computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209
International Scientific
Journal of Computing

MODERN APPROACHES OF SECURING COMPUTER NETWORKS FROM DENIAL OF SERVICE ATTACKS

Andrian Piskozub

National University "Lviv Politechnic", Ukraine, piskozub@polynet.lviv.ua

Abstract: *The aim of this paper is to understand reasons why denial of service (DoS) attacks are happening; to find ways how to avoid these attacks or lessen their influence; to work out strategy of detecting and preventing these attacks.*

Keywords: *denial of service attack, distributed attack, flooding attack, network security, firewall, intrusion detection system, traffic shaping, application level defense, TCP Interception, IP hopping, scanner.*

1. INTRODUCTION

DoS attacks first appeared in the early nineties and up to 2000 became one of the most dangerous attacks in internet. From 2000 up to now thousands of DoS attacks have been taking place every week and their complexity is growing all over the time. The reasons of this phenomenon are: plenty of vulnerabilities in software; time to exploit now is measured in days or even hours; automated attack tools have been rapidly improving.

There are 3 main reasons for automation of attacks detection and prevention:

- as the number and frequency of attacks are multiplied all the time, it is very important to identify the attacks on the early stage of their execution and react on them in time;
- in the critical case interference with the attack must be realized faster, than a man is able to react;
- automated attack tools are used more often.

2. DOS BASICS

The aim of this paper is to understand reasons why DoS attacks happen; to find ways how to avoid these attacks or lessen their influence; to work out strategy of detecting and preventing these attacks.

2.1. DOS VARIANTS

The DoS attacks classification is presented on Figure1:

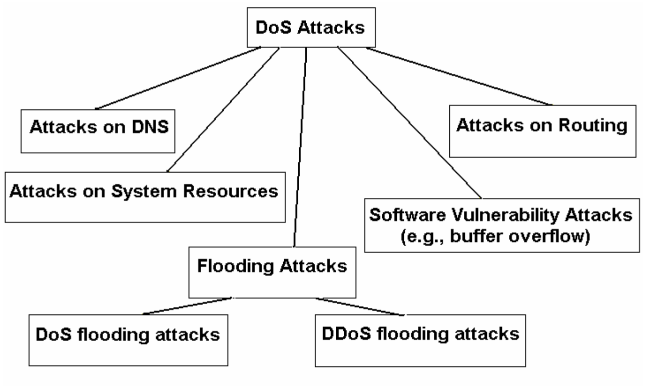


Figure 1. DoS Attacks Classification

Software vulnerability attacks are based on imperfect software. A lot of programs use stack for their operation. This is a prerequisite for the so-called buffer overflow exploits, which make stack ("smashing"). In case of success the hacker gains access on victim computer with the privilege of the service that was compromised.

In case of an unsuccessful buffer overflow attack, the targeted process on victim computer may simply stop responding and in this case we consider it a DoS attack.

We may suggest the following solutions for this case:

- use the newest operating system (OS) version, apply the latest service packs and patches. Use the latest version of application software packages;
- to prevent buffer overflow attacks:
  - for compiling programs it is necessary to use a special compiler such as StackGuard [1], which does not allow to change the return address in stack

(due to usage of the so-called canary word, which is placed just after the return address during function call. If this canary word differs from its original value after finishing the function operation, a buffer overflow attempt has been taking place);

- use Immunix [2] – version of Linux, fully compiled using StackGuard;
- use the Openwall patch [3] for Linux kernel – it does not allow to execute code from the stack.

Sometimes computer criminals try to attack systems by exhausting all system resources of victim computers, e.g., using CPU or memory resources, flooding hard disk space with unnecessary information, etc. The following solutions for this case are:

- restrict system resources by using ulimit (e.g., memory, CPU, maximum number of open file descriptors etc.);
- use quota to restrict utilization of hard disk space;
- use other application specific restrictions.

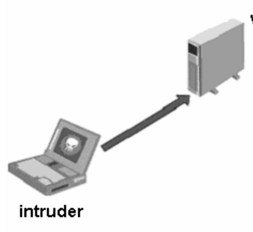
The attacks on routing are based on records manipulation of routing tables that can result in the termination of service to legitimate systems and networks. Most routing protocols, such as RIPv1 or BGP, generally use no or weak authentication algorithms. As a result of such attacks, the traffic of victim network is routed through attacker's network or to nowhere. To avoid this, we should use routing protocols with strong authentication.

