Journal of Computer Security 13 (2005) 807–837
IOS Press

Mitigating denial of service attacks: A tutorial

Jarmo Mölsä
National Defence College, Department of Technology, P.O. Box 7, FI-00861 Helsinki, Finland
and Communications Laboratory, Helsinki University of Technology, Finland
E-mail: jarmo.molsa@tkk.fi

This tutorial describes what Denial of Service (DoS) attacks are, how they can be carried out in IP networks, and how one can defend against them. Distributed DoS (DDoS) attacks are included here as a subset of DoS attacks. A DoS attack has two phases: a deployment and an attack phase. A DoS program must first be deployed on one or more compromised hosts before an attack is possible. Mitigation of DoS attacks requires thus defense mechanisms for both phases. Completely reliable protection against DoS attacks is, however, not possible. There will always be vulnerable hosts in the Internet, and many attack mechanisms are based on ordinary use of protocols. Defense in depth is thus needed to mitigate the effect of DoS attacks. This paper describes shortly many defense mechanisms proposed in the literature. The goal is not to implement all possible defenses. Instead, one should optimize the trade-off between security costs and acquired benefits in handling the most important risks. Mitigation of DoS attacks is thus closely related to risk management.

Keywords: Network security, denial of service, attack mechanisms, defense mechanisms

1. Introduction

Denial of Service (DoS) attacks have proved to be a serious and permanent threat to users, organizations, and infrastructures of the Internet [26]. The primary goal of these attacks is to prevent access to a particular resource like a web server [8]. A large number of defenses against DoS attacks have been proposed in the literature, but none of them gives reliable protection. There will always be vulnerable hosts in the Internet to be used for DoS purposes. In addition, it is very difficult to reliably recognize and filter only attack traffic without causing any collateral damage to legitimate traffic. This paper describes, how DoS attacks can be carried out and how a victim can mitigate them in ordinary IP networks. Especially wireless ad hoc networks have their additional vulnerabilities, but these kind of wireless networks are not the subject of this paper.

A DoS attack can be carried out either as a flooding or a logic attack [43]. A *flooding DoS attack* is based on brute force. Real-looking but unnecessary data is sent as much as possible to a victim. As a result, network bandwidth is wasted, disk space is filled with unnecessary data (e.g., spam E-mail, junk ftp data, intentional error messages), fixed size data structures inside host software are filled with bogus information, or processing power is spent for unuseful purposes. To amplify the effects,

DoS attacks can be run in a coordinated fashion from several sources at the same time (Distributed DoS, DDoS). A *logic DoS attack* is based on an intelligent exploitation of vulnerabilities in the target. For example, a skillfully constructed fragmented IP datagram may crash a system due to a serious fault in the operating system (OS) software. Another example of a logic attack is to exploit missing authentication requirements by injecting bogus routing information to prevent traffic from reaching the victim's network.

There are two major reasons making DoS attacks attractive for attackers. The first reason is that there are effective automatic tools available for attacking any victim [9], i.e., expertise is not necessarily required. The second reason is that it is usually impossible to locate an attacker without extensive human interaction [12,59] or without new features in most routers of the Internet [11].

This paper gives a short tutorial on DoS attack mechanisms in IP networks and some important defenses proposed in the literature. The emphasis of this paper is on DoS attacks in general, and DDoS attacks are treated as a subset of DoS attacks. DDoS attacks are based on the same mechanisms as basic DoS attacks, but there is one exception during the deployment phase. A DDoS tool needs to be installed on many vulnerable hosts. The spreading mechanisms for DDoS tools are described in a separate section. The installation of DoS software on a single vulnerable host is, however, a common prerequisite for most DoS attacks. Thus defenses described in this paper are applicable to both DoS and DDoS attacks. The set of defenses described in this paper is definitely not exhaustive, but it gives a good overview of the different possibilities in combating DoS attacks.

It is claimed in this paper that a comprehensive set of defenses are needed to get defense in depth against DoS attacks. It is important to have defenses for both the deployment and the attack phase. The earlier the preparation or actual use of a DoS tool is detected, the better the chances are for mitigating an attack. The selection of a cost-effective set of defenses must, however, include many business aspects. The most important assets of an organization must be protected with a finite amount of money. The selection and implementation of different defenses should be guided by a risk management process.

This paper is organized as follows. First the basic terminology is explained. Then the deployment phase of a DDoS attack is described including a short description of the possibilities for increasing the worm propagation rate and some results from studies on real-life worm propagation. In Section 4 this paper describes the DoS attack phase, its underlying mechanisms, and some estimates about real-life DoS activity in the Internet. Section 5 explains the major phases in handling DoS attacks at a victim site. The next section gives an overview of a wide range of defenses useful in the deployment and the attack phase. Risk management and the selection of a cost-effective set of defense mechanisms are shortly discussed in Section 7. The final section concludes this paper.

2. Terminology

Information security has three fundamental objectives: confidentiality, integrity, and availability [2,22]. *Confidentiality* is defined as the property that information is not disclosed to unauthorized entities. *Integrity* is defined as the property that data has not been changed, destroyed, or lost in an unauthorized or accidental manner. *Availability* is defined as the property of a system or a system resource being accessible and usable upon demand by an authorized entity.

A DoS attack aims in degrading availability. *Denial of Service* has been defined as the prevention of authorized access to resources or the delaying of time-critical operations [22]. Examples of these resources are network bandwidth, processing capacity, disk space, memory, and static memory structures [8].

DoS attacks can be classified based on the number of sources included in the attack [26]. In a *basic DoS attack* the attacker uses a single source host to send attack traffic to a victim. In a *DDoS attack* an attacker uses multiple source hosts to send attack traffic to one or more victims simultaneously.

Typically the participants in a DDoS attack form a hierarchical *DDoS network*, where an attacker controls a few *masters* (or handlers), which in turn control a much higher number of *agents* (or daemons or zombies) to carry a real attack against a victim.

DoS attacks may be either destructive or degradative ([28], p. 160). A *destructive* DoS attack prevents the availability of a service completely. For example, an attack can crash a system or fill disk partitions. In these cases human intervention is needed for recovery. In a *degradative* (non-destructive) DoS attack the performance of a service is reduced, such as in a flooding attack overloading a network link or a host CPU. This will typically cause only temporary problems, and a system will recover automatically as soon as an attack terminates. A prolonged wide-bandwidth flooding attack, however, may have unexpected results, such as system crashes.

A DoS attack can be seen to have two different directions. It is an *inward attack* from the victim point of view, but from the attack source point of view it can be classified as an *outward attack*.

The term *intrusion* means unauthorized usage or misuse of a computer system [50]. In the context of this paper an intrusion is thus a successful DoS attack where a victim suffers from a degradation of availability for any service [2]. From an attacker's perspective an attack is successful only, if at least one objective of an attacker is fulfilled.

DoS attacks consist of two major phases [26]. In the *deployment phase* an attacker installs a DoS tool in one or more vulnerable hosts. In the *attack phase* an attacker coordinates a flooding or a logic attack against a victim. Both of these phases make use of deficiencies in the design or implementation of applications, protocols, and the Internet architecture [37].

A *vulnerability* is a flaw in security procedures, software, internal system controls, or implementation of an information system that may affect the integrity, confidentiality, and/or availability of data or services [60]. An *exploit* is a program, script, or

technique that makes it possible to take advantage of a vulnerability in a system [2]. A *threat* is any circumstance or event that could harm a critical asset through unauthorized access, compromise of data integrity, denial or disruption of service, or physical destruction or impairment. A *risk* is the probability that a particular vulnerability is exploited by a particular threat weighted by the impact of that exploitation.

3. Distributed DoS deployment phase

This section is DDoS specific, because it describes how a DDoS network can be built. Originally DDoS attack tools were deployed manually, but now worms are typically used for that. Worms are self-propagating malicious software which often either include directly the DDoS attack capability (e.g., Code Red I [41] and Slapper [3]) or contain the possibility to execute arbitrary code (e.g., Code Red II [41]). Some worms, however, are designed to propagate fast without any malicious payload (e.g., Slammer [40]) or their full functionality is not known (e.g., Nimda [57]).

Viruses can also be used for the deployment phase to build a large DDoS network, but they cannot replicate automatically by themselves. Typically social engineering is required to get a human to start a program containing a virus (e.g., an E-mail attachment). A target for a DDoS attack has typically some time to prepare, because viruses are identified and reverse-engineered as soon as they are found in the wild. Infected hosts can also be disinfected as soon as antivirus updates are available. A worm, on the contrary, propagates fast and can cause a sudden attack. This makes a worm a more serious deployment tool for DoS attacks.

Combining a DDoS tool with an efficient worm propagation mechanism makes installation of DDoS networks fast. Practically any DDoS tool can be wrapped inside a self-propagating worm [26]. Also, worms make it possible to quickly create multiplatform DDoS networks. Several worms can be launched at the same time to install an identical DDoS tool on hosts with different operating systems or applications.

3.1. Modeling worm propagation

Propagation of worms can be modeled well with the *epidemic model* describing the spread of infectious diseases [42,57]. Epidemic model gives the average infection rate in a population where the infected individuals (*infectives*) contact uninfected individuals (*susceptibles*) with the average contact rate of β. According to this model the proportion of infectives i in a population of N at time t is given by the following equation:

$$i(t) = \frac{e^{\beta(t-T)}}{1 + e^{\beta(t-T)}},$$

where T is a constant of integration, and it fixes the time of the incident, i.e., the start of the fast increase. For a $t < T$, i grows exponentially. For a t significantly after T, i goes to 1, which means that all susceptibles are infected.

The value of i grows initially very slowly, because there are few infectives to infect others. A reasonably large amount of infectives (a critical mass) is a prerequisite for the start of the phase of fast increase of i.

