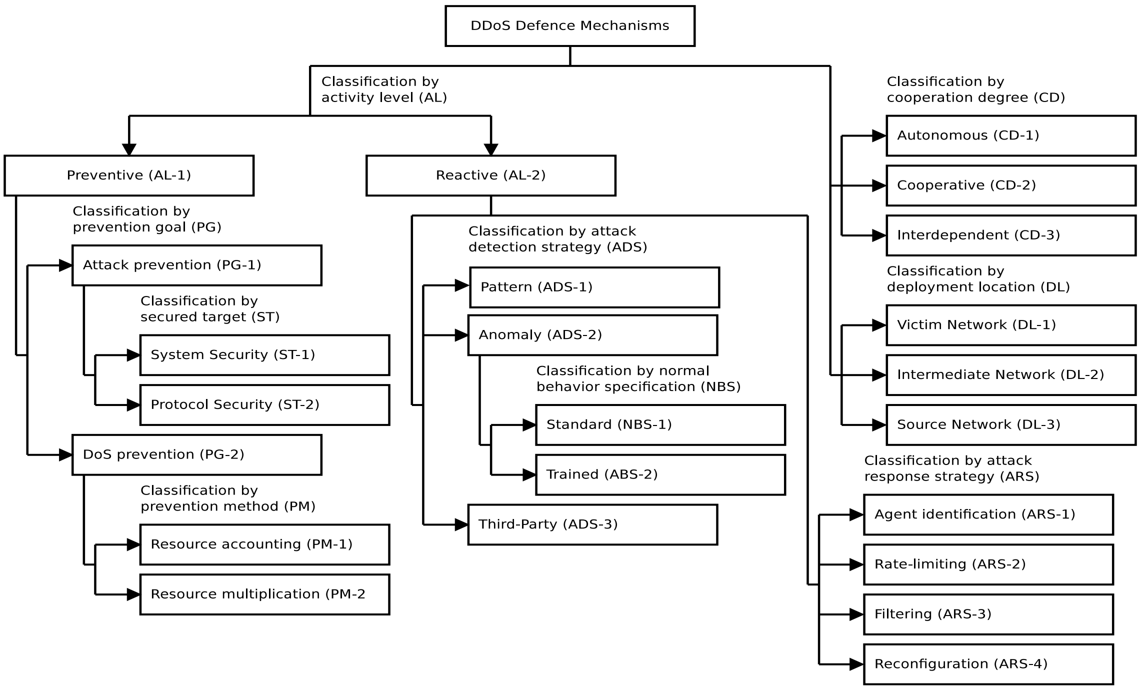

DDoS Defence Mechanisms
Classification by activity level (AL)
Preventive (AL-1)
Reactive (AL-2)
Classification by prevention goal (PG)
Attack prevention (PG-1)
Classification by secured target (ST)
System Security (ST-1)
Protocol Security (ST-2)
DoS prevention (PG-2)
Classification by prevention method (PM)
Resource accounting (PM-1)
Resource multiplication (PM-2
Classification by attack detection strategy (ADS)
Pattern (ADS-1)
Anomaly (ADS-2)
Classification by normal behavior specification (NBS)
Standard (NBS-1)
Trained (ABS-2)
Third-Party (ADS-3)
Classification by cooperation degree (CD)
Autonomous (CD-1)
Cooperative (CD-2)
Interdependent (CD-3)
Classification by deployment location (DL)
Victim Network (DL-1)
Intermediate Network (DL-2)
Source Network (DL-3)
Classification by attack response strategy (ARS)
Agent identification (ARS-1)
Rate-limiting (ARS-2)
Filtering (ARS-3)
Reconfiguration (ARS-4)

Figure 2.5: Taxonomy of DDoS Defence Mechanisms (From Mirkovic & Reiher (2004))

One of the most common prevention mechanisms used against DoS attacks are Intrusion Detection Systems (IDS). IDS monitor the events that occur in a network or system and analyse them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices. Apart from analysing, it is necessary also to distinguish between legitimate and not events. After that an IDS can attempt to stop possible incidents and/or inform the system administrator so that he can take control over the situation (Scarfone & Mell 2007).

The most common way that an IDS can automatically stop a possible incident is by applying certain rules on a firewall. Firewalls are software components that control flows in a network and make actions according to the rules configured. A rule can be to allow, deny or block a flow(in more advances configurations we can also do actions like encrypt), and can be applied to incoming, outcoming or both type of network flows. Firewalls can be either hardware or software. Hardware firewalls provide a higher level of security since they are separate appliances that have their own operating system and resources. Software firewall require a host to be installed on and will consume resources from it. While using firewalls we can defend against a large amount of attacks (e.g. smurf attacks), they are not sufficient under all circumstances.

An extension to IDS are Intrusion Prevention Systems (IPS) which are computer security devices capable of not only monitoring networks and/or appliances for malicious or unwanted behaviour but also react to it in real time in order to block, prevent and log those activities. For example a network-based IPS will operate in-line to monitor all network traffic for malicious code or attacks. When an attack is spotted, it can drop the attacking traffic while allowing the legitimate one. IPSs were invented in the late 1990s in order to resolve ambiguities in passive network monitoring by placing detection systems in-line. They are a considerable improvement upon firewall since they make access control decisions based on application content rather than IP addresses and/or posts as traditional firewall do (Newman 2009).

When using a server, the administrator must always keep it in good condition and configuration. What that basically means is that the server must be configured in such ways to reduce the probability of a DoS attack. Possible measures that need to be taken are limits in the resources used by specific applications (or in general) to avoid application layer type attacks.

Something else that should be considered is defence against bandwidth drawing attacks. One for example can configure the network in a such a way that important hosts would be reachable by more than one links. The secondary links can be active constantly or when there is increased bandwidth usage in the link.

Apart from those solutions, one can also implement certain protocols that can help against DoS attacks. The most common group of such protocols require clients to pay a price (in resources) in order for the server to accept communication from them. An example are puzzle based defence mechanisms (Fallah 2010), where the client is called to solve a puzzle issued by the server in order for the server to accept its request.

In general DoS attacks are a difficult thread to defend against. That makes the area attract attention and require solutions. The general idea so far is to create defences that try to identify illegitimate users and block them. To do so though, one must be able to understand how an attacker acts and this is where game theory comes to help us.

2.6 Summary

This section included a description of DoS attack and defence mechanisms and the way the can be classified into different categories. To make that categorization better, some taxonomies for both attack and defence mechanisms have been presented. As one can see, DoS attacks can be encountered in various forms, something that makes the creation of efficient defence mechanisms difficult. That is the reason that the research area around defences, for not only DoS attacks but in general, has been of great interest and attention.

3 Game Theory

Game theory is a set of analytical tools used to model and understand the interaction between decision makers under certain assumptions. Those assumptions are that the decision makers are rational and also take into account their understanding of the other decision makers' behaviour (Osborne & Rubinstein 1994). Game theory is dated back in 1838 when Antoine Cournot gave the first example of a formal game-theoretic analysis in the study of duopoly. The first formal theory though was introduced in 1921 by Emile Borel and in 1928, John van Neumann extended that theory in a "theory of parlor games". Game theory's establishment as an individual field came about 20 years later, in 1944, after von Neumann and Oskar Morgenstern published their work titled "Theory of Games and Economic Behaviour". This book is fundamental even today, since it includes the basic terminology and problem setup that is still used (Turocy & Stengel 2001).

In computer science and especially in computer security we model various situations that require decision making processes, such as in our case DoS attacks. Game theory is an efficient tool that helps us model such decision making processes. That is because it provides a methodology for structuring and analysing problems of strategic choice. In information security it is always better to follow Murphys Law that says "If anything can go wrong, it will!". In our case this means that we assume the attacker to be rational and that he even has partial or full knowledge of our network/system. This kind of models can be efficiently studied using game theory. In the rest of this section we give an overview of some basic game theory terminology, game types and models.

3.1 Basic Definitions

A *game* is the full description of the strategic interaction between the decision makers. It includes the action constrains and interests of the decision makers but not the list of actions that each player chooses to take. *Solution* of a game is the description of how the game will be played under the best possible strategies and the set of the outcomes it will have.

The *consequence function* associates a *consequence* with each action a player makes. The relation between the set of consequences for a players preference in a game is called *preference relation. Strategy* is the complete plan of all the actions that a player will take under any possible situation in a game. Strategies can be *pure* in the sense of taking a unique action in a situation and *mixed* when actions are chosen under a specified probability distribution for the plan. Another type of strategy that can be used is the *max-min* strategy in which a player tries to maximize his worst payoff against all possible moves of his opponent.

A *Nash equilibrium* is a solution that describes a steady state for a game but does not examine the process by which that state was reached. In other words it is a solution in which no player would want to change their strategy because that would lead to a decrease of their payoffs.

Decision makers are also referred to as *players* and form the basic entity in game theory models. Players can be individuals, machines or groups that act together.

In game theory players can be characterised as *rational* or *irrational.* Rational players are those who choose actions that gives the desired outcome with respect to the other player decisions. Irrational players do not take under consideration other player decisions and always make decisions that promise to give them the best possible results. *Payoff* for a player is the positive or negative outcomes for an action he/she makes in a game.

3.2 Game types

Games can be introduced in various models depending on certain factors (Osborne & Rubinstein 1994). We distinguish games as *cooperative* (or *coalitional*) or *non-cooperative* according to whether the set of possible primitive actions refer to groups of players of individuals accordingly.

Games can also be classified as *strategic/static* or *extensive/dynamic.* In strategic games each player chooses a set of actions once, is not allowed to change them afterwards and also all decision are made simultaneously (the players do not know the actions that the other players will take when they decide their plan). On the other hand extensive games specify an order of events meaning that players are allowed to alter their plan any time during the game when they are called to make a decision.

Another distinction between games is the one of *perfect* or *imperfect information.* In games of perfect information all players are aware of the moves that the other players have taken so far. If just one player is not aware of the other players' previous moves then the game is of imperfect information.

When it comes to the payoffs of the game, again we can name some categories (Shor 2001 - 2006). *Constant sum* games are those in which the sum of the payoffs of all the players is the same for any outcome of the game. This means that the expense of a player is the gain of his opponent. *Zero sum* games are a special category of constant sum games in which the sum of all the players' payoffs is always again the same under any outcome, but is always zero. Lastly, in *variable sum* games the sum of all the players' payoffs differs according to the strategies they choose.

One very common theorem used in game theory is the *minimax theorem* which states that (Osborne & Rubinstein 1994):

> For every two-person, zero-sum game with finitely many strategies, there exists a value V and a mixed strategy for each player, such that (a) Given player 2's strategy, the best payoff possible for player 1 is V, and (b) Given player 1's strategy, the best payoff possible for player 2 is -V.

What this means is that regardless the strategy used by the second player, the first's player strategy always guarantees him a payoff of V. Similarly regardless the strategy used by the first player, the second's player strategy always guarantees him a payoff of -V. If we try to explain it as the max-min strategy then we can say that in the minimax theorem, each player tries to minimise the other player's maximum payoff. Because the game that this theorem applies is zero-sum, each player also maximises his minimum payoff, which means that he follows a max-min strategy.

Example game

Lets give a brief example of a two player game. Rock-Paper-Scissors is a classic two player game that can be modelled using game theory. In Table 3.1 we can see the payoffs table for the two players. This example describes a zero sum game and the only equilibrium can be found using mixed strategies. That means that both players make a move with equal probability to be one of the three choices. Doing that, then the payoff of the opponent player is expected to be zero.

	Rock	**Paper**	**Scissors**
Rock	0,0	-1,1	1,-1
Paper	1,-1	0,0	-1,1
Scissors	-1,1	1,-1	0,0

Table 3.1: Payoffs for Rock-Paper-Scissors

Bayesian Games

Bayesian games are related to strategic games with the difference that they are used to model situations in which some of the parties are not certain of the characteristics of some of the other parties. Bayesian games introduce the idea of *states of nature*, denoted by a finite set for convenience, that describe the relevant characteristics of all players. Each player has a belief about each state of nature which is modelled by a profile of signal functions; where signal is the event that the player observes and has a belief for.

Repeated Games

In repeated games players are able to interact by playing a similar stage game many times, even infinite.

Evolutionary Games

In evolutionary games all players are not required to be sophisticated and also players do not necessary assume that the other players are rational (Turocy & Stengel 2001). In this type of games the notion of rationality is replaced with the concept of reproductive success. This means that strategies that are successful on average will be used more frequently and thus prevail in the end.

Stochastic Games

In stochastic games the transitions between states are probabilistic. In each state, the players make their moves and receive their payoffs according to the current game state. After that the game transitions into a new state with a probability based on the players' actions and the current state.

3.3 Summary

In this section we defined some basic notations of game theory and explained some game models and strategies. After understanding how game theory works one can also see the negative aspects of using it in network security. For starters is it difficult to learn some aspects of network security and there exists also information limitation. The other reason is that it is difficult to interpret the equilibrium and to define the payoffs in network security since they are affected by many aspects and some of them are abstract and cannot be easily measured.

Despite those negative aspects though, game theory remains a strong tool that can help us model network security scenarios and to create better defences against smart attackers.

4 Simulations

Simulations provide a low cost experimentation mechanism used massively in a variety of situations. They are valuable since they can be seen as a low cost testing step before trying to conduct experiments in real systems that come with higher costs. As it is obvious they are very popular among researchers as they provide them with an easy way to conduct their experiments.

Simulations can be executed in various forms. For example we can conduct them using simulating software that provides a way of representing real nodes and links as software entities that interact as in a real network. The problem with that kind of software is that it is often not able to cooperate with any situation we want, since several mechanisms might not be implemented in the simulating software.

Another way of performing simulations is though interconnected virtual machines. This is a more advanced technique that provides more flexibility as we can also have access to operating systems and software operations that in our case can be used to experiment with some kinds of DoS attacks. On the other hand the configuration of the simulation can be trickier and also the required resources that will be needed for the virtual machines can be huge depending of the number of required virtual machines. That can be a problem because often we need to simulate a large amount of nodes.

In our experiments we will stick with software simulations since we can easier have access to the required resources. As for the software we will use, it will be Network Simulator 2 (Mccanne, Floyd & Fall N.d.) with a custom module that we created to help us conduct our experiments.

4.1 Network Simulator 2

Network Simulator 2, known mostly as NS2, is an event-driver network simulation tool that is widely used in the academia. It is capable of performing simulations both for wired and wireless functions and protocols (e.g. TCP, UDP, routing algorithms). Generally speaking, NS2 provides an environment for specifying network protocols and testing their behaviour and efficiency.