Attacks on DNS are happening more often lately as soon as current DNS version uses both UDP and TCP protocols. This fact allows under certain conditions to spoof DNS records which gives the attacker the possibility to redirect users traffic to attacker's machine or to nowhere. There is no universal solution for current version of DNS against this kind of attacks for the time being.

Lately the most devastating kind of attacks became the so-called flooding DoS attacks. To consider their nature and elaborate strategies and solutions against them we have to consider models and stages of attacks.

2.2 MODELS AND STAGES OF ATTACKS

The traditional attack models are built on principle "one to one" connection (Figure 2) or "one to many" connection (Figure 3), i.e. attack comes from one source. Developers of network defense facilities (firewalls, intrusion detection systems (IDS), etc) were oriented exactly to these traditional attack models. However such models do not manage with the relatively recently (in 1998) discovered threat - distributed attacks. In the models of the distributed or coordinated attack other principles are used - “many to one” connection (Figure 4) and “many to many” connection (Figure 5) [4].



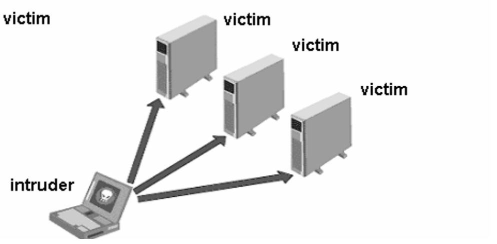
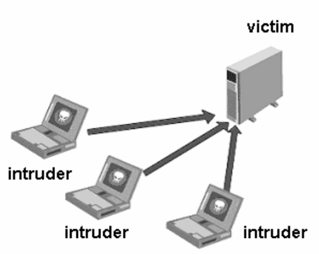


Figure 2. Connection “one to one”



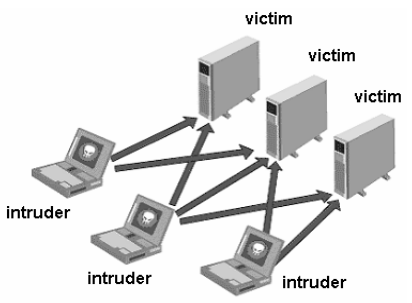


Figure 4. Connection “many to one” **Figure 5. Connection “many to many”**

The distributed attacks are based on the classic buffer overflow attacks, or rather on their subset, also known as flood or storm attacks. The principle of these attacks consists in sending a storm of packets directed at the victim system, so that the targeted system will “choke” and it will cause a denial of service attack. However for an attacker to succeed in a flood attack, it is important to have a channel with a higher bandwidth that the one of the targeted system. This is not always possible if the classical attack model is used, but in case of distributed attack model the situation cardinally changes.

It is important to understand that every attack has its three stages – information gathering, exploitation and wiping out tracks (Figure 6) [4].

information gathering
exploitation
wipe out tracks

Figure 6. Stages of attacks realization

Traditional security tools, such as firewalls, operate on the second stage, "forgetting" about the first and the third stages. That is why it is very difficult to stop attack even using powerful and expensive security tools. Besides, traditional perimeter security technologies such as firewalls and IDS do not provide adequate distributed DoS (DDoS) protection, since filtering solutions such as router–based access control lists (ACLs) simply can't separate good traffic from bad for most attacks, resulting in legitimate transactions being blocked.

Some well known DoS flood attacks are presented in table 1 [5].

Basic model of DDoS tools is shown on Figure 7.

For realization of DDoS attack to be successful and to involve as many computers as possible intruder has to compromise (zombify) them:

- •he (she) scans internet for finding known vulnerabilities;
- •compromises vulnerable hosts for gaining access;
- •installs DDoS tool on every compromised machine;
- •uses compromised hosts for subsequent scanning and compromise other hosts.

Some well known DDoS tools are presented in table 2 [6].

3. DEFENSE AGAINST DOS AND DDOS FLOOD ATTACKS

The defense methods against DoS and DDoS attacks are as follows: OS modification, firewalling, IDS, traffic shaping, application level defense, TCP Interception and IP hopping.