3.2. Increasing the worm propagation rate

There are several mechanisms which can make the worm propagation faster. All these methods increase the value of β in the epidemic model.

First, an efficient *scanning strategy* increases the probability of finding new vulnerable hosts, and this decreases the time spent on trying to infect non-vulnerable, non-existing, or already infected hosts [57]. Simple worms use the *random scanning* strategy which is based on scanning the whole address space in a random order [41]. A worm using the *localized scanning* strategy tries to infect hosts with the same address prefix with a higher probability [41]. This strategy infects quickly networks with many vulnerable hosts. The problem with these strategies is that the infection rate is slow at the very beginning of the initial phase of the epidemic model, and the total time to infect most of the vulnerable hosts will be long. Instead, a good scanning strategy should maximize the initial infection rate. The *hit-list scanning* strategy is based on a list of potentially vulnerable machines preferably with high-speed network connections. An alternative to hit-list scanning is the *topological scanning* strategy which is based on using information from an infected host to select new victims. Scanning same addresses several times and trying to infect same hosts several times can be prevented with the *permutation scanning* strategy in which infected hosts share a joint pseudo-random permutation of the IP address space. Every host works through a different subsequence in this permutation, and if a host finds an already infected host, it selects a new random starting point. The permutation can be implemented, for example, by encrypting target addresses with a 32 bit block cipher and a preselected key. Different scanning strategies can be combined.

Second, *the scanning rate* can be increased. The objective is to be able to scan at a rate proportional to a host's access rate [40]. A *latency-limited worm* cannot use all the available bandwidth, because the transmission rate is restricted by the delays associated with the connection setup (the TCP three-way handshake), the TCP congestion control, or the inefficiency of an implementation. A *bandwidth-limited worm*, on the other hand, scans and infects hosts as fast as a network is able to transmit packets. Selecting victims with higher access rates will make a worm more virulent. Home users with wide-bandwidth Digital Subscriber Lines (xDSL) are seldom security-conscious which makes them easy to exploit [43]. Universities are appealing due to wide-bandwidth Internet connections and open usage policies [21]. By using threads carefully a worm can send even congestion controlled TCP packets with the full access rate [57]. By using UDP a worm does not experience any delays due to

connection setup or end-to-end congestion control. With UDP it is even possible to infect a vulnerable host with a single packet [55].

Third, *multi-vector propagation* utilizes several propagation mechanisms simultaneously [57]. For example, a worm can propagate by exploiting a vulnerability to infect hosts directly, by downloading itself through backdoors installed by other worms, by attaching itself in E-mails, by copying itself to open network disk shares, and by modifying Web pages to infect hosts browsing these pages.

Fourth, exploitation of vulnerabilities in *Peer-to-Peer (P2P) applications* can be easy, because all participants are using the same application-level protocol and there are at most few different implementations [57]. Insertion of infected files or exploitation of vulnerabilities in a P2P application-level protocol can compromise a large number of hosts participating in the same P2P network. A popular P2P system can have millions of users logged on at any time [32].

The possibilities to increase the worm propagation rate are shown in Fig. 1.

All in all, it is possible to infect a large amount of hosts in a very short time, and small susceptible populations are vulnerable also. It has been estimated that a flash worm can infect three million hosts out of 12.6 million hosts in 30 seconds by combining complete hit-lists with fast-rate scanning [57]. It has also been estimated that a relatively small population of 20 000 hosts randomly located in the Internet can be infected in less than an hour [40]. Exploits for less popular software can thus be utilized effectively for worm propagation.

3.3. Worm propagation in real-life

The real-life propagation of the random scanning Code Red I v2 worm was analyzed in [41]. This worm exploited a vulnerability in the Microsoft IIS web server. On July 19, 2001, this latency-limited worm managed to infect more than 359 000 hosts within 24 hours. At maximum the worm generated a total of 510 000 TCP SYN probes per hour [57]. The measured infection rate corresponded with the epidemic model reasonably well with parameters $N = 360\,000$ hosts, $\beta = 1.6$ infections per hour and $T = 16$ hours. A single infective was thus able to compromise a susceptible every 37 minutes.

The real-life propagation of the random scanning Slammer worm was analyzed in [40]. This worm exploited a vulnerability in the Microsoft SQL server. On January 25, 2003, this bandwidth-limited worm infected at least 75 000 hosts in 10 minutes, which was more than 90% of the susceptibles. At maximum the worm generated in total over 55 000 000 UDP scans per second. A single infective was able to compromise a susceptible every 8.5 seconds.

In addition to infection rate, the epidemic model also describes well the total probing/scanning rate of worms, and this has been verified for the Code Red I v2 and the Slammer worms.

The Slapper worm was analyzed in [3]. This worm exploited a vulnerability in the OpenSSL software used by many Linux distributions and Apache Web servers. On September 13, 2002, it compromised approximately 6000–16 000 hosts.

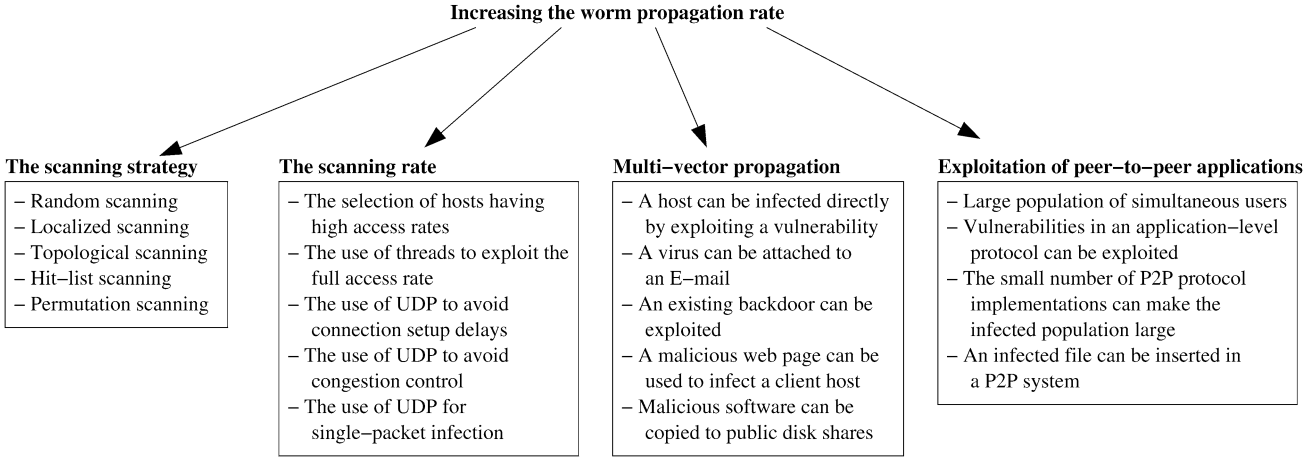

Fig. 1. Factors having an effect on the worm propagation rate.

The Witty worm [55] was bandwidth-limited and used hit-list scanning. It exploited a vulnerability in the ISS RealSecure firewall. This worm was launched only one day after the disclosure of the underlying vulnerability. On March 19, 2004, it infected approximately 9000 hosts out of a population of 12 000 hosts within 45 minutes. A single UDP packet was used to infect a vulnerable host.

4. Attack phase

Once DoS software has been deployed, an attacker is able to proceed to the final attack phase. An actual attack will consist of a flooding or a logic attack against a single victim.

The different attack mechanisms described in this section are shown in the Fig. 2.

4.1. Coordination of DDoS agents

In case of a DDoS attack an attacker must first coordinate all DDoS agents to attack in unison for effectiveness reasons. This coordination requires attack commands to be transmitted to every agent through a *control channel*. There are several choices for transmitting this control channel information, usually in an encrypted form. Basic TCP or UDP are used by the simplest DDoS tools. Many ordinary protocols provide a way to tunnel commands, for example inside the ICMP or the Domain Name Service (DNS) payload data. Especially packets containing a reply (e.g., an ICMP *Echo Reply* or a DNS answer) have a higher probability of passing through firewalls. Public Internet Relay Chat (IRC) protocols and networks are used recently in an increased fashion [26]. The Slapper worm found in September 2002 contains the ability to execute DDoS attacks, and the coordination traffic is carried over a specific peer-to-peer (P2P) protocol, where all commands and responses are transmitted through a random chain of agents to make the DDoS network more robust against tracing [3].

Coordination of a DDoS attack and a related control channel is not necessarily required, because the information about a future attack can be hard-coded in a DDoS tool. This makes it possible, however, to reverse-engineer a detected DDoS tool to find the time and victim of a DDoS attack. The sooner this information is available, the more can be done to prevent or at least prepare against this kind of an attack.