Due to the fact that it is an opensource project with high flexibility and modular nature, NS2 has gained constant popularity in the networking research community and also in academia since its birth in 1989. Several people have helped in the growing and maturing process of the project even from its first steps by adding new functionalities and improving existing ones. Some of the most important participants in this effort are the University of California and the Cornell University who developed the REAL network simulator[2], which is the foundation of NS. Starting from 1995 the Defence Advanced Research Projects Agency (DARPA) supports the development of NS through the Virtual InterNetwork Testbed (VINT) project (Bajaj, Breslau, Estrin, Fall, Floyd, Haldar, Handley, Helmy, Heidemann,

[2]REAL was originally implemented as a tool for studying the dynamic behaviour of flow and congestion control schemes in packet-switched data networks.

Huang, Kumar, McCanne, Rejaie, Yu, Xu & et al. 1998)[3]

In this document we will not get into very much detail about NS2 and how it works. We are only going to give some insights to make the reader comfortable with the way this software works and what changes needed to be made in order for the simulating experiments that will be presented to be carried out using NS2. Thus in the following sections we are going to discuss briefly about the basic architecture of NS2 and the changes that we made in the source code to be able to conduct our simulations.

4.2 Basic Architecture

NS2 is a dual language project as it consists of both C++and Object-oriented Tool Command Language (OTcl) code. The back end of the simulator and its functionalities are written in C++while OTcl is used as a front end for setting up the simulations by configuring the simulating objects, their parameters and the events that take place during the simulation. TclCL is used as a link between C++and OTcl, meaning that is maps C++objects to the corresponding OTcl ones. We must note though that OTcl supports inheritance and so some OTcl objects do not map directly to a C++object. We refer to objects in the OTcl world that are mapped to a C++object as handles. Those objects can be nodes, links, queues etc. Conceptually handles are only strings in the OTcl domain and they contain no functionality. All the provided functionalities are implemented in the mapped C++objects (e.g packet receiving). Handles are used as a front end to the user allowing him to set up their functionalities and events easily, and for interaction with other OTcl objects. For example one can create two nodes and also a link with certain bandwidth and delay to connect them.

Figure 4.1 shows how a simulation takes place in NS2. The user creates a simulation script written in Tcl which he passes to the NS2 executable. According to the settings that the user has specified in the simulation script, two types of trace files will be given as an output. The text trace file in which each line represents a packet interaction in the network and the nam trace file which represents a visual representation of everything that happened during the simulation. It is not possible to output graphs regarding the network's behaviour immediately but there are other means. Since every packet interaction is included in the text trace file, the user can use a scripting language (e.g. awk, python etc.) to parse those results and create any type of statistics that he needs. Also this way he can create plots regarding those statistics. Another way is to use one of the trace analysing software that have been written by third parties. The downside though is that most of them are used for wireless traffic and do not work with wired networks.

4.3 Intrusion Prevention System module

For the purposes of our simulations with NS2 the functionality of dropping packets from specified nodes was crucial. Unfortunately this option is not available in NS2

[3]The VINT project, which is funded by DARPA, aimed at the creation of a network simulator that will initiate the study of different protocols for communication networking.

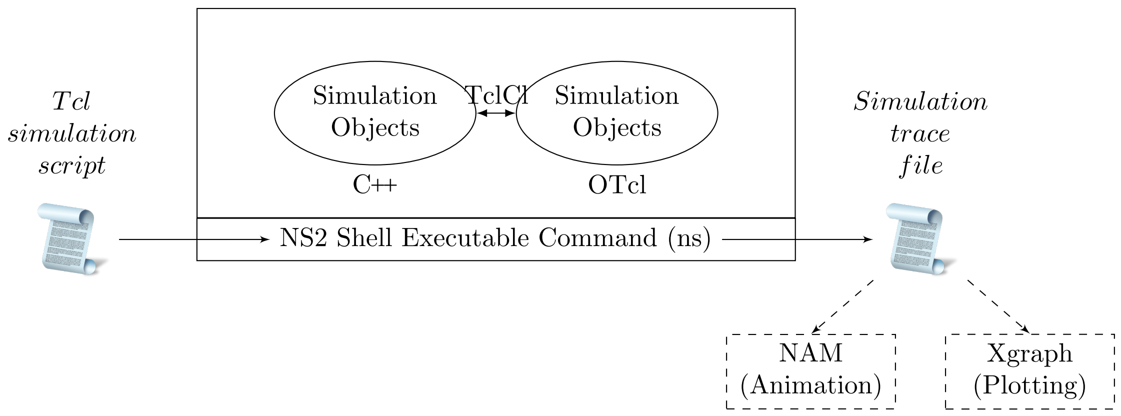


Figure 4.1: NS2 basic architecture

by default so it became important to change the simulator's source code to include it. In order to describe how we implemented this functionality we must first give an outline of the basic C++classes used in NS2. One can examine that in Figure 4.2. We must also note that NS2 does not use a send function, it only uses a receive (**recv**) function in the classes that handle the network traffic. Thus when we create a link between two nodes, NS2 also creates some other objects without us noticing that help traversing the packets through the network. One of the most common objects are *Connectors* which are responsible for packet creation, forwarding, filtering and destruction.

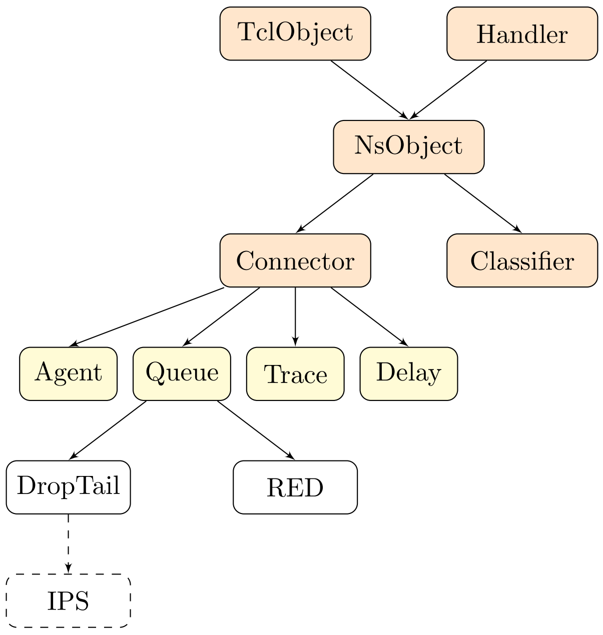


Figure 4.2: NS2 class hierarchy

We came to the conclusion that the most suitable place to do our packet dropping would be in a queue algorithm that is used in a link between two nodes. Since we didn't want to create a totally new algorithm for queue management, we extended the functionality of an already existing one. As shown in Figure 4.2 our queue management algorithm, called for simplicity *IPS*, extends the *DropTail* queue management algorithm. The only change we made in the queue's algorithm was to add some code in the **recv** function to check the received packet's header in order to get the sender's id. Once we acquire the sender's id, we then check whether or not we need to block packets from that node by checking the corresponding **block_me_** boolean attribute in the node. That attribute didn't exist as well and we needed to add it in the *Node* class. We also provide a way of setting that variable for each node using Tcl code in our simulation script file. Instructions about how our queue management algorithm was implemented and how to use it can be found in Appendix A.

The way that our implementation works, we can only have one configuration for all the dropping points in the network but we can have as many blocked nodes and/or dropping points as we want. Also it only works for wired scenarios since for wireless ones we can follow a different and better way that can be used with more functionalities than this one. When it comes to wireless, one can modify a routing algorithm (e.g. AODV) and in the packet handling function he can filter packets not only according to the sender but also to the current node, something that is not easily doable in the wired scenarios. This happens because for wired networks NS2 does not provide an easy way to get the current node's id from the components connected to the node. We leave as future work the implementation of a better and smarter Intrusion Prevention System-like mechanism that will be able to block traffic according to both the packet sender's id and the current node's id.

4.4 Summary

In this section we explained the importance of simulations, and their types. We then gave a brief description of the software we used to perform our experiments and its architecture. Lastly we presented the modification we needed to make to that software so that we could conduct our simulations.

5 Related Work

Applying game theory in cyber warfare can be very informative but the following issues arise as Hamilton, Miller & Ott (2002) mentioned:

- There are limited examples to draw from.
- Players can make multiple, simultaneous moves.
- Opponents are under no time control constraints.
- Opponents may have different end goals from us.
- The set of known legal moves may change during the game.
- Opponent resources and end goals may change during the game.
- Timing for move and state updates is not well defined.

Hamilton, Miller & Ott (2002) also introduced some ways of solving some of those issues, like the usage of null types moves for the multiple move problem, or setting a timespan in which a move can be completed to solve the last issue.

Despite those problems, game theory has been proved to be a powerful tool to help model network security scenarios, more and more researchers use it to create better defence mechanisms. Especially in DoS attack defence mechanisms where we want to model intelligent and rational attackers to make our defences better, game theory is one of the most commonly used techniques to model mechanisms and simulate attack scenarios before actually simulating or even implementing software. What follows is several researches around (D)DoS attacks and defences that used game theory as an analysis tool in order to create better defence mechanisms and note the weaknesses of existing systems.

5.1 Modelling and Simulating

One major issue is how to efficiently model attack scenarios in game theory. In order to give a solution to this problem, Liu, Zang & Yu (2005) presented a methodology that can be used to model DDoS attacks on many different scenarios. Their methodology focuses on the attackers mostly and suggests that is very important to model and infer an attacker's intent, objectives and strategies (AIOS) since it can lead to effective risk assessment. The game model they chose to describe their methodology is an incentive based game theoretic one. Moreover they presented the taxonomy of game-theoretic AIOS models than we can see in Figure 5.1, based on the correlation among attack actions and the agility and accuracy of the defending intrusion detection. In their work, they also included Bayesian modelled experiments with their results, which were simulated using NS-2 (Network Simulator 2). In those experiments the defence strategy chosen is the pushback mechanism. The methodology presented is flexible and can be used to simulate many different types of scenarios, even irrational attackers (something that they did not focus on, on this work).

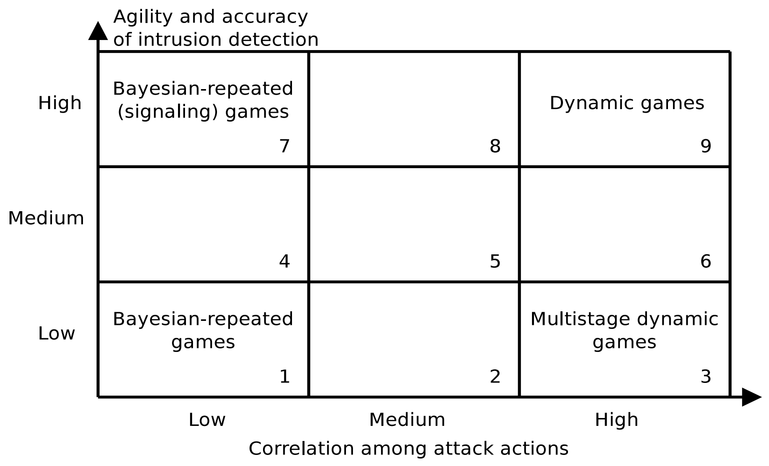

Agility and accuracy of intrusion detection
High
Medium
Low
Bayesian-repeated (signaling) games
7
8
Dynamic games
9
4
5
6
Bayesian-repeated games
1
2
Multistage dynamic games
3
Low
Medium
High
Correlation among attack actions

Figure 5.1: A taxonomy of game-theoretic AIOS models. (From Liu, Zang & Yu (2005))

5.2 Studying DoS using game theory

Many researches have been made that try to model and simulate attacks under game theory. Since an attack can have many factors that affect its outcomes, often a research makes certain assumptions and tries to work with these. The following paragraphs describe some of the most interesting efforts.

Lin, Chen, Chen & Chien (2009) using game theory, try to to solve of problem of both attacker and defender in a network to get the best possible payoffs they can. They modelled the attack using a zero-sum game and in particular they tried to find the solution using the minimax theorem and the saddle point theory. The saddle point theory says that in a zero-sum two-player game, exist optimal strategies for both players. In order to represent the model decisions and the process of the game they used a tree diagram in which nodes represent game states and the links between them the process that brought the system in a state. Each node has also a weight that represents the significance it has for both the attacker and defender. The approach used in this work can get very complicated because of the complexity to create a correct tree for the game. Moreover the cost of accessing and saving the tree can be big since its size can increase a lot (with higher rate when the tree has many nodes) in each new introduced environment variable. Also since there is not standard way of assigning weights in the tree nodes, they tried to assigned weight from the attacker's point of view. That can cause differences in the results compared to having weights from the defender's point of view.

Wu, Shiva, Roy, Ellis & Datla (2010) explore how game theory can apply to cyber security and especially on active bandwidth depletion attacks. Their experiments are for both DoS and DDoS attacks, where the defender's goal is to find the optimum firewall rules to stop the attacker(s), and the attacker's goal is to compromise the network and find the most effective rate or botnet size. The games are modelled as two-player static games and for the simulation Network Simulator

3 is used. In order to be able to satisfy the filtering specification required, they developed a module for NS3 called NetHook, that allows direct access to the packets as they traverse the network stack. The experiments done are only for static games, and the players' payoffs are only measured with respect to the percentage of bandwidth consumed by the legitimate and attacking nodes. Those aspects make this work quite restrictive that requires more extensions and more scenarios to be studied. Also it is mentioned in their work that they would contribute the NetHook module to the NS3 codebase, something that has not happened yet.

In the work presented by Dingankar & Brooks (2007) the simulation of DDoS attacks is modelled as a game in which the defender tries to find an optimum network topology to prevent the attack, whilst the attacker tries to place zombies in the network that he uses to launch the attack on the network links. The game works in a way that each player is allowed one move at each state. The defender starts by selecting a network topology. After than in each round he must select one of the possible reconfigurations for the network. This way he tries to find a "loopy" game, where he can always return to a previous configuration. A "loopy" game is a game in which after making some moves the state of the system is the same as it was at some point in the past. The game presented is a simplistic version of a DDoS attack without taking under consideration many factors. The defender is obliged to select a reconfiguration at each round of the game something that would not always be realistic. Also both players do not have any overheads for their moves. This is something that should be included since the weights of each move should be taken under consideration to create a more analytic and realistic game.

Computer networks can be modelled under certain assumptions as social networks, where nodes collaborate and communicate with each other. Some of the nodes have high connectivity and can be interpreted as routers. Matusitz (2009) combine game theory and social network theory in an effort to model and understand how cyber-terrorism works. The abstract model they present is one where the attacker tries to gain the maximum possible outcome. What is really interesting is that the attacker's gains described have not only to do with the amount of nodes or links that are compromised. The attacker also gains when the defence of the network increases since that produces overhead in the network's communications and provides lower quality of service. This is something that has not been used by any other researchers. It is though a very realistic view since when the defender tries to implement a defence mechanism, an overhead is introduced in the network. That means that the defender cannot keep implementing unlimited defence mechanisms because that would make the quality of service very bad and the network possibly unusable. So a rational defender must also take under consideration that each defence mechanism comes with a certain cost. Thus he must choose the right defence mechanisms when that is necessary and not overdo it.