3.1 OS Modification

It is necessary to use hardened operation systems (OS) and platforms - now OS can reduce DoS'ing ability (e.g., stand against SYN flood attacks) due to increase of queue size for TCP-connections establishment and reducing timeouts. Advantages here are as follows: simple to implement, the fastest possible solution. Drawbacks: it is not the best way of problem solution, it should be used only together with other solutions.

Table 1. Some well known DoS flood attacks

Name of Attack	Flooding Capabilities	Short Description
SYN	TCP	Sending large numbers of TCP connection initiation requests to the target. The target system must consume resources to keep track of these partially opened connections.
ICMP flood	ICMP	Sending large numbers of ICMP Echo Request packets to the target system, thus tying up network resources.
Smurf	ICMP	ICMP (Internet Control Message Protocol) ping requests to a directed broadcast address. The forged source address of the request is the target of the attack. The recipients of the directed broadcast ping request respond to the request and flood the target's network.
Fraggle	UDP	Same as Smurf, but rather than ICMP uses UDP to broadcast address for amplification.
UDP flood	UDP	Sending large numbers of UDP (User Datagram Protocol) packets to the target system, thus tying up network resources.
TCP flood	TCP NUL, TCP RST, TCP ACK	When TCPs communicate, each TCP allocates some resources to each connection. By repeatedly establishing a TCP connection and then abandoning it, a malicious host can tie up significant resources on a server.

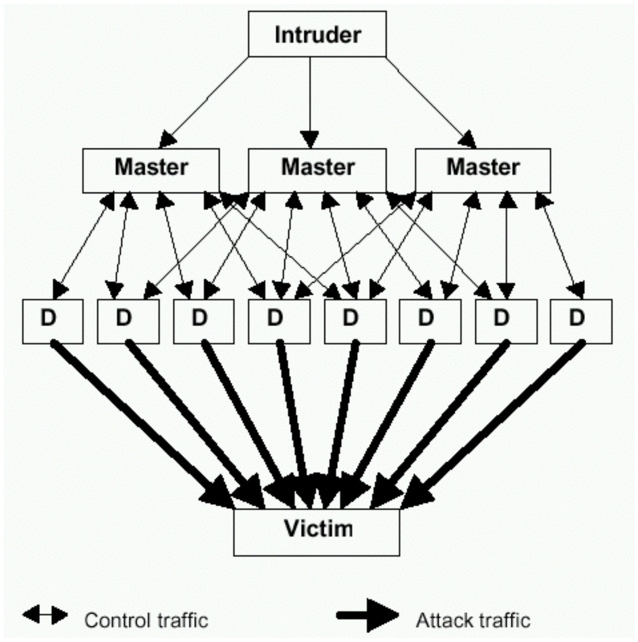


Figure 7. Basic model of DDoS tools

Table 2. Some well known DDoS tools.

Name of Tool	Flooding Capabilities	Short Description
TFN	UDP, ICMP Echo, TCP SYN, Smurf	Uses IP spoofing. Uses ICMP Echo reply packets to communicate between zombie and master.
Trinoo	UDP	Only initiates UDP attacks to random ports. Communication between master and slave is via unencrypted TCP and UDP. No IP spoofing. Uses UDP ports 27444 and 31335.
Stacheldracht v2.666	UDP, ICMP, TCP SYN, Smurf, TCP ACK, TCP NUL	Uses encryption for communications (but not for ICMP heartbeat packets that zombie sends to master) and has an auto-update feature (via rcp). Has ability to test (via ICMP Echo) if it can use spoofed IP addresses.
TFN 2K	UDP, ICMP Echo, TCP SYN, Smurf	Same as TFN - but the slave is silent so difficult to spot. No return info from slave. Zombie to master communication is encrypted.
Targa	ANY	Works by sending malformed IP packets known to slow down or hangup many TCP/IP network stacks.
NAPTHA	TCP	Naptha attacks exploit weaknesses in the way some TCP stacks and applications handle large numbers of connections in states other than "SYN RECVD," including "ESTABLISHED" and "FIN WAIT-1."

3.2 FIREWALLING

By using firewalls we can: block ports and protocols used by DoS and DDoS tools; do ingress and egress filtering against spoofed IP packets [7]. Advantages: ingress and egress filtering is considered the best solution against spoofing attacks. Drawbacks: firewall does simply block communication from a specific IP address or port and has no ability to drop the packets that contain the attack and allow normal traffic to go through; the blocking ports technique is useless for DDoS tools that use arbitrary ports.