4.2. IP spoofing

A basic mechanism in all DoS attacks to hide the location of an attacker is *IP spoofing* which means sending packets with a false source IP address. A certain kind of a value in the source IP address field is also a prerequisite for some DoS attacks. Setting the victim's address in the source field makes it look like the packet

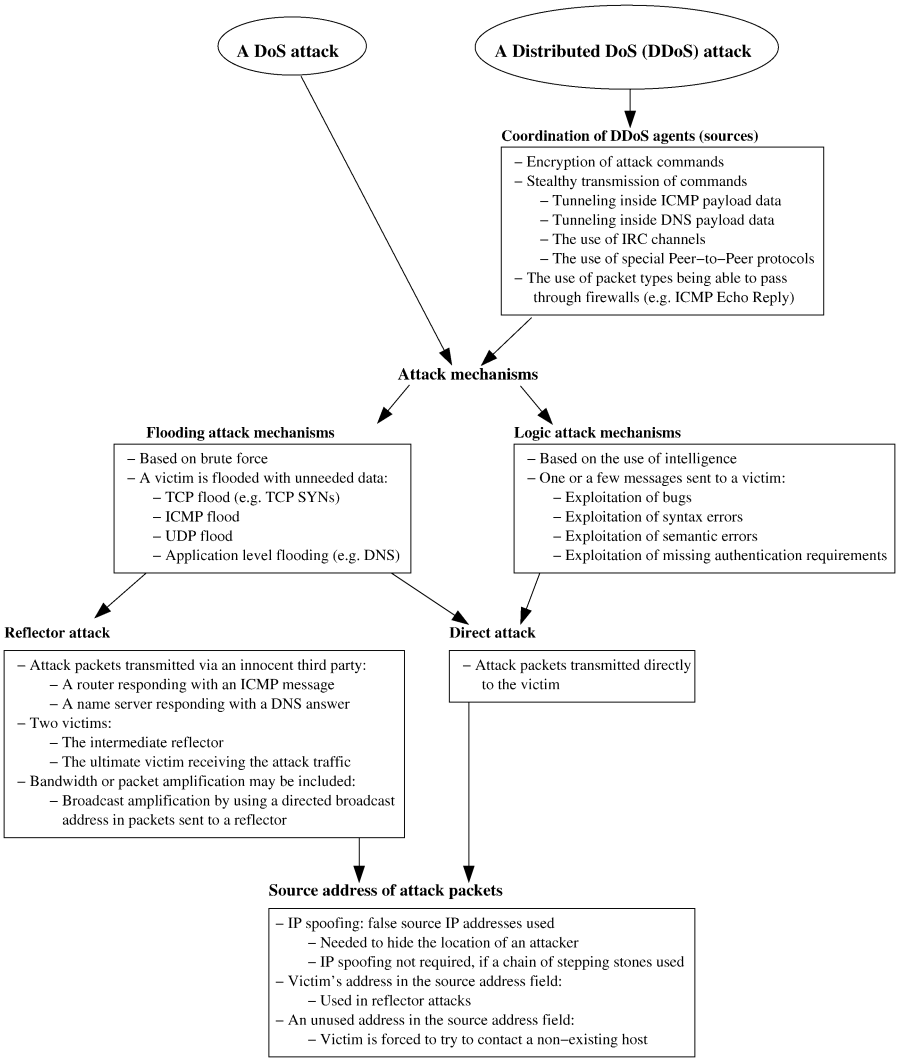

Fig. 2. The major attack mechanisms used by DoS or DDoS attacks.

was originally sent from the victim. Setting an unused IP address in the source field forces the victim to try to contact with a non-existing host.

It is possible to carry out DoS attacks without IP spoofing, if an attacker has compromised enough hosts, or if a chain of compromised hosts is used. Tracing an attacker through a chain can be made difficult or impossible by selecting the compromised hosts of a chain from sites with poor security practices or from countries with suitable legislation. In this kind of a case IP spoofing is not necessarily required for

protecting an attacker.

4.3. Flooding attack mechanisms

Flooding DoS attacks are generally divided into direct and reflector attacks [11]. In *direct attacks* spoofed packets are sent directly to the victim. In *reflector attacks* packets with the victim's address in the source IP address field are sent to an innocent third party, which in turn will send the reply to the victim. Examples of innocent third parties are web servers, DNS servers, and routers. Reflector attacks have thus at least two victims at the same time [18].

Basically any ordinary protocol behavior can be utilized as the underlying mechanism for flooding attacks. Any protocol layer is suitable for attack purposes.

Direct attacks use typically only few mechanisms, namely TCP SYN flooding, ICMP `Echo` flooding, or sometimes UDP data flooding [12]. In TCP SYN flooding the victim is sent SYN packets with an address of a non-existing host in the source IP address field, which will result in lots of half-open connections that fill static data structures and prevent legitimate connections. These half-open connections will timeout by default in 75 seconds [23]. In ICMP `Echo` flooding the victim is forced to handle a large amount of ping-packets. In UDP data flooding one possible objective is to connect `chargen`- and `echo`-ports between two victims. These attacks do not consume resources permanently, so at least in theory, the victim should be able to continue serving legitimate users normally after the attack is over. Some flooding attacks may, however, have longstanding effects. For example, IP fragment flooding may consume all available memory for storing partial IP datagrams, after which hosts may crash due to unavailability of free memory.

Reflector attacks utilize any protocol behavior, where an attack packet triggers a response packet to be sent to the ultimate victim [49]. Reflector attacks can also include the use of a technique called *bandwidth or packet amplification*. The innocent third party will either reply with a longer packet or with several packets to a single attack packet, respectively. A form of packet amplification is the broadcast amplification where an attack packet is sent to a subnet directed broadcast address. All hosts receiving the attack packet will send their own response to the ultimate victim. A well-known example of this is the smurf attack [46], where a single ICMP `Echo` is amplified into several ICMP `Echo Reply` packets.

Flooding attacks against routers can be effective, because routers are usually optimized for forwarding traffic instead of handling data sent directly to them [9]. Flooding attacks against DNS can cause widespread Internet slowdowns or effective outages.

A wide-bandwidth stream of packets is not necessarily required for a flooding attack. Bandwidth less than a typical analog modem speed may be enough in exploiting deficiencies in the implementation of data structures [16]. These kind of *algorithmic complexity attacks* can, for example, degenerate binary trees and hash tables into linked lists. Due to the requirement of intelligence in selecting the attack traffic, these algorithmic complexity attacks might as well be classified as logic attacks described in the next subsection.

4.4. Logic attack mechanisms

The objective of logic DoS attacks is to build a small number of specific packets exploiting vulnerabilities which cause the victim to do abnormal things. The packets are normally sent directly to a victim, because special knowledge about a vulnerability is required. There is a wide variety of logic attacks. Typically an attack is based on more than one of the following issues at the same time:

- Exploitation of bugs: All software contains bugs which can e.g., cause a host to crash due to errors in dynamic memory structure handling, like in the Teardrop-attack based on overlapping IP fragments ([46] pp. 54–55).
- Exploitation of syntax errors: Implementations are not always able to handle syntactically incorrect data, like in the Internet Group Management Protocol (IGMP) attack based on malformed headers [53].
- Exploitation of semantic errors: Implementations may process normally all syntactically correct messages, even if these messages are semantically incorrect. For example, in DNS cache poisoning a bogus mapping (a DNS answer) may be appended to an innocent looking query message [9].
- Exploitation of missing authentication requirements: Lack of authentication makes it possible to enter false information into many protocols (e.g., dynamic routing protocols) and services (e.g., DNS) [47].

Implementations of protocols may include non-standard or missing features which can be exploited. For example, some implementations of the TCP state machine are non-standard and include extraneous state transitions, or not all states have well-defined timeouts. In these kind of hosts it is possible to force the state machine to enter a state which cannot be exited or has a very long timeout. These vulnerabilities in some implementations can be exploited by TCP SYN-FIN packet streams or by replying to a TCP SYN packet with a TCP SYN packet [23].

Logic attacks against routers and security devices can affect large parts of the Internet infrastructure and enable other kind of attacks, when part of the defenses are no more operational. DoS attacks against the Internet infrastructure are becoming more common. The routing infrastructure is considered to be an excellent target for DoS attacks, because an attacker is able to cause severe network outages without a significant effort by simply injecting false routing information [47]. DNS is another infrastructure service vulnerable to DoS attacks, for example when a domain is hijacked [9].

4.5. DoS attacks in real-life

Real DoS incidents on the Internet between years 1989 and 1995 were investigated in [28]. The three most typical effects were the following: 51% of these incidents filled a disk, 33% of the incidents degraded network service, and 26% of the incidents deleted some critical files. A single incident was able to cause several types of damages at the same time (the sum of percentages is more than 100%).

The first reported large-scale DDoS attack occurred in August, 1999, against a university [21]. This attack shut down the victim's network for more than two days.

In February 7, 2000, several high-profile Web sites were attacked, which caused them to go offline for several hours [21]. In some cases these DDoS attacks were able to produce about 1 Gbit/s of attack traffic against a single victim. All or almost all DDoS agents were Unix or Linux hosts, some of which resided in university networks.

The *backscatter analysis* was used to assess the number, duration, and focus of DoS attacks in the Internet [43]. Backscatter is called the unsolicited response traffic which the victim sends in response to direct attack packets with spoofed IP source address. The results indicate more than 12 000 attacks against more than 5000 distinct victims during the 3-week period examined in February, 2001. More than 50% of the recognized attack traffic were TCP RST packets, which are a result of either TCP SYN or other unexpected TCP packets. Also ICMP `Host Unreachable` packets were mostly result of a TCP packet. The majority of identified direct DoS attacks are thus TCP-based, probably TCP SYN floods. The median attack duration was 10 minutes.

Fragmentation in real networks was studied in [56]. Bugs in the fragment handling software are exploited in many logic DoS attacks, and the results of this study still indicate the presence of these kind of DoS attacks in the Internet.

5. Handling DoS attacks at a victim site

In general, protection against DoS attacks consists of preparation, detection, and reaction phases [27]. The *preparation phase* includes, for example, creation of a security policy, installation of required security devices, separation of the critical services from each other, overprovisioning of capacity, monitoring ongoing operations to learn to know what is normal behavior, training of analysis capabilities, creation of an incident response plan, and making a cooperation plan with an Internet Service Provider (ISP). It should be noticed that incident response should not depend on the proper operation of a system under attack [31]. For example, a local compromised host can give unreliable information about an incident, it may be impossible to reach the real administrators of a remote compromised host, or it can be difficult to contact participants through an overloaded network.

The *detection phase* should be automatic. To be able to react as fast as possible, an *early warning system* is required, which means detecting DoS attacks as early as possible [36]. The later an attack is detected the less administrators have time to react before clear damages are caused to legitimate traffic, e.g., in the form of decreased availability of services.