5.3 Defence mechanisms

As it is also mentioned before, game theory can help in testing of new defence mechanisms as well as in the testing of existing ones. Some of the most interesting defence mechanisms tested using game theory follow in the next paragraphs.

In the approach presented by Walfish, Vutukuru, Balakrishnan, Karger & Shenker (2006), in order to defend against DoS attacks, offence is used as defence. In other words the defence mechanism used is one that when an attacker tries to perform a DoS attack on the network, the defender encourages the legitimate nodes in the network to increase their data sending rate. Under the assumption that the attacker is using already all his available bandwidth whilst the legitimate users are not, the legitimate users will capture a larger fraction of the attacked server's resources, crowning out the attacker. This approach is considered to be effective under certain parameters. Walfish et al. (2006) also try to find and analyse those parameters and also suggest solutions for any occurring issues.

Puzzles are a technique that can be used to reduce flooding DoS attacks and have been of increasing interest in the last few years. Yet they have not been formalised and there exist issues that require solutions. Fallah (2010), using game theory, proposed a series of puzzle-based strategies to reduce the number of DoS attacks. The game used to model the attack scenario was a two-player infinity repeated one. The attacker tries to compromise the network whilst the defender tries to find an appropriate puzzle-based defence to stop the attacker. In this extensive and well presented work, four different defence mechanisms have been introduced, each solving the problems of the previous. The first mechanism can be applied to defend against DoS and DDoS attacks but cannot support the higher payoffs being feasible in the game. The second mechanism resolves the problem of the first but lacks the feature of defending against DDoS attacks. In the third mechanism the feature of defending against DDoS attacks is introduced, assuming that the defender is aware of the attack coalition. Lastly the final form of the defence mechanism resolves all the previous issues and does not require knowledge of the attack coalition. One main technique used to prevent the attacks is a "punishment" phase in which the attacker receives harder puzzles that require more resources. Puzzle defence mechanisms are a promising defence tool against (D)DoS attacks since they require high costs on the attacker's side, something that makes the attack procedure costly. The work presented in this paper is a great effort to model and present efficient puzzle-defence mechanisms despite the overhead that they introduce to the network.

Trust management systems authenticate users' digitally signed credentials using the delegation mechanism. In delegation, each domain is able to determine who can access its resources and the credentials required. A user in order to be verified as a legitimate one, submits a chain of credentials that the trust management system needs to verify. The DoS attacks that can be launched against trust management systems are node resource draining attacks, that try to exploit the expensive verification of the user's credentials from the trust management system. Li, Li, Wang & Yu (2006) studied two existing trust management systems (KeyNote and TrustBuilder) and also proposed a defence technique using credential caching. In the two-player zero-sum game model used the attacker tries to drain as much resources as possible whilst the defender tries to identify the attacker as quickly as possible. Their study came to the conclusion that by the time the defender identifies the attacker, he would already suffer from severe resource draining. The proposal of credential caching made, assigns weights on the credential to efficiently exploit the cache mechanisms. This mechanism is beneficial even if no attacks take place since many legitimate credential chains share common credentials. Moreover this also allows the trust management server to set an upper limit on the size of the credential chains, something that can reduce the workload. One unique characteristic of this defence mechanism is that the more legitimate users that are verified, the quicker can an attacker be identified. This happens because attackers will share credentials with legitimate users and since those credentials will be cached, it will take less time to identify an attacker. As Li et al. (2006) noted for future work, trust management systems are very complicated systems that might have other DoS attack vulnerabilities that need to be explored in order to provide defence mechanisms for them.

5.4 Summary

This section describes some of the relevant work that has been published around our topic. As we saw, game theory can be used to simulate DoS attacks in order to make efficient defence mechanisms or in order to test if a network can suffer from them. In our work we will mostly focus on the second part and try to study as many configurations as possible.

6 Work

During our research we tried to find an efficient and smart model to study Denial of Service attacks using game theory. In this section we describe the problem we are trying to solve and the game theory model that we used for our experimental study.

6.1 Problem Definition

When studying Denial of Service attacks one of the most famous topologies used is the dumbbell topology since it is the one that most server topologies look like in real life. Thus in our model we used the dumbbell topology that can be seen in Figure 6.1.

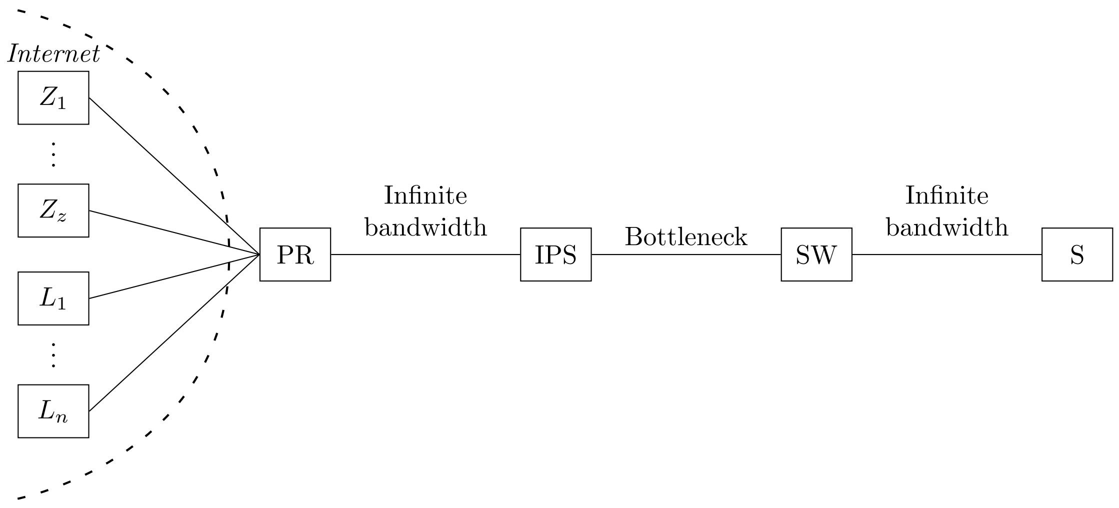


Figure 6.1: Model dumbbell topology

For better understanding we will give an explanation of what Figure 6.1 represents. In the right end is our server denoted with an S. Left to it we have a switch, denoted with SW, and next to it we have a node that hosts our Intrusion Prevention System. The link between the IPS and the switch is a bottleneck (meaning that it has less bandwidth available than the other links in the internal network) ans is thus subject to a denial of service attack. Left to the IPS we have a perimeter router that simply forwards the traffic from the internet to the internal network. For simplicity reasons we suppose that the bandwidth of the links connecting the perimeter router with the IPS, and the one connecting the switch with the server is infinite. In the left side of the figure we have the internet world which consists of legitimate nodes (denoted by $L_i, i \in [1, n]$ where n is the number of legitimate nodes) that want to use our server and from attacking nodes (denoted by $Z_j, j \in [1, z]$ where z is the number of attacking nodes) that want to perform a denial of service attack in our network.

6.2 Game theory model

In this section we will give the definition of our game model that is based on the model proposed by Wu et al. (2010) but with some different factors and requirements.

Suppose we have the topology shown in Figure 6.1. We have n legitimate users, where each one is denoted as $L_i, i \in [1, n]$. The legitimate users send traffic to the server, each one at a specific constant flow rate chosen by a normal distribution i.e. $F_i^l \sim \mathcal{N}(f_l, \sigma_l^2)$, where f_l is the mean flow rate and σ_l^2 the standard deviation for the legitimate users.

We model the attacker as an individual who takes control of a number of devices, denoted with z, making them act like his zombie nodes and we name each zombie node as $Z_j, j \in [1, z]$. Furthermore we suppose that the attacker defines the flow rate for each zombie node in a way similar to the legitimate users. Thus each zombie node's flow rate is given by a normal distribution i.e. $F_j^z \sim \mathcal{N}(f_z, \sigma_z^2)$, where f_z the mean flow rate and σ_z^2 the standard deviation for the zombie nodes.

We use B to describe the bandwidth of the bottleneck and T_B to model the average flow rate in the bottleneck where T_B is calculated as

$$T_B = \sum_{i=1}^{n} F_i^l + \sum_{j=1}^{z} F_j^z$$

In the case that $T_B > B$ we experience a denial of service attack in the bottleneck. We furthermore suppose that we choose B in such a way that $B > T_B^l$ with high probability, where T_B^l is the average flow rate in the bottleneck from the legitimate users.

In the case of a denial of service attack and considered that the network works in a fair way towards all the users, only a fraction y of each flow will pass through the bottleneck, meaning that $(1-y)$ will be dropped. We can compute y as $y = \frac{B}{T_B}$ and also we can compute the fraction that will pass through the bottleneck of each flow using $y \cdot f_q, q \leftarrow \{z, l\}$. Also we can compute the average bandwidth consumption for the attacker in the bottleneck and the lost users due to the congestion using the following equations:

$$v_B = \frac{z \cdot y \cdot f_z}{B} = \frac{z \cdot f_z}{T_B} = \frac{z \cdot f_z}{n \cdot f_l + z \cdot f_z} \tag{1}$$

$$v_U = P[F_i^l < \frac{n \cdot f_l + z \cdot f_z}{B}] \tag{2}$$

Now that we have given the basic requirements and definitions of our game model, let us further explain it. We set up a two-player, one-shot, non-cooperative, zero-sum game in which the attacker's goal is to increase v_B and v_U. It is a two-player game since we suppose that we have one attacker that controls all the attacking nodes (zombies) and one defender in the inner network. Also it is one-shot since we suppose that from the moment the players choose their strategies they are bound to them and cannot change them until the end of the game. Lastly it is non-cooperative and zero-sum since as Lin et al. (2009) mentions information warfare games are between an attacker and the administrator and ones player's gain is the other's loss in most cases. That holds in the case that we don't take under consideration personal factors, like for example personal satisfaction or bragging for the attacker and money cost that a denial of service attack brings to the defender.

The attacker's strategy consists of three factors. The number of zombies he will use, the mean and the standard deviation for their flow rate. We suppose on the other hand that the defender uses a defence mechanism (e.g. an Intrusion Prevention System) that given a threshold t drops traffic from the flows whose flow rate is above that threshold. Thus the defender's strategy consists of only setting that threshold.

In case that the defender does not use the defence mechanism we can get the attacker's payoff using the following equation:

$$V = w_B \cdot v_B + w_U \cdot v_U - w_Z \cdot z \tag{3}$$

where w_B, w_U and w_Z are weight coefficients for the bandwidth consumption and zombie creation accordingly. Since the game is zero-sum, the defender's payoff is given by $-V$.

In case that the defender uses the defence mechanism then we need to change the way that we calculate the payoff function. Since some flows will be cut out by the defence mechanism, we need to calculate the new average flow rate for both the zombies and legitimate users that will pass through the defence mechanism. We do so using the two following equations:

$$f'_z = f_z \cdot P[F^z_j < t]$$

$$f'_l = f_l \cdot P[F^l_i < t]$$

where, $P[x < y]$ is the probability that the random variable x is smaller than a number y. If we now substitute the new values in equation (1) we can compute the average bandwidth consumption in the bottleneck by the attacker as follows:

$$v'_B = \frac{z \cdot f'_z}{n \cdot f'_l + z \cdot f'_z} \tag{4}$$

Similarly by substitution in equation (2) and defining that F'^l_i is a random variable from the normal distribution used by the legitimate users but using f'_l as mean, we get:

$$v'_U = P[F'^l_i < \frac{n \cdot f'_l + z \cdot f'_z}{B}] \tag{5}$$

Lastly if we substitute from equations (4) and (5) in equation (3) we get the attacker's payoff in the presence of the defence mechanism as follows:

$$V = w_B \cdot v'_B + w_U \cdot v'_U - w_z \cdot z \tag{6}$$

6.3 Other distributions

In terms of experimentation we found interesting to test what would happen in case that the attacker uses a different distribution than the one the legitimate users use. Thus we tested how the model's results would differ in the case that the attacker uses a Normal, a Poisson or an Exponential distribution to set the flow rates for the zombie hosts. In the case of Poisson and Exponential distributions the attacker does not have to choose a standard deviation, only the mean flow rate. Using this extension we do not need to change anything in the way that the model and the previously stated equation are defined.

Since that approach gives us more interesting results and more comparisons we adopted it and tried to see how the model changes with each different distribution.

6.4 Summary

In this section we described the game model that we propose, the difficulties we came across and how we solved them. Moreover we defined an extension to our model that we adopted and studied for which we give the results in the following section.

7 Work analysis

So far we have given the necessary background information about our study area and our proposed model. In terms of analysis from the defender's perspective it is crucial to find Nash equilibria in our model. The reason is because those strategies that are characterised by a Nash equilibrium are the optimum defending in case of an attack from a rational attacker. In this section we present the results of our research and experiments and we analyse them.

7.1 Studying environment set up

For the purposes of this research we used Matlab to perform the calculations regarding the game theory model. Since we wanted to make choices for t,z,f_z and σ_z^2 we tried several values for them and performed the payoff calculations for every combination. Since is becomes unnecessary to try values that exceed some limits we set some boundaries in the testing values in regard to the model configuration.

Our main model configuration is as follows. We suppose that there are $n = 20$ legitimate users whose flow rates derive from a normal distribution with mean flow rate $f_l = 50$Kb and standard deviation $\sigma_l^2 = 20$. When it comes to the network we suppose that the bottleneck's bandwidth is 2Mb. Lastly we set the following values for the weight coefficients: $w_B = w_U = 1000$ and $w_Z = 10$ since we suppose that we value the network's efficiency highly and the zombie creation cost is relatively low.

Moreover for this model we set the following limits for our tests: $0 \leqslant t \leqslant 110$, $1 \leqslant z \leqslant 120$, $1 \leqslant f_z \leqslant 130$ and $10 \leqslant \sigma_z^2 \leqslant 120$, with $\sigma_z^2 = \kappa \cdot 10$, where $t, z, f_z, \sigma_z^2, \kappa \in \mathbb{N}$ and the network related values are being expressed in Kb.

After performing all the required calculations, we looked for saddle points that relate to Nash equilibria in every possible configuration in our test samples. We must note that in the following figures the saddle points that have been chosen for the normal distribution are those that have the most "balanced" payoff between the two users, meaning the points whose payoff is closest to zero. Also except of the saddle points it was interesting to look for the strategies that lead to the maximum payoffs for each player and of course what would happen in case we tried to change somehow our default configuration. Having acquired the results from the game theory model we performed simulations to test how our model corresponds to reality.