3.3 INTRUSION DETECTION SYSTEMS

Not so long ago, a good practice for network security was to use IDS with DoS and DDoS attack detection. Now we should consider using IDS with DoS and DDoS attack prevention – so called inline IDS. Whereas intrusion detection is designed to make the security administrator aware of potential attacks, intrusion prevention goes one step farther and works actively to prevent the intrusion. E.g., SNORT, which is a well-known IDS, can be compiled in both modes – as a normal or inline IDS. Advantages: in order to prevent attack inline IDS are closely integrated with firewalls so they together have ability to drop the packets that contain the attack and allow normal traffic to go through.

Drawbacks: in case of misconfiguration or malfunctioning of inline IDS the possibility of "false positives" could interrupt normal network functionality.

3.4 TRAFFIC SHAPING

Traffic shaping is relatively new solution which successfully complements all other solutions mentioned here. We will consider two solutions of traffic shaping on the base of Linux border router to reduce the influence of DoS flood attack originating from Internet (Figure 8) and from inside of our network (Figure 9).

In case of DoS flood attack originating from Internet it is necessary to implement ingress shaping. It can be done in 2 ways - using ingress queuing discipline (qdisc) or intermediate queuing device (IMQ).

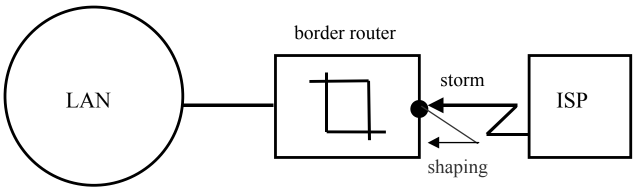


Figure 8. Border router on Linux platform with ingress shaping

Another example of border router on Linux platform with egress shaping is shown on Figure 9. Its purpose is to reduce the influence of DoS flood attack originating from our network (from zombified hosts) to Internet. But this is not the primary solution – we should better avoid zombifying our hosts completely and use for this purpose other mentioned here preventive recommendations. Egress shaping here results only in reduction of DoS attack influence. For realization of egress shaping it is necessary to use whatever egress qdisc, preferably classful one like class based queueing (CBQ) or hierarchical token bucket (HTB) qdiscs. Advantages: this is the

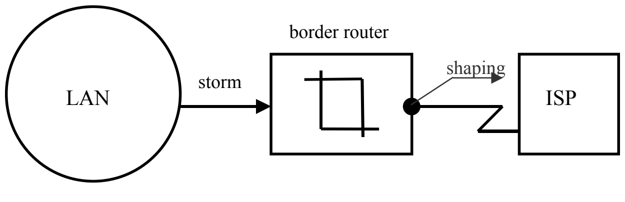


Figure 9. Border router on Linux platform with egress shaping

most powerful solution if we need to supply some public legitimate services, limiting at the same time the influence of DoS and DDoS flooding attacks. Drawbacks: not known for traffic shaping. The examples of scripts that perform ingress and egress shaping are presented later in this paper.

3.5 APPLICATION LEVEL DEFENSE. EXAMPLES

We present here some examples which can be applied on application level. First of all we can use features in daemons of limiting simultaneous sessions per source IP address. Certainly, not all daemons have such features.

Second, if start daemons from *xinetd* daemon, we can use some its features, e.g., *cps*, *instances*, *max_load*, and *per_source* keywords. Here is example of *xinetd* configuration file, situated in */etc/xinetd.conf or /etc/xinetd.d/someservice* (if this configuration is special only for this *someservice*):

Example 1. xinetd *configuration file*

```
service myservice {
# Limit to 10 connections per second. If the limit is exceeded, sleep for
#30 seconds.
cps = 10 30
# Limit to 8 concurrent instances of myservice.
instances = 8
# Limit  to 5 simultaneous sessions per source IP address. Specify
# UNLIMITED for no limit, the default.
per_source = 5
# Reject new requests if the one-minute system load average exceeds 3.0.
max_load = 3.0
}
```

Another example of application level defense is the mod_dosevasive [8] module for the Apache web server, that provides evasive action in the case of an HTTP DoS or DDoS attack or brute force attack. This is done by denying any single IP address from any if the following events take place: requesting the same page more than a few times per second; making more than 50 concurrent requests on the same child per second; making any requests while temporarily blacklisted (on a blocking list).