The *reaction phase* consists of two subphases, namely characterization and mitigation. In the *characterization phase* the victim must verify, if an attack is really

going on, and the victim must also analyze the attack to be able to find distinguishing characteristics of the attack traffic. A good understanding of the nature of the attack is required for the *mitigation phase*, in which the victim installs the required defenses, such as filters to block attack traffic.

5.1. Detection of DoS attacks

Intrusion Detection Systems (IDS) are tools for detecting intrusive network or host activity, and announcing alerts [45]. These systems can be divided in two major classes. *Network Intrusion Detection Systems* (NIDS) are passive nodes which have access to all traffic in a network link. *Host Intrusion Detection Systems* (HIDS) are applications which analyze log files and other security related information and try to detect intrusive use of a single host. NIDSes and HIDSes have non-overlapping advantages and disadvantages, so an important site needs to employ a combination of them.

An IDS consists of three major components [2]. *Sensors* are responsible for collecting data possibly from several locations (e.g., networks, files). *Analyzers* receive data from sensors or other analyzers. If an intrusion is recognized, an alert is created. *User interface* is the main tool for administrators for checking details of an alert.

There are two distinct analysis methods to decide, whether an intrusion has been found or not. Signature-based *misuse detection* tries to locate known patterns from the incoming sensor data, much like the existing antivirus software does. The major problem with misuse detection is the requirement for exact signatures of attacks, which makes these kind of systems reactive and place strict requirements on the speed of signature updating [44]. This means inability to detect new or even slightly modified attacks. *Anomaly detection* is based on observing significant deviations from typical or expected behavior of systems or users [36]. The major problem with anomaly detection is the difficulty in defining, what is typical or expected behavior and what is not [2]. Anomaly detection systems can detect some new or modified attacks.

5.2. Effectiveness of DoS attack detection

IDSes have proved to be necessary tools for detecting attacks [46]. An IDS can provide log files and traces of network traffic which can be used to get further information about the involved hosts and the amount of damages. Later this information can be used as a proof of an attack in lawsuits etc. IDSes are used in an increasing fashion to show the presence of attacks against corporate and even home networks.

Detection of DoS attacks is not simple, because these attacks exploit features of ordinary protocol behavior. By choosing an attack method suitably an attacker has the possibility of escaping the detection by an IDS. In general there are three possibilities for this.

First, an IDS may not be able to collect all desired information. An IDS may simply be not effective enough to handle all available data, or implementation bugs may make it inclined to crashing. In [44] several signature-based NIDSes were compared. Many systems dropped a relevant part of network frames or were not robust enough crashing from time to time. An IDS can be overloaded intentionally by an attacker, so that at least part of the attack traffic is dropped by the IDS. This makes it possible to evade detection [48]. An IDS is also susceptible to a crash attack, in which the attacker knocks down whole or part of an IDS by utilizing some vulnerability [48]. Flooding and logic DoS attacks can thus be used against an IDS to prevent it from collecting information.

The second reason for an IDS being not able to detect all intrusions is the possibility for an *evasion* or *insertion* attack against an NIDS. These attacks exploit ambiguities in the payload of packets [25]. For example, an NIDS may reassemble overlapping IP fragments in a different fashion than an end-host. This can prevent an NIDS from seeing complete signatures broken down in multiple overlapping fragments [48]. In an insertion attack an NIDS accepts a packet that an end-system rejects or does not receive. In an evasion attack an NIDS rejects a packet that an end-system accepts. Both insertion and evasion attacks break signatures and thus prevent an NIDS from recognizing an attack. Insertion and evasion attacks can be implemented e.g., by using a low TTL-value not reaching the end-system, by using a packet longer than the MTU of the end-system network and setting the `Don't Fragment` flag in the IP header, by using source-routed packets discarded at the end-system, by exploiting different IP reassembly timeouts, by sending overlapping IP fragments, by sending overlapping TCP segments, or by using special combinations of TCP flags not accepted by every TCP/IP stack implementation [50].

The third reason for an IDS being not able to detect all intrusions is the inability to recognize intrusions correctly even from a complete and correct sensor data. This can happen if attack traffic resembles legitimate traffic too much. The frequency of false positives (false alerts) is important, because they need to be checked by humans. Too many false positives a day make an IDS completely useless. The number of false positives can be reduced at the expense of the amount of true positives (detected real intrusions). If an IDS is tuned to create few enough of false positives, this IDS also gives less true positives, i.e., less true attacks are detected. The relation between false positives and true positives can be represented as a *Receiver Operating Characteristics (ROC) curve*. Some ROC curves of real IDSes can be seen in [34] and [17]. An ROC curve for an IDS clearly indicates, what kind of effect reducing the number of false positives has on the number of detected true positives. In one study the average detection rate for known DoS attacks was about 80%, but for new or slightly modified DoS attacks the average detection rate was only about 20%, when the maximum amount of false positives was set to approximately 10 a day [34].

Examples of ROC curves are shown in Fig. 3, where the continuous line shows an ROC curve for an IDS based on anomaly detection, and the dotted line shows an ROC curve for an IDS based on misuse detection. An IDS based on misuse detection

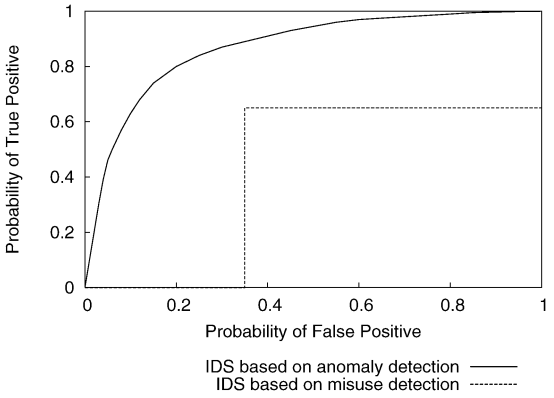

Fig. 3. Sample ROC curves for an IDS using anomaly detection and an IDS using misuse detection.

operates practically in a single point at (0.35, 0.65) in Fig. 3, because this kind of an IDS either detects an attack or not. An IDS based on anomaly detection, on the other hand, assigns different warning values for suspected attacks. Weak signs of an attack are indicated with lower warning values and clear signs of an attack are indicated with higher warning values. One point of the ROC curve is calculated by selecting one of the possible warning values as a criterion value. The probability of a true positive is the amount of those real attacks having a warning value greater or equal to this criterion value, divided by the total amount of real attacks. The probability of a false positive is the amount of those legitimate sessions/connections having a warning value greater or equal to this criterion value, divided by the total amount of legitimate sessions/connections. By looping this criterion value over all warning values, it is possible to create a complete ROC curve ([61], pp. 26–34). The shape of an ROC curve depends on the test material.

5.3. Reaction against detected DoS attacks

As was shown in the previous subsection, detection of DoS attacks is not a simple task. An experienced attacker can hide DoS activity. This has implications on the reaction phase. Automatic reaction mechanisms are fast, but the problem with false positives must be tackled somehow. Typically human intervention is required at some moment of time.

A prerequisite for the mitigation of DoS attacks is a detailed knowledge of the details of an ongoing attack (the characterization subphase). If the exact signature of attack traffic is not known, the mitigation of flooding DoS attacks can easily cause damage for legitimate users.

A widely used way to react against DoS attacks has been a labor-intensive manual procedure by network administrators, which means manual input debugging to locate routers on the path of the attack traffic step by step towards the attack source, and

manual installation of packet filtering or rate-limiting rules in these routers handling attack traffic [58].

An automatic mechanism is needed for a quick early reaction. The implementation of a reaction mechanism can reside either in an end-host or a network security device. When comparing the two implementation locations, network security devices are better places for reacting against inward flooding and many logic DoS attacks, because the attack must be mitigated as near the actual source as possible. Host implementations, however, have an advantage in reacting better to outward attacks and the DoS tool deployment phase.

Reaction mechanisms concentrate on mitigating the effects of a DoS attack. A reverse attack, where a victim starts fighting back, is typically not feasible due to IP spoofing. A self-defense or a revenge would only hit another innocent site. A known self-defense mechanism can even be used as a logic reflector DoS attack against a final target.

6. Defending against DoS attacks

In this paper it is claimed that no single defense is enough against a DoS attack. A comprehensive set of defenses has to be utilized to get *defense in depth* ([2], pp. 96–97). If one layer of defense fails, the other defense layers still have the possibility to detect and mitigate an attack. A successful intrusion requires all defense layers to fail. Defense in depth is a widely used term also in human safety [51].

The Internet is as secure against DoS attacks as its weakest hosts. As there will always be exploitable hosts accessible from the Internet, DoS attacks can be launched even against a site with a comprehensive set of defenses. In this respect all hosts in the Internet are dependent on the protection of other hosts. Detection of compromised hosts (deployment phase) is thus as important as detection of DoS attacks (attack phase).

The earlier a DoS attack can be detected and mitigated, the better. Defenses for the deployment phase are important, because they can be used to prevent or detect installation of DoS tools.

This section gives an overview of some available defenses for both the deployment and the attack phase. The list of defenses is definitely not exhaustive, but gives an understanding of the variety of defenses available. The order in which defense mechanisms have been listed is not relevant.

Figure 4 gives an overview of the defenses described in this section. This figure is organized as a simple tree. Defenses against DoS attacks can be selected by traversing this tree from the root towards the leaves. The root of this tree contains necessary defenses required by any network element, like an end-host, a router, or a name server. If basic defenses are not enough and DoS attacks need to be mitigated, one typically needs mechanisms to detect intrusions more effectively. Defense mechanisms against both the deployment and the attack phase are shown in their own boxes in the tree.