7.2 Nash equilibria

As we have already mentioned, we try to find Nash equilibria in our model since they are very important as they provide the best strategy for the defender under an attack. In this subsection we analyse the results we got and present graphs that show the existence, or not, of Nash equilibria under some specified configurations.

Lets start with an example and see how the payoffs are represented in plots under each distribution for the attacker. Consider the configuration presented previously and that $f_z = 60$. In Figure 7.1 we see the values of V under specified z

and t when the attacker uses a Normal distribution with $\sigma_z^2 = 20$. Similarly Figures 7.2a and 7.2b represent the values of V for when the attacker uses a Poisson and an Exponential distribution respectively. For all three figures, the pointed spot represents a Nash equilibrium under the specified configuration.

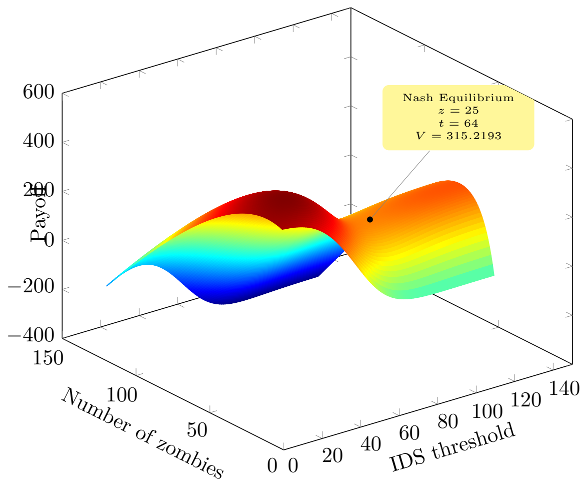
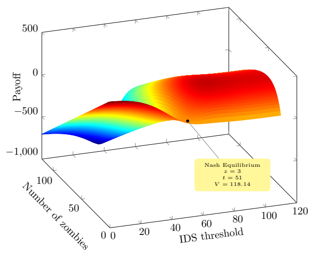
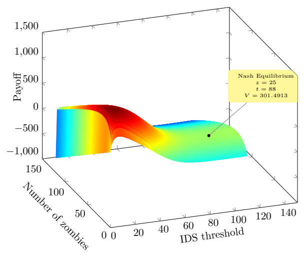


Figure 7.1: Payoffs of Normal distribution for $f_z = 60$ and $\sigma_z^2 = 20$



(a) Poisson



(b) Exponential

Figure 7.2: Payoffs for Poisson and Exponential distributions for $f_z = 60$

In Figure 7.3 we can see a comparison, regarding the game payoff, between three different distributions that the attacker uses. As we can see, when the attacker uses a normal distribution saddle strategies start making their appearances when $f_z \geqslant 11$ with the standard deviations that appear in Figure 7.4. When it comes to using an exponential distribution, saddle strategies start appearing when $f_z \geqslant 23$ and continue until $f_z = 130$ which is the limit of our model testing, the same as the normal distribution saddle strategies. On the other hand when we use a Poisson distribution the saddle strategies appear in a smaller fraction of our test area which is when $29 \geqslant f_z \geqslant 74$.

In terms of which distribution gives the biggest payoffs for the attacker when using a saddle strategy, we notice that when both Poisson and Exponential distribution start appearing Nash equilibria, the Poisson distribution leads to bigger

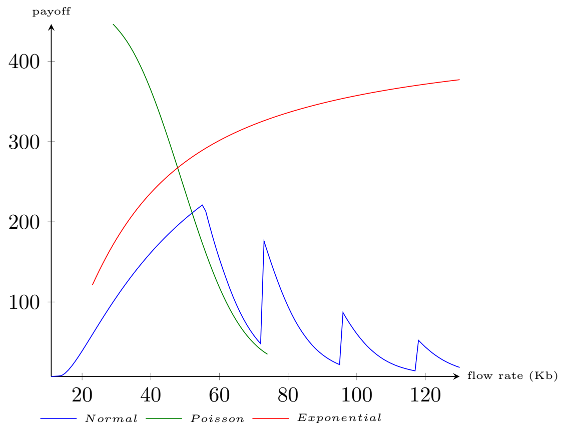


Figure 7.3: Nash equilibria payoffs

payoffs for $f_z \leqslant 47$ while for $f_z > 47$ the Exponential distribution does so. Using a normal distribution leads to less payoffs compared to the other distributions except when $52 \leqslant f_z \leqslant 74$ that the Poisson distribution appears even less payoffs than the Normal distribution and when $f_z \leqslant 22$ where the other distributions appear no Nash equilibria at all.

In general what we can extract, is that it would be better for the attacker to use a mixed strategy where he chooses the distribution that gives him the best payoffs. In this occasion it would be something like the following (where D the attacker's chosen distribution):

$$D = \begin{cases} \mathit{Normal} & \text{if } f_z < 23 \\ \mathit{Poisson} & \text{if } 29 \leqslant f_z \leqslant 47 \\ \mathit{Exponential} & \text{if } 23 \leqslant f_z \leqslant 28 \text{ or } f_z > 47 \end{cases}$$

In terms of the defender's point of view though, the ideal choices for the attacker, giving the optimum payoffs for the defender, would be the following:

$$D = \begin{cases} \mathit{Normal} & \text{if } f_z < 52 \text{ or } f_z > 74 \\ \mathit{Poisson} & \text{if } 52 \leqslant f_z \leqslant 74 \end{cases}$$

In Figures 7.5 and 7.6 we present the choices for the threshold values and the number of zombies respectively that led to the Nash equilibria that appear in Figure 7.3. As we can see when using a Poisson distribution both the threshold and the number of zombies that the players choose to lead them in a Nash equilibrium, have to be relatively smaller that those for the other distributions. On the other hand, when using an Exponential distribution, both the threshold and the number of zombies are larger compared to the other distributions.

What is of great interest though is that for the Normal distribution, the threshold drops about 50 points when $f_z \approx 55$ and $\sigma_z^2 = 10$, where the attacker uses a distribution with similar characteristics with the legitimate users and is hard to

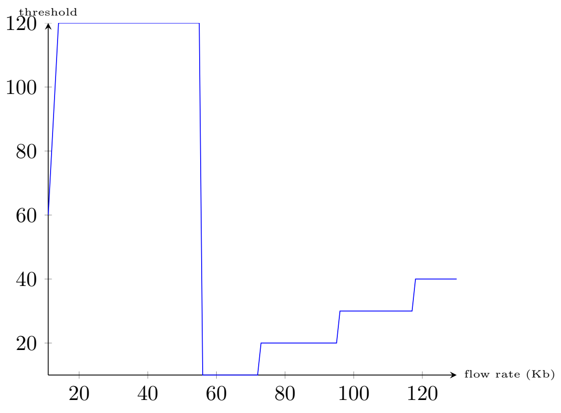


Figure 7.4: Nash equilibria standard deviations

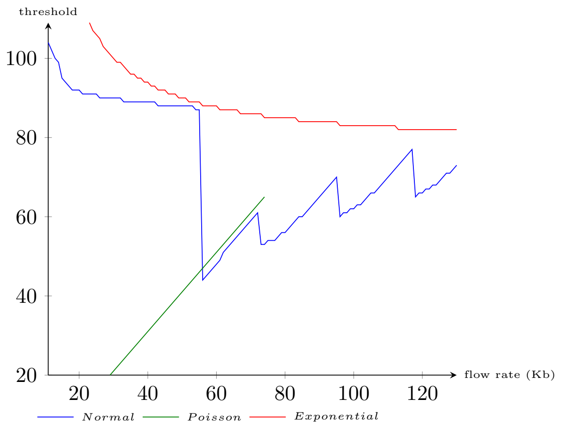


Figure 7.5: Nash equilibria threshold value

be detected. It is in that point that the defender sets a relatively low threshold to try and block as much zombie hosts whilst allowing as much legitimate ones as possible.

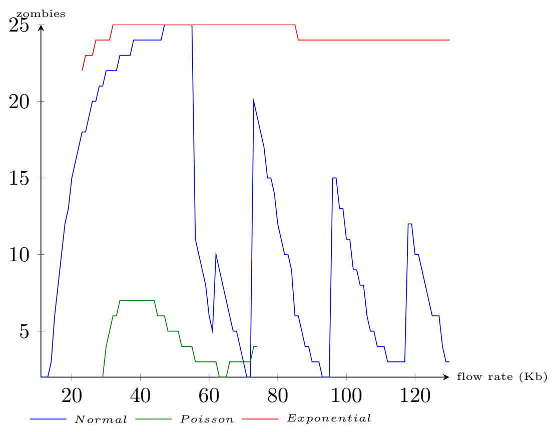


Figure 7.6: Nash equilibria number of zombies

7.3 Number of legitimate users

Another interesting aspect to explore is how the number of legitimate users in the network alters the presence of Nash equilibria in our model. Using our basic model configuration we gave as input 6 different values for the number of legitimate users.

In Figure 7.7 we show how the number of legitimate users affects the presence of Nash equilibria when the attacker uses a normal distribution. As we can spot, the lines follow similar paths in each case. One observation that can be made is that, Nash equilibria start making their appearances for higher f_z as the number of legitimate users increases. Also, increasing the number of legitimate users leads to less payoff for the attacker, who seems to achieve the higher gains when the network only has a few legitimate users in a Nash equilibrium configuration.

Figure 7.8 demonstrates the same results but when the attacker uses a Poisson distribution. As we can see the curves are very close one to the other, gathered in a small area, and the payoffs for the attacker are about the same in every occasion with a slight deviation in f_z. Something that is noticeable and of most interest is that not all the curves are continuous, but some have missing values in some points, meaning that for that f_z under that specific number of legitimate users we cannot have a Nash equilibrium. The missing values are more when we have 5 or 10 legitimate users whilst the most Nash equilibrium values appear when there are 20 legitimate users in the network, who on average would consume half the bottleneck's bandwidth under our basic model configuration. What we can make out of it is that when having a small amount of legitimate users, the defender can easier affect the network's state while when we have a larger amount, we have better network efficiency and the attacker would require a lot of zombie hosts to cause damage to the network.

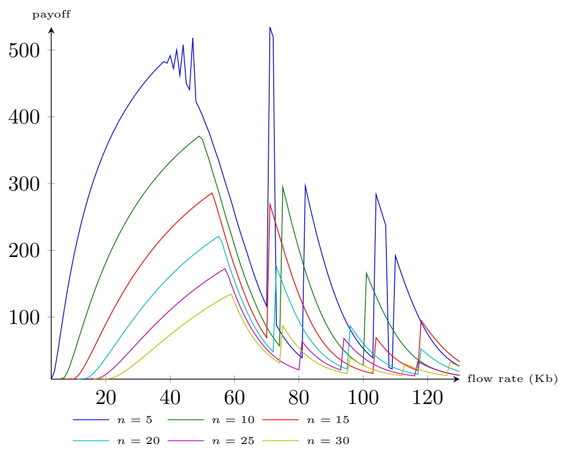


Figure 7.7: Scaling legitimate users for Normal distribution

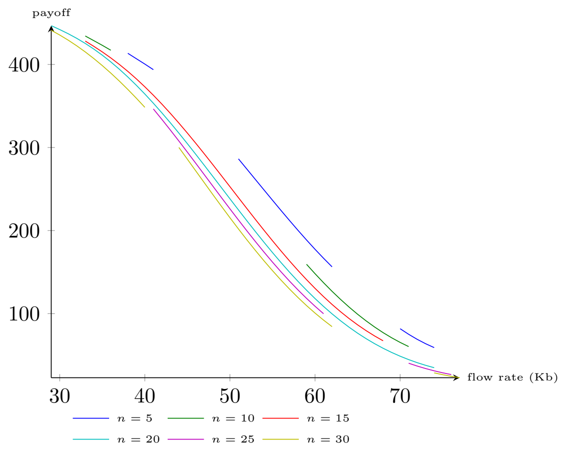


Figure 7.8: Scaling legitimate users for Poisson distribution

Lastly Figure 7.9 shows the results for when the attacker uses an Exponential distribution. As we can see in this occasion Nash equilibria appear on average slightly after $f_z > 20$ regardless the number of legitimate users and keep on appearing as we increase f_z. The only difference that we can spot is that increasing the number of legitimate users in the network, reduces the attacker's payoff from the attack. Thus from the defender's point of view, it is better to keep the network constantly busy with legitimate users and/or choose the networks capacity according to the needs of the legitimate users.

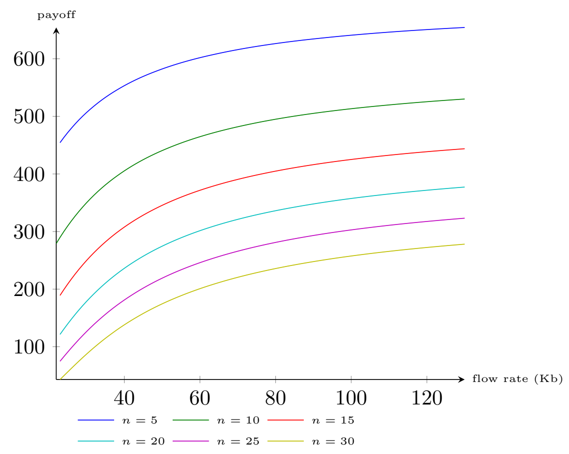


Figure 7.9: Scaling legitimate users for Exponential distribution

7.4 Choosing attacker's standard deviation

As we have mentioned earlier, in our plots for an attacker that uses a Normal distribution we present the results of the most saddle strategies, meaning the results that are closest to zero. What happens though when the attacker uses another σ_z^2? Figure 7.10 represents the existence of Nash equilibria under different values of σ_z^2.

As we can see, increasing σ_z^2 leads to more Nash equilibria appearances regarding the flow rates. When the attacker uses a small σ_z^2, in our case less than 40, Nash equilibria appear in a shorter range of f_z and for $\sigma_z^2 \leqslant 20$ the payoffs are larger when Nash equilibria first appear and decrease later. On the contrary when $\sigma_z^2 > 20$ the payoffs start out small for $f_z \approx 15$ and reach their peak on average when $f_z > 80$. Moreover we observe that it is better for the attacker to use a small σ_z^2 since it provides him with the highest payoffs while increasing it gives him approximately on average at most $V = 260$.