One more example of application level defense software – so-called DNS Flood Detector [9], which detects abusive usage levels on high traffic name servers and makes quick response in halting the use of one's name server to facilitate spam.

3.6 TCP INTERCEPT

TCP Intercept software, implemented in particular by Cisco products [10], intercepts TCP

SYN packets and establishes a connection with the client on behalf of the destination server, and if successful, establishes the connection with the server on behalf of the client and knits the two half-connections together transparently.

3.7 IP HOPPING

IP hopping [11] is very interesting DoS and DDoS attacks avoiding technique. If attack tools target a specific IP address or addresses and do not perform DNS lookups to resolve an IP address from a name, it may be possible to avert an attack by changing the IP address used to access the target.

If the attacker monitors the attack, then repeated and automated IP address changes may be performed to avert or mitigate the effects of the attack. Two approaches can be taken: to have the mechanisms for IP address changing in place and only enable them if attacked, or conduct regular and frequent IP address changes as a matter of course, even if an attack is not present.

4. DEFENSE STRATEGY AGAINST DDOS ATTACKS

The best defense strategy against DDoS attacks is:

- to defend our system or network against being attacked;
- to defend our system or network against being zombified and not be used as an amplifying unit for further DDoS attacks;
- to scan our system or network and detect already zombified hosts.

4.1. DEFENSE OF SYSTEM OR NETWORK AGAINST BEING ATTACKED

We suggest the following solutions:

- general solutions in all cases:
  1. Use ingress and egress filtering on routers and other filtering devices.
  2. Routers and other filtering devices can drop packets to non-required TCP and UDP ports.
  3. We should contact our ISP.
- SYN Flood attack:
  1. Use OS modification - increase of queue size for TCP-connections establishment and reducing timeouts.
  2. Use TCP SYN-cookie support (in Linux).
  3. Use TCP intercept.
  4. Use traffic shaping of TCP SYN-packets.
  5. Use inline IDS (if possible).
- ICMP Flood attack:
  1. Block ICMP Echo Request packets on firewall.
  2. If You can not block these packets – use traffic shaping to make some reasonable rate on them.
  3. Use inline IDS (if possible).
- UDP Flood attack:
  1. Use traffic shaping of necessary UDP-packets (e.g., UDP port 53).
  2. Use inline IDS (if possible).SMURF attack:
  1. Completely block ICMP Echo Reply packets or block ICMP Echo Reply packets without corresponding ICMP Echo Request packets(using stateful inspection firewalls).
- Fraggle attack:
  1. Block UDP/TCP Echo (and Chargen and Discard) packets on firewall.

4.2 DEFENSE OF SYSTEM OR NETWORK AGAINST BEING ZOMBIFIED

We suggest the following solutions in order not to be zombified with the clients, handlers or agents installed, and not to be used as an amplification system in further DDoS attacks and to minimize consequences:

- general recommendations in all cases of DDoS tools:
  1. As soon as all DDoS tools use mentioned above DoS flood attacks, all corresponding security considerations are correct and needed to be taken.
  2. To prevent exploitation of our hosts at the initial stage of attack we should: use only necessary services; install the latest patches and service packs for operating systems and applications; set the adequate access rights on directories and files; regularly audit our systems; perform cryptography check of our every system – configuration, system files, installed software, etc.(e.g., use Tripwire).
  3. On our border routers we should allow only necessary TCP and UDP traffic.
  4. Use IDS (if possible, use inline IDS).
  5. Contact our ISP and ask him to use ingress and egress filtering on his border router.
- TFN [12]:
  1. To prevent the operation all ICMP Echo Reply traffic should be blocked on the border router. If ICMP cannot be blocked, disallow unsolicited ICMP Echo Reply packets.
- Stacheldraht [13]:
  1. All ICMP Echo Reply traffic should be blocked on the border router. If ICMP cannot be blocked, disallow unsolicited ICMP Echo Reply packets.
- Trinoo [14]:

1. All unnecessary UDP traffic should be blocked on the border router (e.g., only legitimate DNS traffic should be allowed).