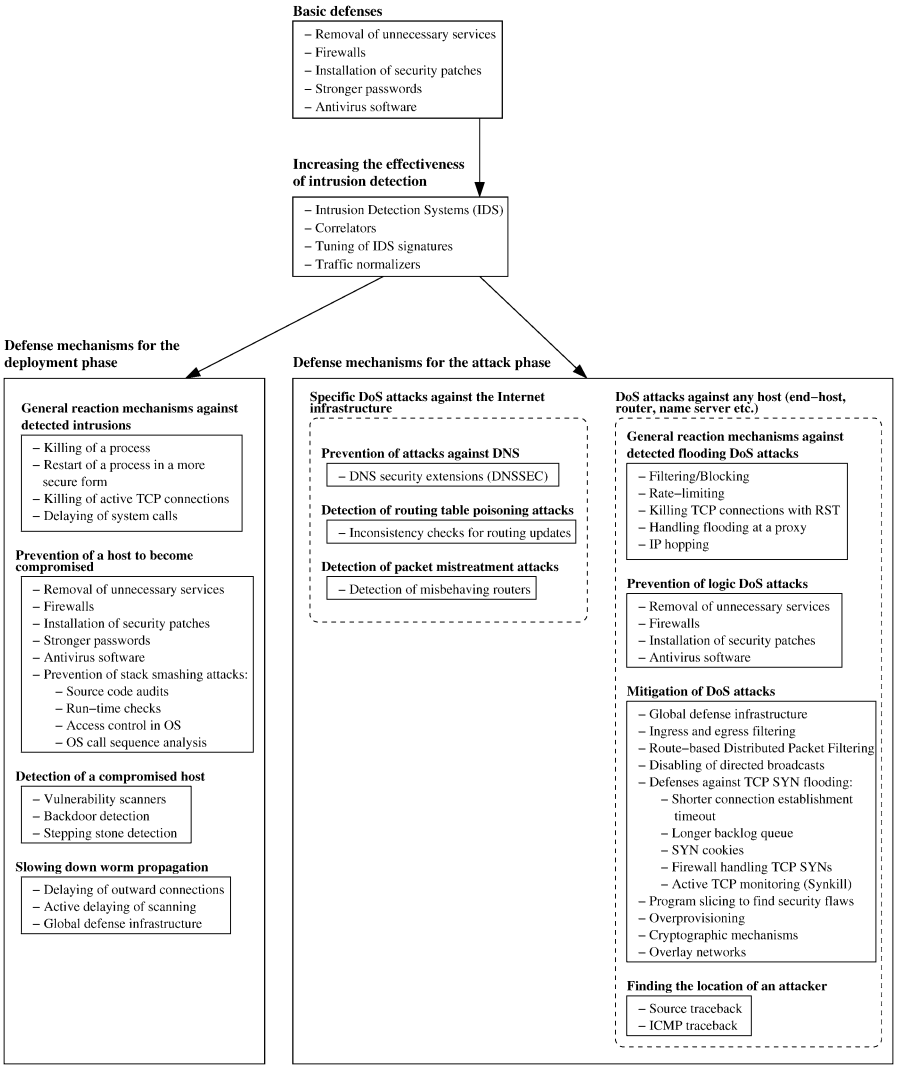

Fig. 4. An overview of different defense mechanisms available to mitigate DoS attacks.

6.1. Basic defenses

Defending against DoS attacks includes some basic issues. These issues should be taken care of by any organization or individual having hosts connected to the Internet. All defense mechanisms listed in this subsection are effective also in preventing or making it more difficult to exploit logic DoS attacks.

All unnecessary services should be removed. The less there are applications and open ports in hosts, the less there are vulnerabilities to be exploited by an attacker. Default installations of operating systems often include many applications not needed by a user. Especially many home-users do not even know, what services are running on their systems. A vulnerability scanner can be used to detect what network services (open ports) are available in a network.

A firewall (or a router with similar abilities) should be used to control access to a network. Even if there are many services available from local hosts, not all of these services need to be accessible from the public Internet.

All relevant security patches should be installed timely. The DDoS tool deployment phase and many logic DoS attacks are based on exploiting vulnerabilities in host software. Removing known security holes prevents re-exploitation of vulnerabilities for example with publicly available scripts. In practice, this important defense is often neglected which makes it possible for available exploits to have lifetimes up to several years [26].

Attackers should not be able to get unauthorized access to hosts, e.g., by exploiting weak passwords. A minimum requirement is to use passwords which are difficult to guess with or without existing password cracking tools.

The antivirus software should be using the most recent virus definition database. This helps detecting known worms and viruses. Antivirus software can thus be considered as an IDS.

6.2. Increasing the effectiveness of intrusion detection

Mitigation of DoS attacks is not possible, if these attacks are not detected. A combination of NIDSes and HIDSes are typically needed in the most important networks and hosts, respectively.

To make intrusion detection more effective, *correlators* can be used to prioritize alerts, group alerts related to different phases of an attack, and group alerts from several different IDS sensors or analyzers [24]. The objective is to combine the benefits of different kind of IDSes and reduce the amount of human interaction.

An IDS needs to be customized to a given environment to reduce the amount of false positives [44]. Instead of technical effectiveness one should increase the cost/benefit trade-off, which means focusing the limited computer and human resources on the most damaging intrusions [33]. This is done by tuning the signatures or training the anomaly detection to detect those attacks, for which the environment is most vulnerable, and to prevent alerting on those attacks, for which environment is not vulnerable. An existing signature should be modified, if it creates too many unnecessary alerts.

Attacks against an IDS should be made as difficult as possible. An NIDS should have defenses against insertion and evasion attacks. If an NIDS itself cannot provide a decent protection against these attacks, a traffic normalizer [25] is one possible tool for this purpose. A normalizer eliminates some ambiguities in the traffic stream, for example by reassembling IP fragments.

6.3. Defense mechanisms for the deployment phase

The goal of the deployment phase is to compromise a host by installing malicious software (a DoS attack tool) on it. A host can be any node connected to the Internet, like an end-host or a router.

Some defense mechanisms described in this subsection only detect the deployment phase, but many defense mechanisms also include an automatic reaction mechanism. Basically it is possible to combine any detection mechanism with one or more reaction mechanisms. The following general reaction mechanisms have been proposed in the literature:

- killing of a process [15],
- restarting an application in a more secure form (e.g., with more effective and more expensive checks against buffer overflow attacks) [15],
- killing of active network connections (e.g., with a TCP RST) [48], and
- delaying of network connections to new destinations [62].

Worm propagation and DoS tool deployment are typically based on so called *stack smashing attacks* [14], where an attacker can get unauthorized access (like a root shell) to a victim host. These attacks are based on overwriting the contents of a stack. Buffer overflow is the most usual software vulnerability used by these attacks. During the time period 1–11/2002 there were 31 CERT Advisories, from which 21 referred to buffer overflow attacks [4]. Another possibility to carry out a stack smashing attack is to exploit a format string vulnerability [5].

Stack smashing attacks can be prevented or made more difficult by paying attention to *software security* which can be enhanced by three different mechanisms [13]:

- *Software auditing* can be used to search vulnerabilities from source code automatically or manually before these vulnerabilities are found and exploited by attackers.
- *Vulnerability mitigation* is based on compile-time installation of special checks that detect certain types of buffer overflow attacks and protect the system at run-time.
- *Behavior management* is based on run-time features in an operating system to limit potential damage or block specific behavior known to be dangerous (access control).

Stack smashing attacks can also be prevented or made more difficult by looking at short sequences of operating system calls [20]. For example, the exploit of a buffer overflow vulnerability in a `sendmail`-program causes it to issue abnormal sequences of system calls, when a `sendmail`-process starts to execute a root shell.

If detection of stack smashing attacks is based on anomaly detection, an attacker may use a service in such an exceptional way that causes the reaction mechanism to halt the process after the detection of a false positive. In this case it should be noted that the misuse of a defense mechanism can result in another kind of a DoS condition,

when an intact process is halted. The length of this kind of a DoS condition may vary depending on whether a process is automatically restarted or if human intervention is required. A service may also be restarted automatically in an enhanced mode which includes additional checks against stack smashing attacks. This can make a service run slower. Naturally it is more important to prevent a host to become compromised.

Compromised hosts can be detected with vulnerability scanners and with an intelligent analysis of network traffic. Malicious software often installs secret backdoors into an infected system. Backdoors are a basic method to provide a hidden access for an attacker into a compromised host. To make it more difficult to locate an attacker, a chain of compromised hosts (or stepping stones) can be used to hide the real source of interactive attack commands. It is possible to uniquely recognize flows containing interactive traffic, because every interactive flow has distinctive packet size and timing characteristics. Encryption does not affect the detection, because the actual content of the flows is not used as the discriminating feature. A backdoor can be detected by recognizing interactive traffic entering an unusual port [64]. A stepping stone can be detected by recognizing two interactive flows with identical characteristics, one entering a host and the other leaving the same host [65].

Worm propagation can be restricted by limiting (delaying) the rate of connections to new destinations [62]. An infected host will try to connect to as many different hosts as fast as possible, but an uninfected host typically makes connections to locally correlated destinations at a lower rate. The slower a worm propagates, the easier it is to prevent the worm from infecting most of the population, because there is more time to react, for example by installing filtering rules in routers.

If a (global) defense infrastructure is available, the amount of infected hosts can be reasonably restricted by installing filtering rules in the most important routers of the Internet. In [42] it was studied how the reaction time affects the size of the infected population in case of the Code Red I v2 worm. To keep the ratio of susceptibles infected within 24 hours below 10%, simple address blacklisting filters (filtering based on IP addresses of the infected hosts) should be installed within 20 minutes, but more generic content filtering (filtering based on the signature of the worm) allows almost three hours for installation. In [42] it was also estimated that in case of the Code Red I v2 worm the infected population could have been restricted to less than 20% of the population really infected. This would have required the installation of worm signatures within 2 hours in the 30 most important Autonomous Systems of the Internet.