Interpreting those results, we see that when the attacker uses a larger σ_z^2, it becomes easier to find saddle strategies since he would have to spread out the flowrate values more and especially if he uses a small amount of zombies then the defence has surely the advantage. This is the case because the defender then would suffer from less legitimate users being cut out due to the value of t and he would

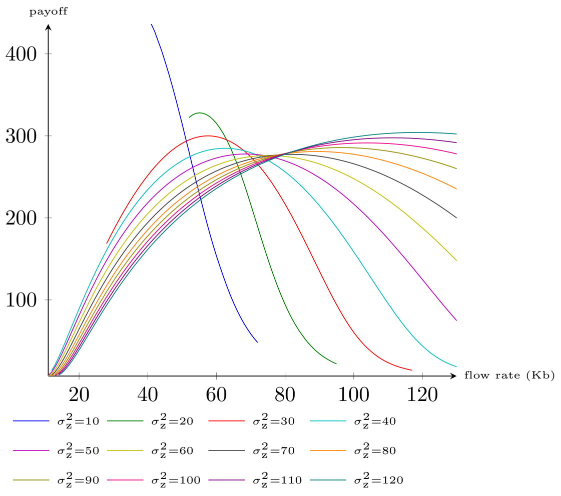


Figure 7.10: Nash equilibria for Normal distribution under different σ_z^2

have probably less zombies in his network.

Regarding which value of σ_z^2 brings better results for the defender and attacker we can look at Figure 7.11 that represents the average payoff for each value of σ_z^2 for various values of σ_l^2 in our model and Figure 7.12 that illustrates the maximum and minimum payoffs that occur under each value of σ_z^2 for $\sigma_l^2 = 20$.

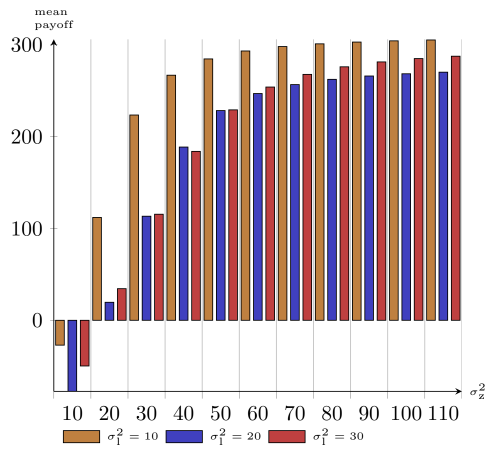


Figure 7.11: Mean payoffs for each σ_z^2 for a Normal distribution

As we can see in general the maximum and minimum payoffs are about the same except from the first two values of 10 and 20 in which we have less maximum and bigger minimum payoffs. This is interesting because we can conclude that when the attacker uses a smaller σ_z^2 (or at least smaller or equal to σ_l^2), he gives the leverage to the defender since the latter can easier block more zombies due to the small range of their flow rates. This also appears to be the case if we look at the means for each σ_z^2. When $\sigma_z^2 = 10$ the defender is the one who has positive average payoff while for any other value the attacker is the one with the positive average payoff. As we can also see, the average payoff increases for the attacker faster for $\sigma_z^2 < 60$ while afterwards the increase makes not too much of a difference.

This also applies for the different values of σ_z^2 in Figure 7.11 with slight differences. For example having a smaller σ_l^2 leads to faster increase of the attacker's payoff (which eventually gets larger than for other σ_l^2 values) and is in general worse for the defender. On the other hand having a larger σ_l^2 leads to slower increase in the attacker's payoff but it still is higher on average that for $\sigma_l^2 = 20$.

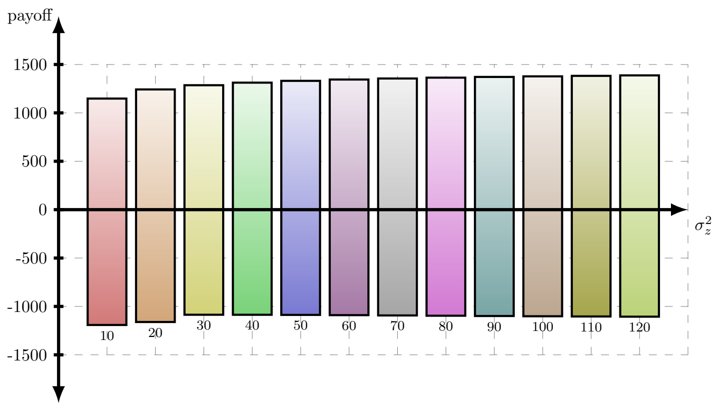


Figure 7.12: Minimum and maximum payoffs for each σ_z^2 for a Normal distribution

From those we can conclude that the defender can probably bare with an attacker that uses a small σ_z^2 but it becomes harder to defend against an attacker with a large σ_z^2 without cutting out several legitimate users. Furthermore, we understand that σ_l^2 plays also a crucial role in our model and there is no general rule (like bigger is better) that can be applied to that. In each configuration its optimum value varies and has to be calculated.

7.5 Affect of weight coefficients

An important part in our payoff equation are the weight coefficients that we choose each time. In other words how much do we care about the bandwidth consumption, the number of lost legitimate users and how much does it "cost" for the attacker to create a zombie host.

Lets see how does the zombie host creation cost affects our model for each distribution. Figures 7.13, 7.14 and 7.15 show the Nash equilibria appearances for using a Normal, Poisson and Exponential distribution respectively, with some discrete values for the zombies creation cost.

The first observation that we can make is that increasing the cost of creating a zombie host means that the attacker's payoff drops, which was expected. As we can see though the differences are less when the attacker uses a Poisson distribution whereas they are larger for a Normal or Exponential distribution. The differences are not always large in Normal and Exponential distributions, but only with the

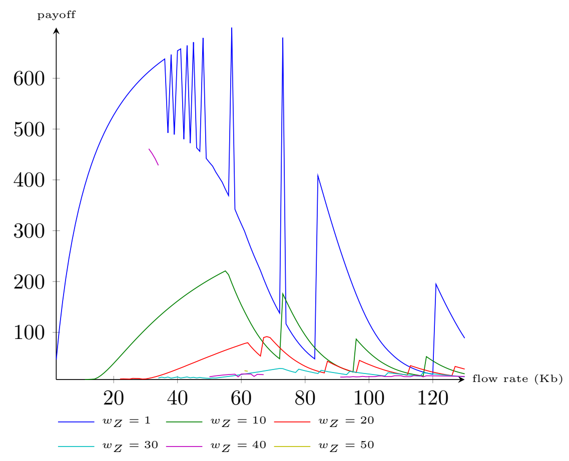


Figure 7.13: Nash equilibria varying zombie creation cost (Normal distribution)

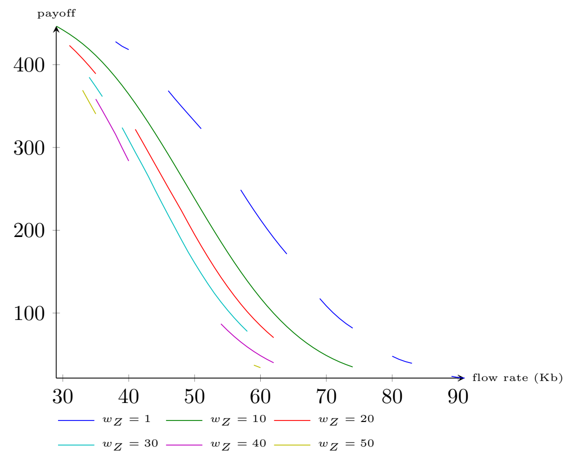


Figure 7.14: Nash equilibria varying zombie creation cost (Poisson distribution)

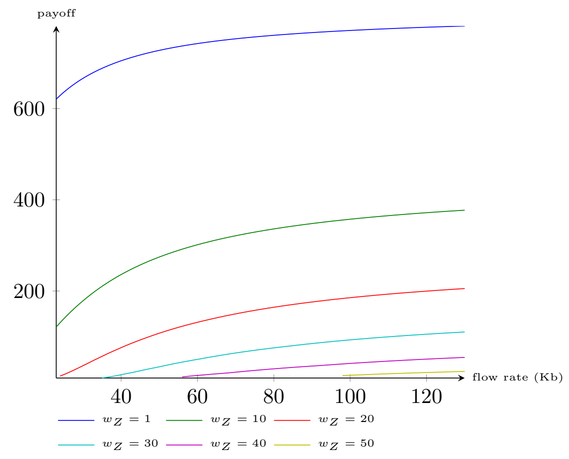


Figure 7.15: Nash equilibria varying zombie creation cost (Exponential distribution)

smaller values of w_Z. As w_Z increases the differences start getting smaller for those distributions as well.

Another thing to mention is that again we have missing values in our plots for some pairs of w_Z and f_z meaning that no Nash equilibria exist under those pairs. For a Normal distribution we can see that when w_Z is 40 or 50 plenty of missing values appear, with 50 having almost none Nash equilibria appearances. Looking at Figure 7.14 considering the Poisson distribution, we observe that for $w_Z = 10$ we have the less missing values, whereas as we increase w_Z more missing values appear. For the Exponential distribution what we can say is that when w_Z increases, the minimum f_z for the Nash equilibria to start making their appearances increases as well.

Considering the minimum f_z that Nash equilibria start appear for each distribution, we notice that for the Poisson distribution, Nash equilibria start making their appearances in lower f_z values when we increase w_Z whilst for the other distributions happens the exact opposite.

7.6 Simulation results

An important component of our research is the validation of our model's efficiency. In order to do so, we use simulations in NS2 with our implementation of an IPS like mechanism.

For our experiments we set the simulation time to 300 seconds. Each legitimate user starts transmitting traffic at a random time moment m_l such that $0 \leqslant m_l \leqslant 1$. Similarly each zombie node starts its transmission at a random time moment m_z such that $40 \leqslant m_z \leqslant 41$. All zombie nodes stop their transmission at the 260th second whereas the legitimate ones at the 299th. In order to acquire our results we repeat our simulations ten times with different values for both legitimate and zombie hosts flow rates each time that we get from Matlab using the *poissrnd*[4],*exprnd*[5] and *normrnd*[6] functions. Out of those ten runs we get the average values. That way we increase our sample and reduce the noise from our results.

Figure 7.16 illustrates the percentages of received legitimate traffic over time in a network which follows our basic configuration. The strategies followed be the attacker and defender are the following: $z = 50$, $f_z = 60$, $\sigma_z^2 = 20$ and $t = 100$. Furthermore there are $n = 20$ legitimate users having mean flow rate $f_l = 50$ and standard deviation $\sigma_l^2 = 20$. Lastly the bottleneck's bandwidth is set to 2MB. As we can see when the defender uses an exponential distribution, he manages to do less damage to the network while using either a Poisson or Normal distribution drops the received legitimate packets percentage to half.

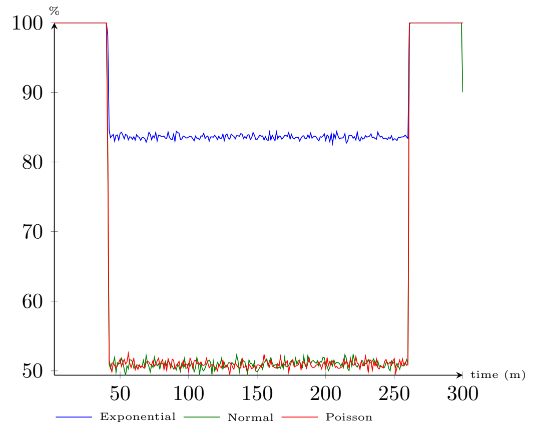


Figure 7.16: Percentages of received legitimate traffic

In order to acquire a better understanding of the affect of the players' strategies and how they appear in the simulation we present the results for each distribution the attacker can use separately starting with the normal distribution.

7.6.1 Normal distribution

Lets first observe how changing only σ_z^2 can affect our network and the legitimate users. Figure 7.17 represents the percentage of the legitimate traffic that was received by the server under various values of σ_z^2 where the configuration is the same with the one we stated earlier ($z = 50$, $f_z = 60$, $t = 100$). Furthermore Figure 7.18 shows the percentage of the received traffic by the server that is legitimate for the different values of σ_z^2. As we can see in both figures, increasing σ_z^2 above 20 leads to less damage to the network's performance. That is what we expected and is logical since increasing the standard deviation for the attacker's distribution leads to more zombie hosts that will be blocked by the installed IPS. Although the increase in the network's performance is not very significant by increasing only σ_z^2.

[4]`http://www.mathworks.com/help/stats/poissrnd.html`
[5]`http://www.mathworks.com/help/stats/exprnd.html`
[6]`http://www.mathworks.com/help/stats/normrnd.html`

The percentage of legitimate traffic that reaches its destination increases about 5% when we increase σ_z^2 above 20 and the ratio of received traffic that is legitimate only by about 2% as we see in Figures 7.17 and 7.18. Thus only changing the value of σ_z^2 cannot lead to much difference regarding the network's bandwidth consumption by the legitimate users but it can lead to a reasonable difference when it comes to the percentage of the legitimate traffic that reached its destination.

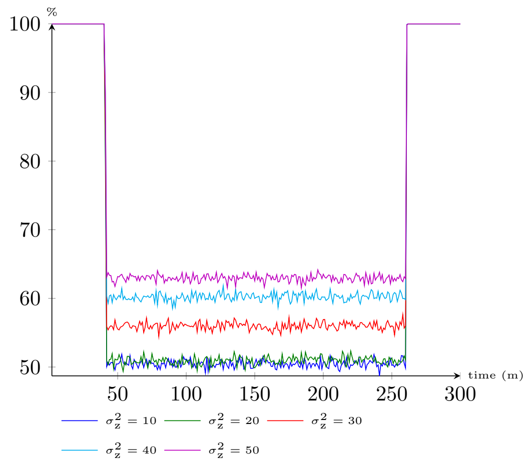


Figure 7.17: Received legitimate traffic percentage for various σ_z^2 values

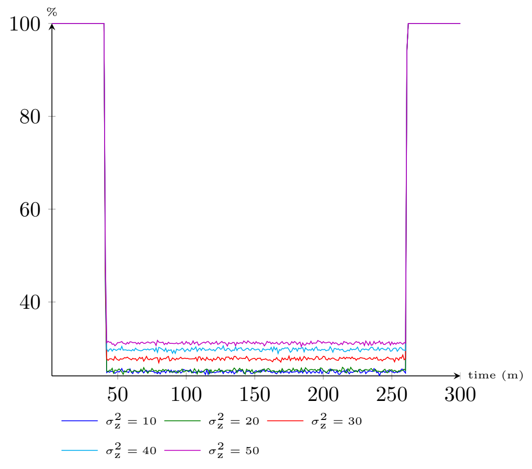


Figure 7.18: Ratio of received traffic that is legitimate for various σ_z^2 values

Although the previous results are significant, most times the attacker will not change only the standard deviation he uses, but also the number of zombie hosts. Figure 7.19 pictures the results for 5 different simulations where we keep fixed the attacker's mean flow rate with $f_z = 50$ and the threshold that the defender user with $t = 88$. In each simulation we try a different duple of σ_z^2 and z. We choose those values in order to show how the network is affected when we try something different than a Nash equilibrium set of strategies. The blue line represents a Nash equilibrium strategy duple for our network with $\sigma_z^2 = 120$ and $z = 25$ apart from the previously mentioned values.