- TFN2K [15]:

1. Use a firewall that exclusively employs application proxies. This should effectively block all TFN2K traffic. Exclusive use of application proxies is often impractical, in which case the allowed non-proxy services should be kept to a minimum.
2. Disallow unnecessary ICMP, TCP, and UDP traffic. Typically only ICMP type 3 (destination unreachable) packets should be allowed.
3. If ICMP cannot be blocked, disallow unsolicited (or all) ICMP Echo Reply packets.
4. Disallow UDP and TCP, except on a specific list of ports.

- NAPTHA [16]:

1. Limit the amount of services running on any system You suspect that might become victim to a Naptha attack, especially public systems.
2. Limit access as to who can connect to exposed TCP ports on a system via firewalling techniques. On public systems this may be impractical, but it should be limited just the same if possible.
3. On Unix systems, use xinetd (inetd) or possibly Dan Bernstein's tcpserver [17] to limit spawned daemon processes. While this will not prevent that particular daemon's resources from being over utilized, it is possible to prevent daemons from crashing the server. This may allow the server to recover.
4. OS modification - adjustments for TCP timeouts and keepalives to potentially allow for quicker recovery (assuming that the Naptha attack did not crash the system). For example, the TCP keepalive settings for Linux might help recovery time: *Example 2. Linux kernel configuration file*

```
#!/bin/sh
# keepalive time is adjusted from 2 hours to 30 seconds
echo 30 > /proc/sys/net/ipv4/tcp_keepalive_time

# the number of keepalive probes is adjusted from 9 to 2
echo 2 > /proc/sys/net/ipv4/tcp_keepalive_probes

# the maximum number of probes sent out to be 100 instead of just 5
echo 100 > /proc/sys/net/ipv4/tcp_max_ka_probes
```

The suggested zombification prevention solutions sometimes are not 100% feasible due to drawbacks such as blocking legitimate traffic, etc. In this case, some compromise in company security policy should be taken and legitimate traffic should be allowed.

4.3. SCANNING SYSTEMS TO FIND ALREADY ZOMBIFIED HOSTS

We may suggest some well known scanning and other useful tools to find already zombified hosts:

- Robin Keir's scanner DDoSPing v2.0 [18]
- Razor's Zombie Zapper utility [19] (works against Trinoo, TFN, Stacheldraht, Troj_Trinoo and Shaft assuming that the default passwords have not been changed. That is why this software will not work against TFN2K)
- NIPC's scanner find_ddos [20]
- David Brumley's RID [21]
- David Dittrich's dds (trinoo/TFN/stacheldraht agent detector) [22]
- David Dittrich's gag (stacheldraht agent detector) [22]
- Simple Nomad's tfn2kpass (tfn2k password recovery tool. from a td or tfn binary) [23].

5. EXAMPLE OF REALIZATION OF LINUX ADVANCED ROUTER WITH ANTI-DOS FEATURES

We suggest some scripts written for Linux OS to perform some modifications that where explained in this material. The first example script (Example 3) is kernel modification script.

Example 3. Linux kernel configuration file

```
#!/bin/sh
#Disabling IP Spoofing attacks.
echo "2" > /proc/sys/net/ipv4/conf/all/rp_filter
#Don't respond to broadcast pings (Smurf-amplifier-protection)
echo "1" >
/proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
#Block source routing
echo "0" >
/proc/sys/net/ipv4/conf/all/accept_source_route
#Kill timestamps
echo "0" > /proc/sys/net/ipv4/tcp_timestamps
#Enable TCP SYN Cookies
echo "1" > /proc/sys/net/ipv4/tcp_syncookies
#Kill redirects
echo "0" >
/proc/sys/net/ipv4/conf/all/accept_redirects
#Enable bad error message protection
```

```
echo "1" >
/proc/sys/net/ipv4/icmp_ignore_bogus_error_respon
ses
#Log martians (packets with impossible addresses)
echo "1" > /proc/sys/net/ipv4/conf/all/log_martians
#Reduce DoS'ing ability by reducing timeouts
echo "30" > /proc/sys/net/ipv4/tcp_fin_timeout
echo "2400" >
/proc/sys/net/ipv4/tcp_keepalive_time
echo "0" > /proc/sys/net/ipv4/tcp_window_scaling
echo "0" > /proc/sys/net/ipv4/tcp_sack
```

The second example script (Example 4) is part of *iptables* firewall script with TCP SYN flood protection.