An example of a more active defense mechanism for slowing down the worm propagation rate is the LaBrea Tarpit ([46], pp. 173–178). When an attacker scans an unused IP address, a router in the destination network will send an Address Resolution Protocol (ARP) request for the unused destination IP address. If the response for this ARP request is not found within a while, a specific server (the LaBrea host) will respond to it with its own hardware address. The LaBrea host will thus receive traffic sent to an unused destination IP address. The scanning application (or one thread of the attack application) may be delayed for quite a long time (possibly indefinitely) by replying appropriately to incoming TCP packets.

6.4. Defense mechanisms for the attack phase

This subsection lists defense mechanisms presented in the literature for the attack phase. Many of these defense mechanisms are suitable against both flooding and logic DoS attacks, but not all. For instance, some defense mechanisms can not prevent logic DoS attacks, because even a single malicious IP packet is able to cause a DoS condition, such as a system crash.

The following general reaction mechanisms have been proposed in the literature for mitigating flooding DoS attacks [11]:

- Blocking: all packets matching a signature are discarded at an upstream router.
- Rate-limiting: a fraction of packets matching a signature is discarded at an upstream router. A support for Quality-of-Service (QoS) features should be provided by the involved routers. Only incoming packets are rate-limited, and outgoing packets can leave the network freely without any additional penalties. Rate-limiting cannot discard too many packets, because legitimate flows matching an attack signature must survive the one-way packet-loss. This limits the effectiveness of rate-limiting on mitigating wide-bandwidth flooding DoS attacks [38].
- Connection tear-down: malicious TCP connections are torn down with an RST message.
- Flood processing in another place: a DoS flood can be handled in a place with better abilities. For example, a router can take the responsibility of handling (proxying) certain resource intensive tasks. This saves resources in a victim end-host.
- IP hopping: the IP address of a victim is changed in the DNS. In the Code Red I v2 worm the IP address of the victim was hard-coded, which made it easy to prevent the attack by changing the victim's IP address in the DNS.

Blocking and rate-limiting of DoS traffic at upstream routers requires a mechanism for distributing the attack description. The proposed mechanisms for an IDS to distribute attack identification information are the Pushback-messages [19] and the Intrusion Detection and Isolation Protocol [58]. These protocols must authenticate every message, because otherwise an attacker can exploit this mechanism by sending spoofed messages, which cause routers to block or rate-limit legitimate traffic and cause DoS.

The term *Internet firewall* [11] has been used to denote a global defense infrastructure, where many routers in the Internet infrastructure detect and filter attack traffic in a coordinated way. At the moment it is mostly the responsibility of the owner of an end-host to protect it from malicious attacks by using conventional firewalls, antivirus software etc. The Internet infrastructure is more optimized for efficient transmission of IP packets than for implementing defense mechanisms against malicious attacks. An Internet firewall means giving the Internet infrastructure a more active

role in mitigating the effect of malicious network behavior on end-users. A wide-scale Internet firewall will, however, have problems handling false positives. It will be difficult to restore the state of a legitimate stream, if it has been accidentally classified as attack traffic. A defense infrastructure consists of systems from several different organizations, and this may slow down the cooperation during error cases. An Internet firewall, however, has found to be a necessary defense mechanism against fast-spreading worms exploiting a newly found vulnerability [55].

IP spoofing can be restricted by using ingress and egress filtering [18] in a border router of a network, such as a network of an ISP or its customer. An ingress filtering router in an ISP network will check that packets coming from a customer network have valid source IP addresses with the associated prefix of that customer. Egress filtering does the same check for packets going in the opposite direction, which prevents an ISP border router from forwarding packets that have a source IP address belonging to the same customer as the destination IP address. It should be noted, however, that ingress and egress filtering do not prevent DoS attacks, because a well-chosen chain of stepping stones makes it unnecessary to use IP spoofing for hiding the attacker. Also, ingress and egress filtering do not prevent sending packets with addresses of non-existent hosts (and with the correct address prefix), because only the address prefix is checked. It is not feasible for an ISP to keep track of all IP addresses really used by the customers.

In route-based Distributed Packet Filtering (DPF) a router will enforce that a packet is received through the same interface as it would be sent back to the original sender. A packet is discarded, if these two interfaces do not match. DPF is not able to consider asymmetric routing or recent route changes [11].

Flooding DoS attacks based on broadcast amplification can be prevented by disabling directed broadcasts in routers [54], because this feature is normally not utilized, except in some implementations of Mobile IP.

TCP SYN flooding is a widely used flooding DoS attack mechanism. Most available DoS tools support this attack type and studies also indicate that most DoS attacks are TCP-based [43]. The effect of TCP SYN flooding attacks can be mitigated by applying the following defenses [52]:

- improve end-system configurations (reduction of the timeout period for half-open connections, increase in the backlog queue size),
- improve connection establishment to prevent storing half-open connections (storing the connection status in the initial sequence number as a SYN cookie),
- move the burden of handling half-open connections to a firewall, and
- monitor actively existing TCP connections (Synkill, sources classified as evil are prevented from making additional connections).

Security flaws in protocol software can be detected by using a technique called program slicing. Those parts of the full source code that have an effect on a value of a certain variable are extracted and carefully studied. For example, spurious (extraneous) state transitions in implementations of the TCP state machine can be identified with this technique [23].

An attacker does not need to attack a victim directly. The attacker can instead target the Internet routing infrastructure to make a DoS attack [47]. Attacks against the Internet infrastructure can be divided in four classes [10]:

- *Attacks against the DNS* can be mitigated by DNS Security Extensions (DNSSEC) which provide end-to-end authenticity and integrity.
- *Routing table poisoning attacks* can be mitigated by using inconsistency checks to detect malicious updates [29]. For example, advertisements in the Border Gateway Protocol (BGP) can be compared to the information in the Internet Route Registry (IRR). Another proposal is to use DNS to verify the contents of routing advertisements. Digital signatures may also be required.
- *Packet mistreatment attacks* include misrouting of packets (e.g., to heavily loaded links), dropping of packets, and delaying of packets. Packet mistreatment attacks can be detected by running a specific detection protocol inside an Autonomous System to locate and isolate routers dropping or misrouting valid packets [6].
- *Ordinary DoS attacks* are flooding or logic DoS attacks already described in this paper. The same defense mechanisms can be used, regardless of the victim being an ordinary host or a node in the infrastructure. One simple mechanism to mitigate effects of flooding DoS attacks against DNS name servers is to increase the Time To Live (TTL) value of host IP addresses [39].

Source traceback is an important task in locating the source of attack traffic. The proposed traceback mechanisms are usually based either on recording information in routers about forwarded packets for later traceback requests or on sending additional information about the route of the packet to the victim (like ICMP traceback) [11]. It is, however, difficult to locate true origins of attack packets, because reflector attacks are based on using innocent third-parties as sources, and network address translators can make it impossible to further trace the route of the attack packets. Also, the use of a chain of stepping stones can make it impossible to locate the real attacker.

Overprovisioning of resources like access bandwidth and processing power can increase the resistance against flooding DoS attacks [7]. Despite of the related costs this defense can be an effective way to protect important targets like DNS name servers.

Cryptographic mechanisms, like IP security extensions (IPsec), can be used to authenticate message sources and encrypt messages. Authentication mechanisms (like end-to-end authentication) are suitable for preventing many logic DoS attacks, but may have difficulties in mitigating the effect of flooding DDoS attacks, because the number of compromised DDoS agents can be huge. Even if the work load on a client would be higher than that of a server, the large number of compromised hosts can still generate more traffic than a victim network or server can handle. It should also be noted that cryptographic mechanisms have difficulties in mitigating insider attacks where an attacker succeeds in getting the access rights and privileges of a legitimate user, e.g., by compromising a host.

The use of overlay networks has been proposed as a defense mechanism against DoS attacks. One example is the Secure-i3 (Secure Internet Indirection Infrastructure) [1], which decouples the act of sending a packet from the act of receiving it. Packets are sent with a logical destination identifier, and receivers express their interest in receiving packets by inserting a trigger for a specific logical identifier in the Secure-i3 overlay infrastructure. This kind of an overlay network is able to mitigate many flooding DoS attacks. In [1] it is argued that the complexity of the Secure-i3 overlay infrastructure does not introduce any new security vulnerabilities, but it is not reasonable to expect, for example, that an attacker would not be able to compromise any parts of an overlay system. Also, an overlay network cannot remove all vulnerabilities of the underlying IP infrastructure.

6.5. Automatic defending against DoS attacks in real-life systems

Automatic reaction against DoS attacks in network security devices has been studied at least in the following three research implementations or architectures: *Bro*, *Aggregate-based Congestion Control* (ACC) and *Cooperative Intrusion Traceback and Response Architecture* (CITRA). These systems can be used to defend against both deployment and attack phases.

Bro [48] is a system for detecting network intruders in real-time, and it can thus be classified as an NIDS with extra capability for defining policies with a specific language. In addition to monitoring, Bro can terminate connections by sending RST packets or ask a router to drop traffic involving a particular address.

ACC [35] tries to prevent general network congestion by detecting some of the most wide-bandwidth aggregates and rate-limiting them. ACC does not make any difference for the origin or reason of the congestion. Regardless of congestion being due to a DoS attack or a flash crowd (a burst of legitimate traffic), the same kind of congestion control is applied. The reason for this is that a congestion signature of the attack traffic will usually contain some innocent traffic, too. An ACC-enabled router tries to keep any outgoing network link fully utilized, so ACC protects mainly the availability of network link transmission capacity. By using Pushback-messages [19] an ACC-enabled router can ask an upstream router to rate-limit a requested aggregate.