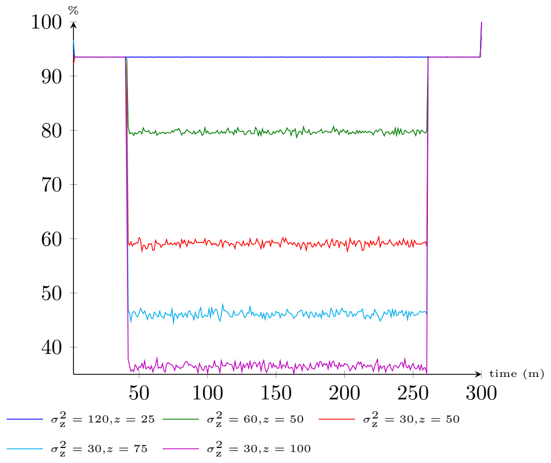


Figure 7.19: Saddle strategy and attacker's variations for $f_z = 50$ (Normal distribution)

What we tried first was to keep the number of zombies fixed and decrease σ_z^2. In the green line we halved it ($\sigma_z^2 = 60$) and in the red one we set its value to one quarter ($\sigma_z^2 = 30$) of the Nash equilibrium one. For the next two simulations we kept σ_z^2 at 30 (one quarter of Nash equilibrium value) where for the cyan line we increased the number of zombies by 50% ($z = 75$) and for the magenta one by 100% ($z = 100$) compared to the Nash equilibrium value. As we can see decreasing σ_z^2 in this configuration gave us much more difference compared to the previous examples. Truly when $\sigma_z^2 = 30$ and the zombie hosts are the same as in the saddle strategy duple, we get a 35% units decrease in the legitimate packets that reach their destination. Increasing the number of zombie hosts moreover gives us further decrease, reaching about a 60% units decrease in the legitimate traffic that reaches the server. Surely if we kept increasing the number of zombies we could have reached a situation where the legitimate traffic reaching the server would be <1%, which is about the case in most real world Denial of Service attacks and also the most common end goal of an attacker.

Under the same configurations, Figure 7.20 shows the ratio of legitimate traffic out of the whole received traffic. As we can see, when using the Nash equilibrium strategy duple, zombie hosts consume about 30% of the network's bandwidth. That percentage increases significantly once we set σ_z^2 to half and reaches ∼60% while when we set σ_z^2 to one quarter of the Nash equilibrium strategy's duple value, it reaches ∼70%. This means that simply by reducing his standard deviation, the attacker managed to swap his network bandwidth consumption with the one of the legitimate users. After that, increasing the number of zombie hosts, keeps increasing their bandwidth consumption, but less dramatically (∼5% in each configuration).

As we can easily guess, the great difference between the Nash equilibrium duple strategies and the configurations with the reduced standard deviation, is due to the fact that either less zombie hosts will now be blocked by the IPS or that the zombie hosts that will not be blocked will have higher flow rates. In either occasion, the damage that is produced is large enough.

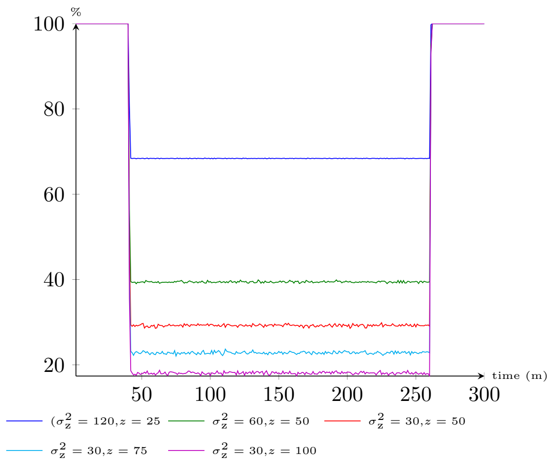


Figure 7.20: Ratio of received traffic that is legitimate for $f_z = 50$ varying attacker's strategy (Normal distribution)

So far we presented results when we try different strategies from the attacker's point of view. Now we are going to present results regarding the actions the defender can take. We continue with the presented configuration and now we fix the attacker's variables and only change the IPS's threshold. That is we have $f_z = 50, z = 25$ and $\sigma_z^2 = 120$. In Figure 7.21 we present the percentage of legitimate packets that reached the server and in Figure 7.22 the ratio of legitimate traffic out of the whole received traffic, where for $t = 88$ we have the Nash equilibrium strategy duple. Increasing t to 98 or 108 allows all legitimate traffic to reach the server but it also allows zombie hosts to consume more and more bandwidth, where for $t = 98$ zombies consume ~76% and for $t = 108$ ~62% of the bottleneck's bandwidth.

Decreasing t leads to less bandwidth consumption by zombie hosts but also to much less legitimate traffic reaching the server. For example if we take $t = 58$, while bandwidth consumption by zombies is reduced by ~7%, the legitimate traffic reaching the server is reduced by ~42% in comparison to the Nash equilibrium strategy duple values. This happens since by decreasing t we cut out legitimate users and because both legitimate and zombie hosts have the same mean flow rate ($f_l = f_z = 50$) it becomes even more difficult to block zombies and not legitimate users as well.

What we can conclude is that the defender can easily set up his configuration in order to cause significant damage to the network. Especially when he tries to mimic the legitimate hosts' behaviour, the defender's job becomes very hard as it becomes inevitable to block zombie hosts without doing so for legitimate ones as

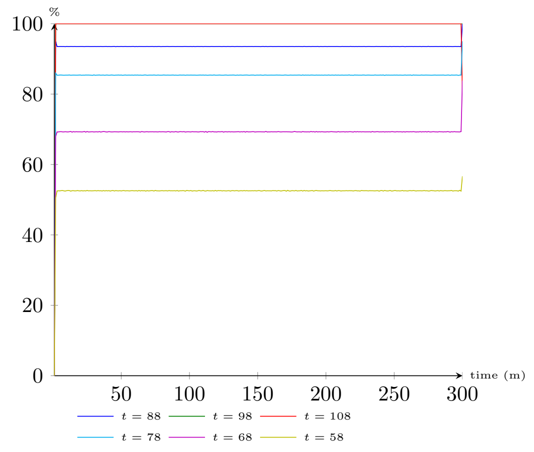


Figure 7.21: Saddle strategy and t variations for $f_z = 50$ (Normal distribution)

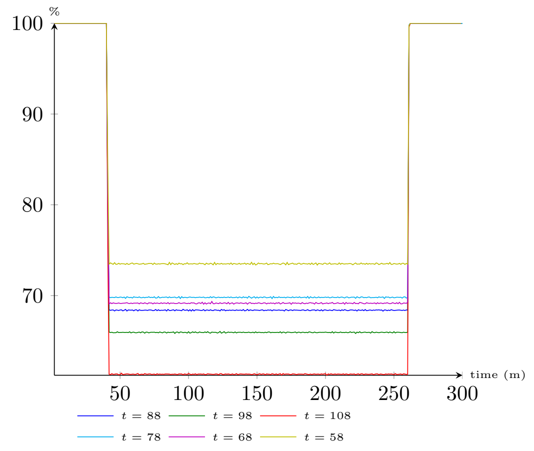


Figure 7.22: Ratio of received traffic that is legitimate for $f_z = 50$ varying t (Normal distribution)

well. Also since when using a normal distribution, the attacker has to set also the standard deviation he will use, even if he is bound to use a specific mean flow rate, he can set his standard deviation in the value that causes the maximum damage possible.

7.6.2 Poisson distribution

In Poisson distribution the attacker does not have to set a standard deviation so we can only see how the rest of the variables change the results of the simulations. We start with testing the affection of the number of zombies. As before we define our saddle strategy and see how results are affected when we deviate from it. For $f_z = 50$ the Nash equilibrium appears when $t = 41$ and $z = 5$. All the other variables are the same as we stated in the beginning of the subsection 7.6.

Figures 7.23 and 7.24 illustrate the percentage of legitimate traffic that reaches the server and the ratio of the received traffic that is legitimate accordingly. As we can see, increasing the number of zombie hosts even 200% compared to the Nash equilibrium value, did not make much difference to the percentage of legitimate traffic reaching the server. As we can see that percentage is always low due to the fact that the threshold is set to a relatively low value leading to the blocking of plenty legitimate users apart from the zombie hosts. Regardless the blocking of legitimate users we say that the threshold value is a good choice in terms that the attacker has to use a relatively large number of zombies to cause damage to the underlying network.

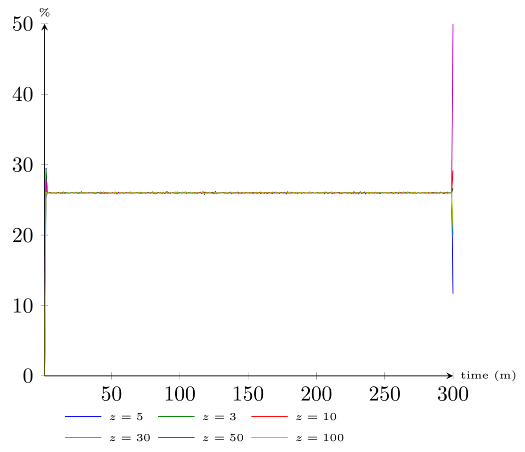


Figure 7.23: Percentage of legitimate traffic reaching the server (Poisson distribution)

On the other hand, as it is obvious, increasing the number of zombies led to a decrease in the percentage of legitimate traffic that was allowed through the IPS. That doesn't mean necessary that the legitimate users suffer from more packet drops due to the network congestion though, as the number of allowed hosts and their flow rates in most cases are not enough to fill the bottleneck's bandwidth. Nevertheless the efficiency of the link is reduced since the amount of bogus traffic

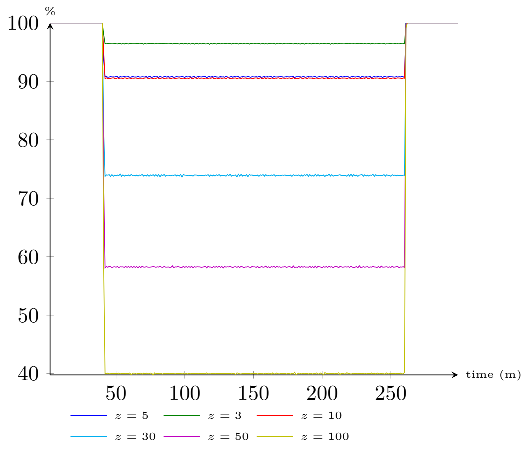


Figure 7.24: Ratio of received traffic that is legitimate (Poisson distribution)

increases alongside the increase in the number of zombie hosts.

Now let us explore what happens for different threshold values. In Figures 7.25 and 7.26 we give our results for seven different configurations. Those are the following:

- Nash equilibrium strategy ($z = 5$ and $t = 41$)
- Threshold variations ($t \in \{41, 50, 55\}$) for $z = 50$
- Threshold variations ($t \in \{41, 50, 55\}$) for $z = 100$

As we can see in Figure 7.25 no matter the number of zombies, the percentage of legitimate packets reaching the server is the same for the Nash equilibrium threshold value. Increasing the threshold to 50 though shows some differences in the results. When we have 50 zombies, the percentage increases to about 42%, since more legitimate users are allowed through the IPS. We can see also that when the zombie traffic starts, no changes appear in the percentage of received legitimate traffic meaning that the allowed zombies are not enough to cause congestion in the bottleneck. Increasing the amount of zombies to 100 though reduced the percentage of legitimate traffic received as long as the zombie hosts send traffic as well. This means that the allowed zombie hosts are starting to cause congestion in the bottleneck and in our case they manage to reduce the legitimate reception percentage by ∼9%.

Increasing the threshold to 55 changes the results even more since more hosts will be allowed through the IPS. We can see that the percentage of legitimate traffic reaching the server when no zombie hosts transmit is about 50%. When the zombie hosts start their transmission though we suffer a decrease in that percentage. When the attacker uses 50 zombies the decrease is ∼5% but when he uses 100 zombies the decrease reaches ∼25% allowing only one quarter of the whole legitimate traffic to reach the server.

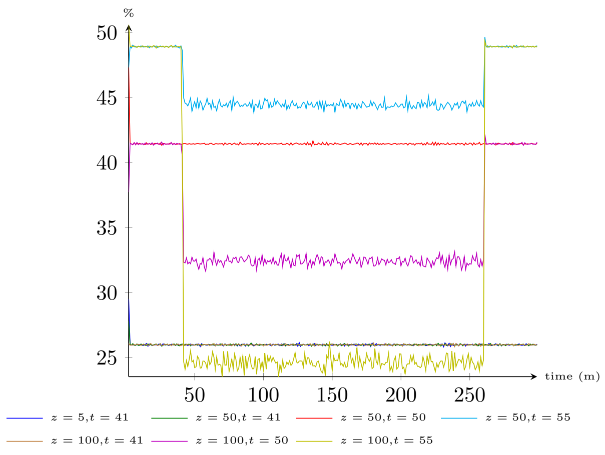


Figure 7.25: Percentage of legitimate traffic reaching the server (Poisson distribution)

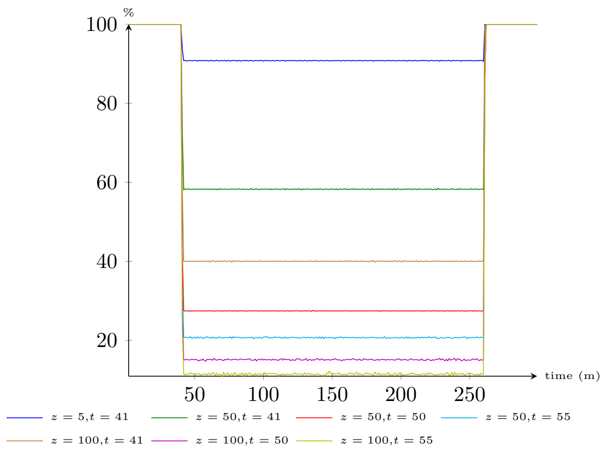


Figure 7.26: Ratio of received traffic that is legitimate (Poisson distribution)

What we can easily say is that when the attacker uses a Poisson distribution, the defender when playing saddle in the case that they both use the same mean flow rate, cannot block easily zombies without also denying access to legitimate users. Also the defender is more vulnerable when the attacker uses a Poisson distribution, as in case the threshold is not set appropriately, causing congestion in the bottleneck becomes only a matter of the number of zombies hosts used by the attacker.