Example 4. Part of iptables firewall script with TCP SYN flood protection

```
#!/bin/sh
#…
#TCPACCEPT - Check for SYN-Floods before
letting TCP-Packets in
# Overall Limit for TCP-SYN-Flood detection
TCPSYNLIMIT="5/s"
# Burst Limit for TCP-SYN-Flood detection
TCPSYNLIMITBURST="10"
# Creating chain for TCP protocol with TCP SYN
flood protection
$IPTABLES -N TCPACCEPT
# Accepting to this chain only allowed limit of TCP
SYN packets
$IPTABLES -A TCPACCEPT -p tcp --syn -m limit
--limit  $TCPSYNLIMIT \
--limit-burst $TCPSYNLIMITBURST -j ACCEPT
# All other TCP packet beyond the settled threshold
– drop
$IPTABLES -A TCPACCEPT -p tcp --syn -j DROP
# Accept all other TCP (not TCP SYN) packets
$IPTABLES -A TCPACCEPT -p tcp ! --syn -j
ACCEPT
```

The third example script (Example 5) is part of script from LARTC maillist which uses *Ingress* qdisc for SYN Flood protection [24].

Example 5. Script which uses Ingress qdisc for TCP SYN flood protection

```
#!/bin/sh
# tag all incoming SYN packets through $INDEV as
mark value 1
$iptables -A PREROUTING -i $INDEV -t mangle -
p tcp --syn   -j MARK  \
--set-mark 1
# install the ingress qdisc on the ingress interface
$TC qdisc add dev $INDEV handle ffff: ingress
# SYN packets are 40 bytes (320 bits) so three SYNs
equals
# 960 bits (approximately 1kbit); so we rate limit
below
# the incoming SYNs to 3/sec (not very useful
really)
$TC filter add dev $INDEV parent ffff: protocol ip
prio 50 handle 1 fw police rate \ 1kbit burst 40 mtu
9k drop flowid :1
```

The fourth example script (Example 6) is part of script which uses *intermediate queueing device* (IMQ) for UDP flood protection of DNS server.

Example 6. Part of script which uses intermediate queueing device (IMQ) for UDP flood protection of DNS server

```
#!/bin/sh
#…
 $IPTABLES -t mangle -N MYSHAPER-IN
 $IPTABLES -t mangle -I PREROUTING -i eth0 -j
MYSHAPER-IN
 /sbin/modprobe imq numdevs=1
 $IP link set imq0 up
 $TC qdisc add dev imq0 root handle 10: htb default
17 r2q 1
 $TC class add dev imq0 parent 10: classid 10:1 htb
rate 1700kbit \
quantum 3030
# let's consider UDP traffic to our public DNS server
to be no more
# 100kbit/s
 $TC class add dev imq0 parent 10:1 classid 10:12
htb rate 100kbit prio 2
 $TC qdisc add dev imq0 parent 10:12 handle 12:
sfq perturb 10
 $TC filter add dev imq0 parent 10:0 prio 0 protocol
ip handle 12 fw \
flowid 10:12
 $IPTABLES -t mangle -A MYSHAPER-IN -p udp
--dport 53 -j MARK \
--set-mark 12
# DEFAULT class with the lowest prio
 $TC class add dev imq0 parent 10:1 classid 10:17
htb rate 30kbit ceil  128kbit \
prio 5
 $TC qdisc add dev imq0 parent 10:17 handle 17:
sfq perturb 10
# finally, instruct these packets to go through the
imq0
 $IPTABLES -t mangle -A MYSHAPER-IN -j IMQ
--todev 0
```

6. SOME COMMERCIAL ANTIDOS FEATURES

Although the task of our paper was to suggest some solutions against DoS and DDoS attacks on noncommercial platforms like Linux, we have to mention that there are a lot of powerful commercial solutions from such famous IT companies as Cisco, Captus Networks, Foundry Networks, Mazu Networks, Radware, Reactive Network Solutions, Top Layer Networks and many others. For example, Cisco, one of the world's networking solutions leaders, has some valuable solutions against DoS flooding attacks [25], which are as follows:

• against IP source address spoofing:
    -Unicast Reverse Path Forwarding (Unicast RPF)
• against SYN Flood attack:
    -TCP Intercept
• against SMURF attack:
    -"no ip directed-broadcast"
• against any flooding attack:
    -Committed Access Rate (CAR)

7. CONCLUSIONS

DDoS attacks are already among the most difficult to defend against. There is no 100-percent protection from DoS and DDoS attacks. But, what can be done, is limitation of their influence over the important parts of networks.