CITRA [58] is an architecture for enabling the cooperation between IDSes, firewalls, routers, and other components to trace, block, and rate-limit intrusions as close to their sources as possible. CITRA is based on the Intrusion Detection and Isolation Protocol (IDIP), which is used as the communication vehicle between all parties. An important objective is to move attack mitigation actions upstream to increase the effectiveness of defenses and to minimize collateral impact on other traffic. CITRA promotes rate-limiting, because many DoS toolkits generate traffic that is difficult to differentiate from legitimate traffic.

Commercial products talk about *Intrusion Prevention Systems (IPS)*, which "prevent DDoS attacks" or "ensure network availability". A commercial IPS typically

includes not only the ability to detect but also some mechanisms to try to stop attacks e.g., by blocking the suspected attack traffic or sending TCP resets to tear down connections. Firewall, antivirus, and vulnerability-assessment capabilities can also be included in an IPS.

7. Selection of defenses

Mitigation of DoS attacks requires a comprehensive set of defenses. Implementing and applying every possible defense is, however, not feasible. This would simply cost too much in terms of resources, like humans, equipment, money, and time [33]. It is not even possible to achieve perfect network security: new vulnerabilities in computer systems are continuously found, security of one site is dependent on the security of other sites in the Internet, and some attacks are difficult or even impossible to distinguish from legitimate network traffic.

Risk management is the deliberate process of understanding the most important risks and deciding how to mitigate them [60]. The goal of risk management is to decide whether to accept a risk, mitigate it to an acceptable level, or transfer it to someone else (or any combination of these) [46]. Some risks can be accepted as such if they are not very probable or if the impact is not too critical. Risks related to the most important assets of an organization must be reduced (mitigated) either partially or completely by applying reasonable defenses. Insurance can be used to transfer a (remainder) risk to another party. The cost of handling a risk must be commensurate with the value of assets being protected.

Combating DoS attacks is primarily an exercise in risk management which must consider both technical and business aspects [27]. Major risks must be avoided, but at the same time the consumption of finite resources on security must be optimized. The costs related to security expenditures must be traded against the acquired benefits. An organization which is able to avoid major security incidents and at the same time minimize security costs is better able to stay in business and make profit.

An organization-wide *security policy* is an important prerequisite for risk management. A security policy defines the main principles (goals) for protecting the most important assets [2]. It also explains why these assets should be protected from certain threats. With the help of a security policy it is possible to concentrate consistently on the major threats and implement a cost-effective set of defenses [33].

Legislation, standards, best current practices, and other documents may dictate parts of security policy and risk management. Legislation may specify requirements for availability of public services and protection of confidential information. There are also recommended security services and procedures for ISPs who are encouraged to become proactive in security issues. The set of recommendations for ISPs in [30] includes, for example, a reasonable resistance to known security vulnerabilities in the network infrastructure. These recommendations are incentives for ISPs to defend themselves against DoS attacks.

The selected set of defenses should be seen to be dynamic. Threats, risks, and attack mechanisms change as a function of time. The set of implemented defenses should be modified to match the current requirements. For example, at the moment the most prevalent attacks exploit code-level flaws such as buffer overflow vulnerabilities. The future malicious software will probably exploit in an increasing fashion vulnerabilities in system's design, architecture, and usability [63].

7.1. Factors affecting the selection process

Choosing a cost-effective set of defenses is not a simple process, because it is difficult to compare different defense mechanisms. There are several factors that need to be considered when assessing defense mechanisms:

- Effectiveness: How capable is a defense mechanism in mitigating DoS attacks?
- Reliability: Does a defense mechanism always mitigate DoS attacks as well, or is it sometimes less effective? Is there a possibility for false positives?
- Misusability: Can an attacker exploit a defense mechanism in an unexpected way as a tool for achieving a DoS condition?
- Collateral damage: Does a defense mechanism cause any negative side effects, like performance problems in routers, a requirement for extensive human intervention to solve false positives etc.?
- Proactivity: Can a defense mechanism prevent attacks or does it only react to existing attacks?
- Completeness: What kind of other defense mechanisms are required? For example, a plain detection mechanism must be combined with a reaction mechanism.
- Reaction delay: How fast does a defense mechanism react to intrusions (e.g., has a host already been compromised and trojaned/backdoored when an intrusion has been detected)?
- Ease of implementation: Is it feasible or possible to implement a defense mechanism (e.g., the number of different organizations involved, access to source code, implementation cost worth the benefit)?
- Ease of use: Is the human interface easy to use? Does a defense mechanism fit with an already existing security infrastructure?
- Installation place: What is the optimal place to implement a defense mechanism (e.g., ISP or customer network)?

Answering these questions will help to understand the real benefit of using a certain defense mechanism to mitigate DoS attacks.

8. Conclusions

This tutorial paper has described what Denial of Service attacks are, how they can be carried out in IP networks, and how one can defend against them. It is not possible

to completely prevent these attacks because there will always be vulnerable hosts in the Internet to be compromised for attack purposes, and many DoS attack mechanisms are based on using ordinary features of protocols or network services. Also, vulnerabilities in applications can be easily exploited by malicious DoS software. In practice this means that mitigation of DoS attacks requires a comprehensive set of defense mechanisms to get defense in depth. Based on an extensive literature study, this paper gives a description of many defense mechanisms available to mitigate DoS attacks.

Worms are typically used to deploy DoS software for attacking a victim host. Theoretically, worms can infect up to millions of hosts in only tens of seconds. The selected defense mechanisms should thus cover both the DoS deployment and attack phases.

Many different defense mechanisms are typically needed to mitigate DoS attacks, but it is not cost-effective to blindly choose a large set of defense mechanisms against DoS attacks. Organizations differ in the way they do business, and this has an effect on what kind of defense mechanisms are needed. For example, a university and a web book store have very different requirements for DoS attack mitigation. Even two similar organizations will probably choose at least partially different defense mechanisms, if one organization is willing to accept a higher risk and push down the associated short-term costs. Also, the size and reputation of an organization can make a difference in the defense strategy. Attacks against well-known organizations have a higher probability of getting publicity. Even though massive, large-scale DoS attacks are found rather seldom, the risk is evident. Small-scale DoS attacks, on the other hand, are part of every-day life in the Internet. The risk of a DoS attack should not be underestimated, but it should not be overestimated either. Security policies are important in defining consistent requirements for defense mechanisms. When these requirements are known, it is easier to manage risks and achieve a reasonable tradeoff between the risk level and cost.

At minimum, any organization or individual should remove all unnecessary services, use a firewall, install relevant security patches timely, avoid using weak passwords, and use antivirus software with the most recent virus definition database.

Detection of the deployment and the attack phases can be improved by using IDSes in networks and/or hosts, using correlators and normalizers to make intrusion detection more reliable, tuning the signatures in misuse-based IDSes, and training anomaly-based IDSes.

During the DoS deployment phase additional protection can be achieved by combining attack detection with effective reaction mechanisms (e.g., to kill a process after being infected), increasing software security to prevent stack smashing attacks (based e.g., on buffer overflow and format string vulnerabilities), using vulnerability scanners, locating secret backdoors, detecting stepping stones, and restricting worm propagation rate.

During the DoS attack phase additional protection can be achieved by using a mechanism to reduce the bandwidth of a flooding DoS attack, using ingress and

egress filtering in border routers, disabling directed broadcasts in routers, enhancing the robustness against TCP SYN floodings, and using program slicing to find security flaws in protocol software.

DoS attacks against the Internet infrastructure can be mitigated by using DNSSEC, using inconsistency checks to prevent malicious route advertisements, using protocols to locate misbehaving routers, and implementing defense mechanisms to mitigate DoS attacks against nodes of the infrastructure (e.g., routers, name servers).

Other general defense mechanisms to mitigate DoS attacks include global defense infrastructures, source traceback to locate the source of attack traffic, overprovisioning of resources (e.g., bandwidth, processing power), cryptographic mechanisms, and the use of overlay networks. Many of these global defense mechanisms are dependent on cooperation between ISPs and organizations.

The current Internet is a very complex network which makes improving the security difficult. Even small improvements in security should thus be considered as a change in the right direction.

Acknowledgements

Most of this work was done when the author was with the Networking laboratory of the Helsinki University of Technology.

The author would like to thank Jorma Jormakka, Jouni Karvo, and all the anonymous reviewers for their helpful comments in improving this paper.

References

[1] D. Adkins, K. Lakshminarayanan, A. Perrig and I. Stoica, Towards a more functional and secure network infrastructure, University of California, Berkeley, Tech. Rep. UCB/CSD-03-1242, 2003.

[2] J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel and E. Stoner, State of the practice of intrusion detection technologies, Carnegie Mellon University, Software Engineering Institute, Tech. Rep. CMU/SEI-99-TR-028, Jan. 2000. [Online] Available: http://www.cert.org/archive/pdf/99tr028.pdf.

[3] I. Arce and E. Levy, An analysis of the Slapper worm, *IEEE Security & Privacy* **1**(1) (2003), 82–87.

[4] S.M. Bellovin, The state of software security, Nov. 2002. [Online] Available: http://www.research.att.com/s̃mb/talks/vuln-legal.ps.

[5] P. Bouchareine, Format string vulnerability, Hacker Emergency Response Team, Tech. Rep., July 2000.

[6] K.A. Bradley, S. Cheung, N. Puketza, B. Mukherjee and R.A. Olsson, Detecting disruptive routers: A distributed network monitoring approach, *IEEE Network* **12**(5) (1998), 50–60.

[7] R. Bush, D. Karrenberg, M. Kosters and R. Plzak, Root name server operational requirements, Internet Engineering Task Force, Request for Comments RFC 2870, June 2000.

[8] CERT Coordination Center, Denial of service attacks, Oct. 1997. [Online] Available: http://www.cert.org/tech_tips/denial_of_service.html.

[9] CERT Coordination Center, Overview of attack trends, Feb. 2002. [Online] Available: http://www.cert.org/archive/pdf/attack_trends.pdf.