7.6.3 Exponential distribution

Again in the exponential distribution the attacker has only to set the number of zombies and their mean flow rate. In the following analysis again we study the difference in the network that appears due to the chosen threshold and number of zombie hosts keeping the attacker's mean flow rate the same as with the legitimate users' one. For the use of an exponential distribution the saddle strategy for $f_z = 50$ is that $t = 90$ and $z = 25$.

First we test how the number of zombie hosts affects the network's performance. We tried a configuration for the saddle strategy, one with fewer zombies and three with more. In Figures 7.27 and 7.28 we can see the ratio of legitimate traffic that reached the server and the ratio of the received traffic that was legitimate accordingly. As we observe when the attacker uses 25 or less zombie hosts, he does not cause congestion in the bottleneck but only reduces the network's efficiency with bogus data. Increasing the number of zombies to 50 and above leads to zombie traffic being the dominant one in the network and causing congestion in the bottleneck. That congestions leads to packet drops reducing the percentage of the legitimate traffic reaching the server to ~55% when 100 zombie hosts are being used.

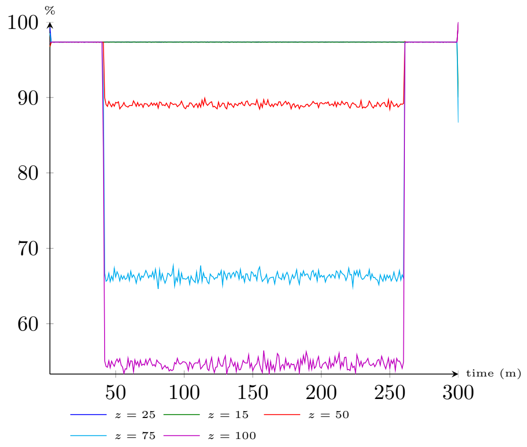


Figure 7.27: Ratio of legitimate traffic reaching the server (Exponential distribution)

As we can see using an exponential distribution is not the best option in terms of reducing the percentage of legitimate traffic reaching the server to low values as

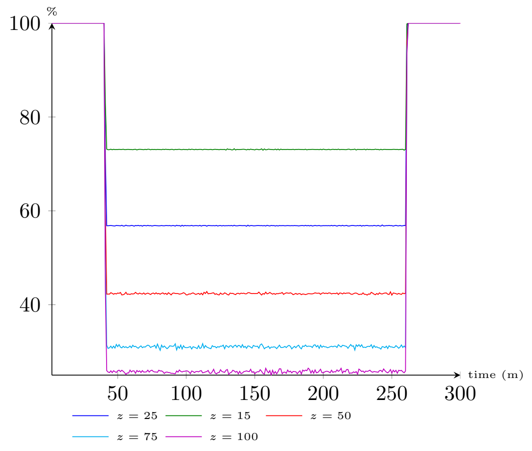


Figure 7.28: Ratio of received traffic that is legitimate (Exponential distribution)

a large number of zombie hosts would be required. For example in our tests that number would be probably above 200 zombie hosts for a 2MB bottleneck link. Of course that is in favour of the defender as when it comes to an attacker of that type, he can set a relatively high threshold and still block out plenty of zombie hosts without having problems of congestion in the bottleneck.

Regarding the threshold's value we can see in Figures 7.29 and 7.30 its affection on the ratio of received legitimate traffic and in the ratio of the received traffic that is legitimate accordingly. As we can see the Nash equilibrium strategy does not give a 100% legitimate traffic reception rate but slightly less without suffering from further loss when the zombie hosts start their transmission.

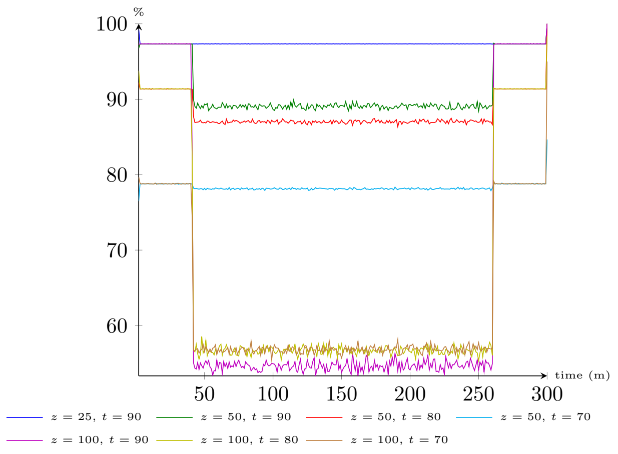


Figure 7.29: Ratio of legitimate traffic reaching the server (Exponential distribution)

Increasing the number of zombies to 50 gives us a clearer look on the results since it causes congestion in the bottleneck. Having set t to 90 again gives us $\sim$98% legitimate traffic reception when no zombie traffic is transmitted whilst

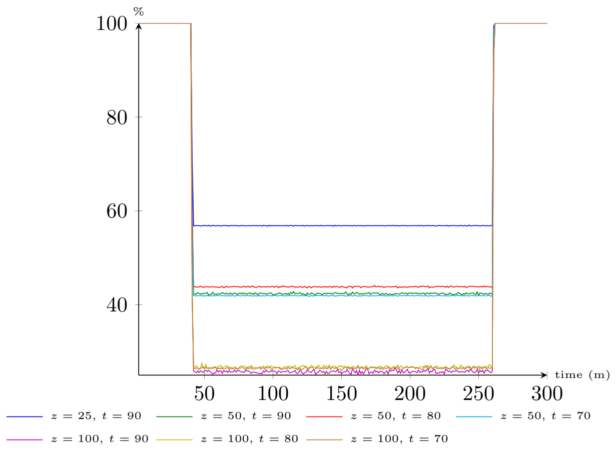


Figure 7.30: Ratio of received traffic that is legitimate (Exponential distribution)

that percentage drops to ~89% when the zombie hosts start their transmission. Reducing t to 80 gives us a ~92% legitimate traffic reception without zombie traffic while the percentage drops to ~87% when the zombie hosts start their transmission. Lastly setting t to 70 reduced further the legitimate traffic reception without zombie traffic which now is ~89%. When the zombie hosts start their transmission though, the percentage drops only by ~1%. Of course in this situation we see that it is better to keep the threshold in a high value ($t = 90$) since even when the zombie hosts start transmitting, we have slightly worse quality of service than even having $t = 80$ and no zombie traffic.

In the case of 100 zombie hosts, when $t = 90$ the received legitimate traffic drops significantly at ~52%. Similarly for both $t = 70$ and $t = 80$ the percentage reaches ~55%, slightly worse that for $t = 90$ (from the attacker's point of view). That gives us to think that, for larger numbers of zombie hosts, having a higher threshold leads to easier bottleneck congestion while having a lower one not only leads to a close enough congestion rate to the high one, but also the IPS would refuse access to plenty legitimate users.

Defending against an attacker that uses an exponential distribution can become tricky. That is because even if the defender decreases the threshold in a value such that $t \geqslant f_l + \sigma_l^2$, the attacker will cause relatively the same damage in the network when he uses a large number of zombie hosts. On the other hand setting the threshold to a lower value would be far more devastating for the network's efficiency. What we can take away is that for an attacker using an exponential distribution to set the flow rates of his zombie hosts, is easier to exploit the threshold's value and cause damage to the network but yet he requires a large number of zombies (probably > 200) to reduce the network's efficiency below 1%.

7.6.4 Summary

To sum up, in this subsection we presented our results regarding the simulations we performed. For reasons of clarity, we examined each distribution that the attacker can use individually. From our tests we learned that the best option for the attacker in terms of reducing the legitimate users' quality of service, is to try and mimic their configuration as much as possible making it harder for the defender to come up with a threshold value that can block zombie hosts without doing the same for legitimate users. Thus the best choice for the attacker is to use a normal distribution with options close to the ones that have been used to set up the legitimate users' flow rates.

7.7 Summary

In this section we presented the work we performed considering our game theory model and how some coefficients can change its outcomes, especially when it comes to the weight coefficients about the attacker's and defender's costs. We also tried to find out whether Nash equilibria exist under different configurations and the values for which they do. Lastly we analysed our simulation results and showed how it is better for an attacker to mimic the legitimate users' configuration making the defender's choice of threshold much more difficult.

8 Evaluation

The objectives of this project were to analyse Denial of Service attacks with Game Theory and to validate the results by performing simulations. In our work we managed to create a Game Theory model that has not been examined so far in the existing bibliography. We assumed a defender that has in his disposal an Intrusion Prevention System capable of dropping flows whose flow rate exceeds a specified threshold. On the other hand we assumed that the attacker spawns a number of zombie hosts whose flow rates derive from either a Normal, a Poisson or an Exponential distribution.

One clear advantage in the game theory model we presented is that it can be easily adapted and studied for different configurations. For example one might take into consideration other gains or costs for the attacker or defender and easily format our payoff formula to compute the results. Moreover the study of a different distribution or what happens when the legitimate users' distribution is not a Normal one can easily be performed by slight modifications in our model. Thus our proposal is flexible and well adapted.

Sadly we were bound to study static games since the research of a dynamic game would be extremely time consuming and the given time for the dissertation's completion was not enough to explore it (by one student at least). Despite the fact that we didn't manage to perform the research of such a game, we present an analysis of how the model can be modified for a dynamic game in section 9. Also another aspect that required time beyond the available was researching how multiple attackers that spawn different groups of zombie hosts can affect the model's results. Again we present how this can be modelled and studied in section 9.

The second part of our objectives, including the verification part, was performed using the Network Simulator 2 simulation software. Since an option of simulating an IPS like mechanism was not available in NS2, we created our own mechanism which is described in section 4 and the implementation details are provided in Appendix A. The module we created though is not optimal since it is highly connected to the queue algorithm that is to be used in the network's links. The reason we followed that route was because in NS2 it is difficult in some portions of the code to get the packets that traverse the node and to acquire the current node's id since the connection between each node and the components attached to it is defined only in the node itself. The optimum solution would be the IPS implementation to be either a *Classifier* or *Application*. In that way some restrictions that apply now, such as the fact that all the IPSs in the system share the same options, would be dealt with. Despite those limitations, our implementation works great allowing us to successful conduct our experiments under the configurations we specified.

Splitting the project in three parts gave us the opportunity to focus first on the implementation of the module we required and afterwards, having set up the foundations for the simulations, start working on our model. Having both the simulation foundations and the game theory model we then proceeded in conducting our simulations to verify our game model.

9 Further work

The proposed game theory model is such that allows plenty extensions that can be easily adapted in it. Moreover one could also perform extensions in the NS2 integration. In this section we explain what those extensions are in terms of model extensions, NS2 implementation and general extensions.

9.1 A dynamic game

The model as we have defined it is a static, one-shot one. One possible and interesting extension would be to change the nature of the game to a dynamic one. A dynamic game is considered as a sequence of time steps. In each step each player chooses a strategy that may or may not reflect to the history of the game (according to what we want the players to know after each round e.g the payoff only or the full information). Thus the game evolves through time and the players adapt to the new information.

Under those assumptions we can model different types of attacking and defending strategies that can exploit the continuous strategy changes. For example lets suppose that the defender chooses his strategy in each round based on the history up to 5 rounds. Then an attacker can try to exploit that by setting z high and f_z low for a few steps, making the defender set t low, and then set f_z high and z low for the following few steps that the defender will be vulnerable due to his history dependence. Of course there are many different scenarios that one can take under consideration. For example we can say that both the players look for bigger long term payoffs or that they use some kind of learning rule to adjust automatically to each step.

Considering the payoffs for each player we have to say that in each step, the payoffs will need to adapt according to the history so far. Furthermore the total payoff for the attacker will be given by $V = \sum V_t$ where V_t is the payoff for the t step. The payoff for the defender will be $-V$ again as we suppose that the game is zero sum. In case that the game is of infinite horizon (the steps are not finite), then we say that the attacker's payoff is $V = \sum \lambda_t \cdot V_t$, where λ_t is a weight parameter with values $0 \leqslant t \leqslant 1$ for each step that defines its significance. The λ parameter follows a decreasing or increasing format according to whether we value more the sort term or the long term steps. The reason why λ is important is in order to avoid getting an infinite payoff. Consider that the payoff in each round is fixed e.g. 5. Then, V would be

$$V = \sum_{t=1}^{\infty} 5 = \infty$$

But in case we use λ, with $\lambda_t = 0.5 \cdot \lambda_{t-1}$, $\lambda_1 = 1$ and we only care about two points precision, then we will get

$$V = \sum_{t=1}^{\infty} \lambda_t \cdot 5 \approx 9.99$$

thus allowing us to easier evaluate the value of the game.

9.2 Multiple attacking groups

Another extension would be to allow the attacker to create groups of zombies with different characteristics in each one of them. That means that each group can have a different distribution, mean flow rate and/or standard deviation where applicable. For example consider an attacker that creates two groups of 5 and 3 zombie hosts accordingly with the first group to be characterised by a Poisson distribution with mean 50 and the second by an Exponential distribution with mean 70. Lets define the number of groups that the attacker chooses to use by $G = G_{\mathcal{N}} + G_{\text{Pois}} + G_{\text{exp}}$, where $G_{\mathcal{N}}$ the number of groups using a Normal distribution, G_{Pois} the number of groups using a Poisson distribution and G_{exp} the number of groups using an Exponential distribution. Furthermore we define with z_i^g the number of zombies in each group, with f_{iz}^g the mean flow rate for each distribution and with $\sigma_{iz}^{\mathcal{N}}$ the standard deviation for each group with a normal distribution, where $g \in \{\mathcal{N}, \text{Pois}, \text{exp}\}$. Keeping those in mind the attacker has to set a varying number of attributes as input to the model according to the values of $G_{\mathcal{N}}$,G_{Pois} and G_{exp}. The number a of those attributes is given by

$$a = 3 \cdot G_{\mathcal{N}} + 2 \cdot (G_{\text{Pois}} + G_{\text{exp}}) + 3$$

meaning that the total attributes required for the model analysis are $a+1$ including the threshold that the defender needs to set, making the analysis of the model computationally expensive since we would have a $(a+1)$-dimensional problem to solve.