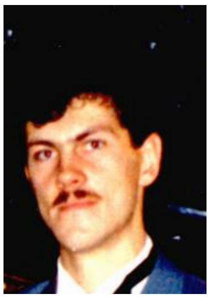
DDoS attacks should not be considered separately from other types of attacks since they use the whole arsenal of different kind of attacks in them.

Responding to and defeating these attacks in a timely and effective manner is the primary challenge confronting Internet–dependent organizations today.

8. REFERENCES

[1] Compiler StackGuard. http://immunix.org
[2] Immunix Project. http://www.immunix.com
[3] Openwall Linux Kernel Patch. http://www.openwall.com/linux/
[4] Lukatskyj A.V. Attacks Detection. –SPb.: BHV-Petersburg, 2001.
[5] Riverhead Networks: DDoS Attacks. http://www.riverhead.com/re/generic_ddos.html
[6] Riverhead Networks: DDoS Tools. http://www.riverhead.com/re/known_ddos_tools.html
[7] P. Ferguson, D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing. RFC2827, May 2000.
[8] Nuclear Elephant: evasive maneuvers module for Apache mod_dosevasive. http://www.nuclearelephant.com/projects/dosevasive/
[9] DNS Flood Detector. http://www.adotout.com/dnsflood.html
[10] Cisco: Configuring TCP Intercept . http://www.cisco.com/univercd/cc/td/doc/product/software/ios113ed/113ed_cr/secur_c/scprt3/scdenial.htm
[11] Federal Computer Incident Response Center. Defense Tactics for Distributed Denial of Service Attacks. http://www.fedcirc.gov/docs/DDOS-defense.PDF
[12] D. Dittrich. "The Tribe Flood Network" Distributed Denial of Service Attack Tool. http://staff.washington.edu/dittrich/misc/tfn.analysis
[13] D. Dittrich. "The stacheldraht" Distributed Denial of Service Attack Tool. http://staff.washington.edu/dittrich/misc/stacheldraht.analysis
[14] D. Dittrich. "The DoS Project's trinoo" Distributed Denial of Service Attack Tool. http://staff.washington.edu/dittrich/misc/trinoo.analysis
[15] J. Barlow, W. Thrower. TFN2K - An Analysis. AXENT Security Team. March 7, 2000. http://packetstorm.decepticons.org/distributed/tfn.analysis.txt
[16] R. Keyes. The Naptha DoS Vulnerabilities. Razor: Security Advisories and Publications. November 30, 2000. http://razor.bindview.com/publish/advisories/adv_NAPTHA.html
[17] Dan Bernstein's tcpserver. http://cr.yp.to/ucspi-tcp.html
[18] Robin Keir's DDoSPing Scanner. http://www.keir.net
[19] Razor's Zombie Zapper Utility. http://razor.bindview.com
[20] NIPC's scanner find_ddos. http://www.nipc.gov
[21] David Brumley's RID. http://www.theorygroup.com/Software/RID

[22] David Dittrich's DDoS detectors. http://staff.washington.edu/dittrich/misc/ddos_scan.tar
[23] Simple Nomad's tfn2kpass (tfn2k password recovery tool). http://razor.bindview.com/
[24] Linux Advanced Routing & Traffic Control HOWTO. Protecting your host from SYN floods. http://en.tldp.org/HOWTO/Adv-Routing-HOWTO/lartc.cookbook.html
[25] Cisco. IOS Essential Features. http://www.cisco.com/public/cons/isp



Andrian Piskozub
graduated from Lviv Polytechnic Institute in 1993 on the speciality "Automation and Telemechanics". In 1997 has defended a candidate thesis on the subject "High accuracy logarithmic analog-to-digital converters" and received PhD degree. In 2002 has got academic status of associate professor. In 2004 has got a position of chief of Information Support Center of National University "Lviv Polytechnic". Author of 28 scientific articles.
Areas of interests: computer network security, firewalls, intrusion detection systems, scanners, vulnerability assessment tools, penetration testing, computer network and system administration, technical information security