[10] A. Chakrabarti and G. Manimaran, Internet infrastructure security: A taxonomy, *IEEE Network* **16**(6) (2002), 13–21.

[11] R.K. Chang, Defending against flooding-based distributed denial-of-service attacks: A tutorial, *IEEE Commun. Mag.* **40**(10) (2002), 42–51.

[12] Cisco Systems, Inc., Characterizing and tracing packet floods using cisco routers, Feb. 2003.

[13] C. Cowan, Software security for open-source systems, *IEEE Security & Privacy* **1**(1) (2003), 38–45.

[14] C. Cowan, S. Beattie, R.F. Day, C. Pu, P. Wagle and E. Walthinsen, Protecting systems from stack smashing attacks with StackGuard, in: *Proceedings of the LinuxExpo*, Raleigh, NC, USA, 1999.

[15] C. Cowan, C. Pu, D. Maier, J. Walpole, P. Bakke, S. Beattie, A. Grier, P. Wagle, Q. Zhang and H. Hinton, StackGuard: Automatic adaptive detection and prevention of buffer-overflow attacks, in: *Proceedings of the 7th USENIX Security Conference*, San Antonio, TX, 1998, pp. 63–78.

[16] S.A. Crosby and D.S. Wallach, Denial of Service via algorithmic complexity attacks, in: *Proceedings of the 12th USENIX Security Symposium*, Washington, DC, USA, 2003.

[17] R. Durst, T. Champion, B. Witten, E. Miller and L. Spagnuolo, Testing and evaluating computer intrusion detection systems, *Communications of the ACM* **42**(7) (1999), 53–61.

[18] P. Ferguson and D. Senie, *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing*, RFC 2827, May 2000.

[19] S. Floyd, S. Bellovin, J. Ioannidis, K. Kompella, R. Mahajan and V. Paxson, Pushback messages for controlling aggregates in the network, July 2001, Internet draft draft-floyd-pushback-messages-00.txt, work in progress.

[20] S. Forrest, S.A. Hofmeyr, A. Somayaji and T.A. Longstaff, A sense of self for Unix processes, in: *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, 1996, pp. 120–128.

[21] L. Garber, Denial-of-Service attacks rip the Internet, *IEEE Computer* **33**(4) (2000), 12–17.

[22] D. Gollmann, *Computer Security*, John Wiley & Sons, Chichester, England, 1999.

[23] B. Guha and B. Mukherjee, Network security via reverse engineering of TCP code: Vulnerability analysis and proposed solutions, *IEEE Network* **11**(4) (1997), 40–48.

[24] J. Haines, D.K. Ryder, L. Tinnel and S. Taylor, Validation of sensor alert correlators, *IEEE Security & Privacy* **1**(1) (2003), 46–56.

[25] M. Handley, V. Paxson and C. Kreibich, Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics, in: *Proceedings of the 10th USENIX Security Symposium*, 2001.

[26] K.J. Houle, G.M. Weaver, N. Long and R. Thomas, *Trends in Denial of Service Attack Technology*. CERT Coordination Center, Oct. 2001. [Online] Available: http://www.cert.org/archive/pdf/DoS_trends.pdf.

[27] A. Householder, A. Manion, L. Pesante, G.M. Weaver and R. Thomas, *Managing the Threat of Denial-of-Service Attacks*, CERT Coordination Center, Oct. 2001.

[28] J.D. Howard, An analysis of security incidents on the Internet 1989–1995, PhD dissertation, Carnegie Mellon University, April 1997.

[29] C. Huitema, *Routing in the Internet*, 2nd edn, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.

[30] T. Killalea, *Recommended Internet Service Provider Security Services and Procedures*, RFC 3013, Nov. 2000.

[31] Lawrence Livermore National Laboratory and Sandia National Laboratories, Intrusion detection and response, Dec. 1996. [Online] Available: http://www.all.net/journal/ntb/ids.html.

[32] G. Lawton, Virus wars: Fewer attacks, new threats, *IEEE Computer*, **35**(12) (2002), 22–24.

[33] W. Lee, W. Fan, M. Miller, S.J. Stolfo and E. Zadok, Toward cost-sensitive modeling for intrusion detection and response, *Journal of Computer Security* **10**(1–2) (2002).

[34] R.P. Lippmann, D.J. Fried, I. Graf, J.W. Haines, K.R. Kendall, D. McClung, D. Weber, S.E. Webster, D. Wyschogrod, R.K. Cunningham and M.A. Zissman, Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation, in: *Proceedings of the DARPA Information Survivability Conference and Exposition*, 2000.

[35] R. Mahajan, S.M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson and S. Shenker, Controlling high bandwidth aggregates in the network, *ACM SIGCOMM Computer Communication Review* **32**(3) (2002), 62–73.

[36] C. Manikopoulos and S. Papavassiliou, Network intrusion and fault detection: A statistical anomaly approach, *IEEE Commun. Mag.* **40**(10) (2002), 76–82.

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

[38] J. Mölsä, Effectiveness of rate-limiting in mitigating flooding DoS attacks, in: *Proceedings of the Third IASTED International Conference on Communications, Internet, and Information Technology at St. Thomas, US Virgin Islands*, M.H. Hamza, ed., ACTA Press, Anaheim, CA, USA, 2004, pp. 155–160.

[39] J. Mölsä, Mitigating DoS attacks against the DNS with dynamic TTL values, in: *Proceedings of the Ninth Nordic Workshop on Secure IT Systems*, S. Liimatainen and T. Virtanen, eds, Espoo, Finland, 2004, pp. 118–124.

[40] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford and N. Weaver, Inside the Slammer worm, *IEEE Security & Privacy* **1**(4) (2003), 33–39.

[41] D. Moore, C. Shannon and J. Brown, Code-Red: a case study on the spread and victims of an Internet worm, in: *Proceedings of the Internet Measurement Workshop*, Marseille, France, 2002.

[42] D. Moore, C. Shannon, G.M. Voelker and S. Savage, Internet quarantine: Requirements for containing self-propagating code, in: *Proceedings of the IEEE Infocom*, 2003.

[43] D. Moore, G.M. Voelker and S. Savage, Inferring Internet denial-of-service activity, in: *Proceedings of the 10th USENIX Security Symposium*, Washington, DC, 2001.

[44] P. Mueller and G. Shipley, Dragon claws its way to the top, *Network Computing* (August 20) (2001) 45–67.

[45] B. Mukherjee, L.T. Heberlein and K.N. Levitt, Network intrusion detection, *IEEE Network* **8**(3) (1994), 26–41.

[46] S. Northcutt and J. Novak, *Network Intrusion Detection*, 3rd edn, New Riders Publishing, Indiana, IN, 2002.

[47] P. Papadimitratos and Z.J. Haas, Securing the Internet routing infrastructure, *IEEE Commun. Mag.* **40**(10) (2002), 60–68.

[48] V. Paxson, Bro: A system for detecting network intruders in real-time, *Computer Networks* **31**(23-24) (1999), 2435–2463.

[49] V. Paxson, An analysis of using reflectors for distributed denial-of-service attacks, *ACM SIGCOMM Computer Communication Review* **31**(3) (2001).

[50] T.H. Ptacek and T.N. Newsham, *Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection*, Secure Networks, Inc., 1998.

[51] J. Reason, *Managing the Risks of Organizational Accidents*, Ashgate Publishing Company, Burlington, USA, 1997.

[52] C.L. Schuba, I.V. Krsul, M.G. Kuhn, E.H. Spafford, A. Sundaram and D. Zamboni, Analysis of a Denial of Service attack on TCP, in: *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, 1997, pp. 208–223. [Online] Available: https://www.cerias.purdue.edu/techreports-ssl/public/97-06.ps.

[53] SecuriTeam, Kiss of Death – a new Denial of Service attack, 1999.

[54] D. Senie, *Changing the Default for Directed Broadcasts in Routers*, RFC 2644, 1999.

[55] C. Shannon and D. Moore, The spread of the Witty worm, CAIDA, Tech. Rep., 2004.

[56] C. Shannon, D. Moore and K.C. Claffy, Beyond folklore: Observations on fragmented traffic, *IEEE/ACM Trans. Networking* **10**(6) (2002) 709–720.

[57] S. Staniford, V. Paxson and N. Weaver, How to 0wn the Internet in your spare time, in: *Proceedings of the 11th USENIX Security Symposium*, San Francisco, CA, 2002.

[58] D. Sterne, K. Djahandari, B. Wilson, B. Babson, D. Schnackenberg, H. Holliday and T. Reid, Autonomic response to distributed Denial of Service attacks, in: *Proceedings of Recent Advances in Intrusion Detection, 4th International Symposium*, Davis, CA, 2001, pp. 134–149.

[59] R. Stone, Centertrack: An IP overlay network for tracking DoS floods, in: *Proceedings of the 9th USENIX Security Symposium*, Denver, CO, 2000.

[60] US Department of Homeland Security, Critical infrastructure, glossary of terms and acronyms.

[61] C.D. Wickens and J.G. Hollands, *Engineering Psychology and Human Performance*, 3rd edn, Prentice Hall, Upper Saddle River, NJ, USA, 2000.

[62] M.M. Williamson, Throttling viruses: Restricting propagation to defeat malicious mobile code, HP laboratories, Bristol, Tech. Rep. HPL-2002-172, June 2002.

[63] J.M. Wing, A call to action: Look beyond the horizon, *IEEE Security & Privacy* **1**(6) (2003), 62–67.

[64] Y. Zhang and V. Paxson, Detecting backdoors, in: *Proceedings of the 9th USENIX Security Symposium*, Denver, CO, 2000.

[65] Y. Zhang and V. Paxson, Detecting stepping stones, in: *Proceedings of the 9th USENIX Security Symposium*, Denver, CO, 2000.