With those modifications in mind we need to also change the previously stated equations. Thus, in the presence of an IPS, v'_B and v'_U will be calculated as follows:

$$v'_B = \frac{\left(\sum_{i=1}^{G_{\mathcal{N}}} z_i^{\mathcal{N}} \cdot f'^{\mathcal{N}}_{iz} + \sum_{i=1}^{G_{\text{Pois}}} z_i^{\text{Pois}} \cdot f'^{\text{Pois}}_{iz} + \sum_{i=1}^{G_{\text{exp}}} z_i^{\text{exp}} \cdot f'^{\text{exp}}_{iz} \right)}{n \cdot f'_l + \left(\sum_{i=1}^{G_{\mathcal{N}}} z_i^{\mathcal{N}} \cdot f'^{\mathcal{N}}_{iz} + \sum_{i=1}^{G_{\text{Pois}}} z_i^{\text{Pois}} \cdot f'^{\text{Pois}}_{iz} + \sum_{i=1}^{G_{\text{exp}}} z_i^{\text{exp}} \cdot f'^{\text{exp}}_{iz} \right)}$$

$$v'_U = P[F'^l_i < \frac{n \cdot f'_l + \left(\sum_{i=1}^{G_{\mathcal{N}}} z_i^{\mathcal{N}} \cdot f'^{\mathcal{N}}_{iz} + \sum_{i=1}^{G_{\text{Pois}}} z_i^{\text{Pois}} \cdot f'^{\text{Pois}}_{iz} + \sum_{i=1}^{G_{\text{exp}}} z_i^{\text{exp}} \cdot f'^{\text{exp}}_{iz} \right)}{B}]$$

Also we need to take under consideration that each attacker might have different costs for the creation of zombie hosts. Thus the total zombie host creation cost that would be used in the modified equation (6) (page 31), where w_Z^{gi} the weight coefficient for attacking group i of distribution g, would be given by :

$$Z = \sum_{i=1}^{G_{\mathcal{N}}} z_i^{\mathcal{N}} \cdot w_Z^{\mathcal{N} i} + \sum_{i=1}^{G_{\text{Pois}}} z_i^{\text{Pois}} \cdot w_Z^{\text{Pois}\, i} + \sum_{i=1}^{G_{\text{exp}}} z_i^{\text{exp}} \cdot w_Z^{\text{exp}\, i}$$

9.3 NS2 multiple IPS implementation

During our work we came across several people that needed the same mechanisms as we did to do their own projects. In our work we required only one IPS point in our network but what happens if one needs more than one IPSs in a network with different configurations each? In this case the code that we used to simulate an IPS in NS2 is not adequate and another module has to be implemented. Since NS2 (and also its newer version NS3) lack the functionality of an IPS-like mechanism, such a project would be a great opportunity to offer to an open source project and its community.

To give an outline of what the project would be like thing the following. We have a network like the one we used in our research only with two IPS systems where for example the one is behind the other in order to have two zones of safety and allow access to some resources while refusing it to others. One must then create a module/application for NS2 that can be attached to a node and act like an IPS, meaning that the way that we perform filtering would be different since we want each IPS to have a different configuration. Thus we must be able to set in each IPS node from which nodes we want traffic to be dropped and not having a flag in each node saying if we must drop traffic from it or not.

Another thing that also can be implemented is to allow two different types of configuring the IPS. The first would be to allow traffic from all nodes by default and block only the ones specified (the route we followed). The other one, which is quite the opposite, would be to block all nodes by default and allow traffic to pass by from only the specified nodes.

9.4 Summary

In this section we presented features and extensions to our current model that would be interesting for our further research. Of course one could also try combinations of all or some of those and see where that leads. The possibilities are many and the possible scenarios that can be studied even more since game theory provides a great tool for efficient study of rational models regarding Denial of Service attacks and many other areas of science of course.

References

Abliz, Mehmud. 2011. Internet Denial of Service Attacks and Defense Mechanisms. Technical report University of Pittsburgh.

Bajaj, Sandeep, Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, Padma Haldar, Mark Handley, Ahmed Helmy, John Heidemann, Polly Huang, Satish Kumar, Steven McCanne, Reza Rejaie, Haobo Yu, Ya Xu & et al. 1998. Virtual InterNetwork Testbed: Status and Research Agenda. Technical report.

Batishchev, Alexander M. 2009. "LOIC (Low Orbit Ion Cannon).".
URL: *http://sourceforge.net/projects/loic/*

Braden, R. 1989. "Requirements for Internet Hosts - Communication Layers.".

CERT/CC. 1996. "TCP SYN Flooding and IP Spoofing Attacks.". CERT Advisory CA-1996-21.
URL: *http://www.cert.org/advisories/CA-1996-21.html*

CERT/CC. 1997. "Denial of Service.".
URL: *http://www.cert.org/tech_tips/denial_of_service.html*

Chang, Di-Fa, Ramesh Govindan & John Heidemann. 2002. An empirical study of router response to large BGP routing table load. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment.* IMW '02 New York, NY, USA: ACM pp. 203–208.
URL: *http://doi.acm.org/10.1145/637201.637233*

Compton, Stuart & Charles Hornat. 2007. "802.11 Denial of Service Attacks and Mitigation.".

daemon9, route & infinity. 1996. "IP-spoofing Demystified (Trust-Relationship Exploitation)." *Phrack Magazine* 7.

Dingankar, C. & RR Brooks. 2007. Denial of Service Games. In *Proceedings of the Third Annual Cyber Security and Information Infrastructure Research Workshop.* pp. 7–17.

Douligeris, Christos & Aikaterini Mitrokotsa. 2004. "DDoS attacks and defense mechanisms: classification and state-of-the-art." *Computer Networks* 44(5):643 – 666.
URL: *http://www.sciencedirect.com/science/article/pii/S1389128603004250*

Fallah, M. 2010. "A Puzzle-Based Defense Strategy Against Flooding Attacks Using Game Theory." *IEEE Transactions on Dependable and Secure Computing* 7(1):5–19.
URL: *http://ieeexplore.ieee.org/ielx5/8858/5408788/04459338.pdf?tp=&arnumber=4459338&isnumber=5408788*

Hamilton, SN, WL Miller & Allen Ott. 2002. "Challenges in applying game theory to the domain of information warfare." *4th Information survivability* .
URL: *http://www.cert.org/research/isw/isw2001/papers/Hamilton-31-08-a.pdf*

Handley & Rescorla. 2006. "RFC 4732 - Internet Denial-of-Service Considerations." RFC 4032 (Experimental).
URL: *http://tools.ietf.org/html/rfc4732*

Joncheray, L. 1995. A simple active attack against TCP. In *Proceedings of the Fifth Usenix Unix Security Symposium.* pp. 7–19.

Kargl, Frank, Joern Maier & Michael Weber. 2001. Protecting web servers from distributed denial of service attacks. In *Proceedings of the 10th international conference on World Wide Web.* WWW '01 New York, NY, USA: ACM pp. 514–524.
URL: *http://doi.acm.org/10.1145/371920.372148*

Li, Jiangtao, Ninghui Li, XiaoFeng Wang & Ting Yu. 2006. Denial of Service Attacks and Defenses in Decentralized Trust Management. IEEE pp. 1–12.
URL: *http://ieeexplore.ieee.org/ielx5/4198788/4198789/04198805.pdf?tp=&arnumber=4198805&isnumber=4198789*

Lin, Jin-Cherng, Jan-Min Chen, Chou-Chuan Chen & Yu-Shu Chien. 2009. A Game Theoretic Approach to Decision and Analysis in Strategies of Attack and Defense. In *Secure Software Integration and Reliability Improvement, 2009. SSIRI 2009. Third IEEE International Conference on.* pp. 75 –81.

Liu, Peng, Wanyu Zang & Meng Yu. 2005. "Incentive-based modeling and inference of attacker intent, objectives, and strategies." *ACM Trans. Inf. Syst. Secur.* 8(1):78–118.
URL: *http://doi.acm.org/10.1145/1053283.1053288*

Matusitz, Jonathan. 2009. "A Postmodern Theory of Cyberterrorism: Game Theory." *Information Security Journal: A Global Perspective* 18(6):273–281.
URL: *http://www.tandfonline.com/doi/abs/10.1080/19393550903200474*

Mavituna, Ferruh. 2008. "DoS attacks using SQL wildcards.".

Mccanne, S., S. Floyd & K. Fall. N.d. "NS2 (Network Simulator 2)." http://www-nrg.ee.lbl.gov/ns/.
URL: *http://www-nrg.ee.lbl.gov/ns*

Mirkovic, Jelena & Peter Reiher. 2004. "A taxonomy of DDoS attack and DDoS defense mechanisms." *SIGCOMM Comput. Commun. Rev.* 34(2):39–53.
URL: *http://doi.acm.org/10.1145/997150.997156*

Newman, R. 2009. *Computer Security: Protecting Digital Resources: Protecting Digital Resources.* Jones & Bartlett Learning.

Osborne, Martin J. & Ariel Rubinstein. 1994. *A course in game theory.* The MIT Press.

Rose, MT. 1991. “Convention for defining traps for use with the SNMP.”.

Scarfone, K. & P. Mell. 2007. “Guide to Intrusion Detection and Prevention Systems (IDPS).” *NIST Special Publication* 800(2007):94.
URL: *http://csrc.nist.gov/publications/nistpubs/800-94/SP800-94.pdf*

Shirey, R.W. 2007. “Internet Security Glossary, Version 2.”.

Shor, Mikhael. 2001 - 2006. “Dictionary of Game Theory Terms.”.
URL: *http://www.gametheory.net/dictionary/*

Stallings, W. 1987. *Handbook of computer-communications standards; Vol. 1: the open systems interconnection (OSI) model and OSI-related standards.* Macmillan Publishing Co., Inc.

Turocy, Theodore L & Bernhard Von Stengel. 2001. “Game Theory*.”.

Vixie, P., G. Sneeringer & M. Schleifer. 2002. “Events of 21-oct-2002.”.
URL: *http://c.root-servers.org/october21.txt*

Walfish, Michael, Mythili Vutukuru, Hari Balakrishnan, David Karger & Scott Shenker. 2006. DDoS Defense by Offense. In *ACM SIGCOMM 2006.* Pisa, Italy: .

Wu, Qishi, Sajjan Shiva, Sankardas Roy, Charles Ellis & Vivek Datla. 2010. On modeling and simulation of game theory-based defense mechanisms against DoS and DDoS attacks. Spring Simulation Multiconference 2010, SpringSim'10 Orlando, FL, United states: China Welding p. Society for Modeling and Simulation International.
URL: *http://dx.doi.org/10.1145/1878537.1878703*

A Network Simulator 2 Intrusion Prevention System-like mechanism implementation

In this appendix we give details of how we implemented our packet dropping queue management algorithm and how it is used in practice. First of all we must create the required attribute in the *Node* class that determines if traffic from the particular node will be blocked or not. To do that we first need to change the header file (*node.h* under common directory) and declare the variable that denotes if the node is to be blocked and also a function that will return us that value. So we need to go under `class Node : public ParentNode` and add

```
inline bool block_this() { return block_me_; }
```

in the public section and the following code in the protected section.

```
bool block_me_;
```

After that we need to set the variable's initial value in the constructor and also create a way for the user to set its value using Tcl code. So we edit the node's implementation file (*node.cc* under common directory) and we place under `Node::Node():` the following to set the initial value to **false** (we don't want to block traffic by default).

```
block_me_(false)
```

Now in order to create a tcl interface, we go under the `int Node::command(int argc, const char* const* argv)` function and more precisely in the case where we have two arguments and add the following code in the argument checking code.

```
else if(strcmp(argv[1], "block-me") == 0) {
  this->block_me_ = true;
  return TCL_OK;
} else if(strcmp(argv[1], "unblock-me") == 0) {
  this->block_me_ = false;
  return TCL_OK;
}
```

This way we provide two commands that the user can execute in the Tcl script to either block traffic from that node or not.

Now that we are done with the node's parameters we have to create the queue management algorithm. So what we have to do is create two files under the queue directory. We name those files *ips-queue.h* and *ips-queue.cc* accordingly. The code for *ips-queue.h* can be seen in Listing 1 and the one for *ips-queue.cc* in Listing 2.

Listing 1: IPS Queue header file

```
#include <string.h>
#include "drop-tail.h"

class IPS: public DropTail {
public:
  IPS() {
    // Do nothing, just call parent constructor
  }
  ;
  ~IPS() {
    // Do nothing, just call parent deconstructor
  }
  ;
protected:
  void enque(Packet*);
};
```

Listing 2: IPS Queue Implementation

```
#include "ips-queue.h"
#include "node.h"
#include "ip.h"

static class IPSClass: public TclClass {
public:
  IPSClass() :
      TclClass("Queue/IPS") {
  }
  TclObject* create(int, const char* const *) {
    return (new IPS);
  }
} class_ips;

void IPS::enque(Packet* p) {
  if (summarystats) {
    Queue::updateStats(qib_ ? q_->byteLength() : q_->length());
  }

  // Drop traffic from blocked nodes
  int source_addr = (int) (HDR_IP(p)->saddr());
  Node* s = Node::get_node_by_address(source_addr);
  if(s->block_this())
  {
    drop(p);
    return;
  }

  int qlimBytes = qlim_ * mean_pktsize_;
  if ((!qib_ && (q_->length() + 1) >= qlim_)
      || (qib_
          && (q_->byteLength() + hdr_cmn::access(p)->size())
              >= qlimBytes)) {
    // if the queue would overflow if we added this packet...
    if (drop_front_) { /* remove from head of queue */
      q_->enque(p);
      Packet *pp = q_->deque();
      drop(pp);
    } else {
      drop(p);
    }
  } else {
    q_->enque(p);
  }
}
```

Lastly we need to set up the default values that are going to be used when we set a link with our queue management algorithm. Since we want it to behave the same way as the *DropTail* queue management algorithm we will simply create the same default values. To do so, we need to add the following lines in the *ns-default.tcl* file that is inside the tcl/lib directory.

```
Queue/IPS set drop_front_ false
Queue/IPS set summarystats_ false
Queue/IPS set queue_in_bytes_ false
Queue/IPS set mean_pktsize_ 500
```

Regarding the way that we use this tool lets first suppose that we want to create a node in NS2 and block traffic from it. In order to do so we have write something like the following:

```
set ns [new Simulator]; # Create the simulator object
set n0 [$ns node];      # Create a node
$n0 block-me;           # Block traffic from that node
```

In order to create a node that drops traffic we need to connect two nodes with a link and use our queue management algorithm in that link. The following code shows how to create a such a link with 1Mb bandwidth and 2ms delay.

```
$ns simplex-link $n1 $n2 1Mb 2ms IPS; # Traffic dropping mechanism will take
    place in node n1
```

With all these in mind we can create complex simulations where for example the dropping mechanism's trust about a node changes at some point and we want to let the packets from that node flow beyond the dropping point. This can be easily done since NS2 provides an event scheduling mechanism. For example in order to start accepting traffic from n0 after 60 seconds we have to use the following piece of code:

```
$ns at 60 "$n0 unblock-me";
```