Protection of Communication Infrastructures

Chapter 3 Denial of Service

- ❑ Introduction
- ❑ DoS Categories and Examples
- ❑ Countermeasures
- ❑ Tracing back the source of an attack

The Threat...

(source: Julie Sigwart -- the creator of the popular comic "Geeks")

Introduction

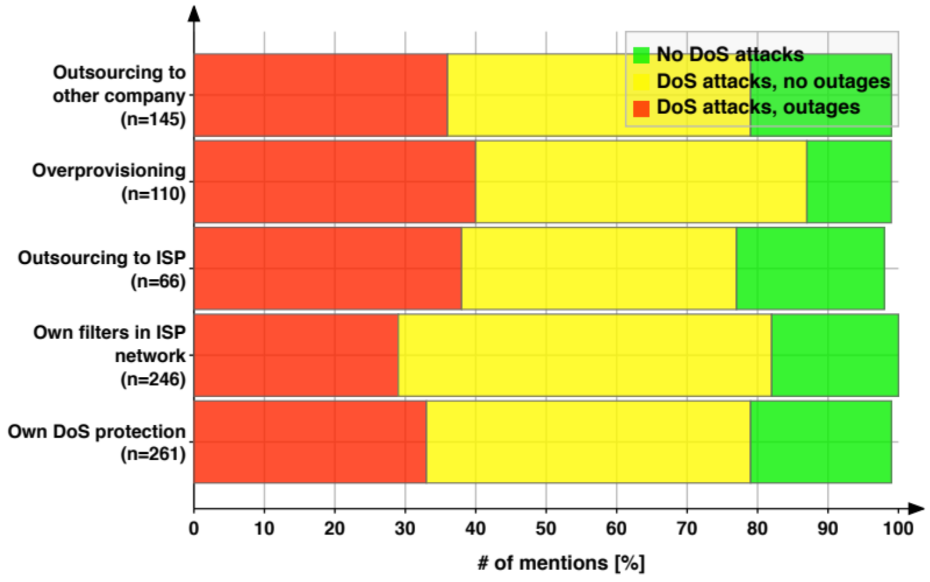
- What is Denial of Service?
  - Denial of Service (DoS) attacks aim at denying or degrading legitimate users' access to a service or network resource, or at bringing down the servers offering such services
- Motivations for launching DoS attacks:
  - Hacking (just for fun, by "script kiddies", ...)
  - Gaining information leap (→ 1997 attack on bureau of labor statistics server; was possibly launched as unemployment information has implications to the stock market)
  - Discrediting an organization operating a system (i.e. web server)
  - Revenge (personal, against a company, ...)
  - Political reasons ("information warfare")
  - ...

How serious is the DoS problem?

- Qualitative answer:
  - Very, as our modern information society depends increasingly on availability of information and communications services
  - Even worse, as attacking tools are available for download
- Quantitative answer:
  - In a CSI/FBI survey [CSI00] 27% of security professionals responded that they detected DoS attacks in the year 2000
  - Another study supervised the link to a class-A subnetwork (~ 1/256 of the Internet address space) for packets like TCP-SynAck, etc. that come spontaneously and thus represent most probably a reply to a "spoofed" attacking packet; during three weeks a total of 200 million suspicious packets were observed (for analysis see [MVS01])

How serious is the DoS problem?

- Another quantitative answer:



Survey among 400 IT executives on DoS attacks [For09]:

Denial of Service Attacking Techniques

- Permanent consequences
- *Resource destruction* by:
  - Hacking into systems
  - Making use of implementation weaknesses as buffer overrun
  - Deviation from proper protocol execution
- *Resource reservations* that are never used (e.g. bandwidth)
  - E.g. TCP connections with window 0
- *Resource depletion* by causing:
  - Storage of (useless) state information
  - High traffic load (requires high overall bandwidth from attacker)
  - Expensive computations (“expensive cryptography”!)
- Origin of malicious traffic:
  - Single source with single / multiple (forged) source addresses
  - Multiple sources with forged / valid source addresses (Distributed DoS)

Examples: Resource Destruction (I)

- *Hacking:*
  - Exploiting weaknesses that are caused by careless operation of a system
  - Examples: default accounts and passwords not disabled, badly chosen passwords, social engineering (incl. email worms), etc.
- *Making use of implementation weaknesses:*
  - See chapter 2 on security aware system design & implementation

Examples: Resource Destruction (II)

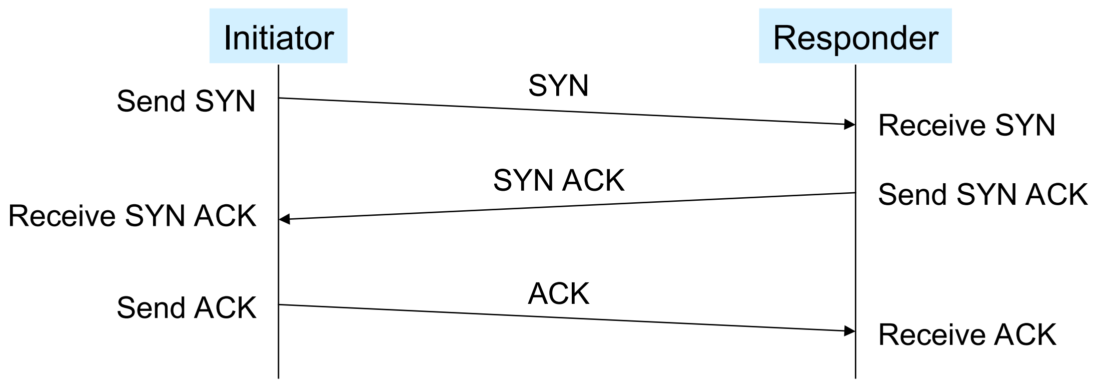
- *Deviation from proper protocol execution:*
- Well-known examples:
- Ping-of-Death
  - Attacker sends IP fragments that exceed the total size of 65,535 bytes
  - After reassembly a buffer overflow occurs…
- Teardrop attack
  - IP fragments may overlap & even be contained in each other (in theory)
  - Attacker send a fragment that is fully contained in another
  - “Length” of fragment part to copy to packet buffer becomes negative
  - If unsigned variables are used, values become LARGE
  - OS memory is being overwritten
- LAND attack
  - TCP spoofing is used to send SYN packet
  - Source & destination address equal
  - OS may run in an infinite loop

Examples: Resource Depletion (I)

- Expensive *computations* (“expensive cryptography”!)
  - Often on “higher” layers
  - On L3/L4: Parallel negotiation of many cryptographic connections
  - Typical example: THC SSL DoS tool (performs permanent renegotiations)
- *Storage* of (useless) state information
  - IP fragment attack
    - Attacker sends IP fragments that never form a complete packet
    - Receiver must store fragments until timeout
  - TCP SYN Flooding (details follow)
- *High traffic load* (requires high bandwidth or amplification)
  - Examples for amplification techniques:
    - Smurf attack
    - TCP bang attack
    - DNS & NTP amplification
    - Bouncing attacks
  - Remember: TCP stacks will throttle, when load becomes too high…

Background: TCP’s Three-Way-Handshake

- The *Transmission Control Protocol (TCP):*
  - provides a connection-oriented, reliable transport service
  - uses IP for transport of its PDUs
- TCP connection establishment is realized with the following dialogue:



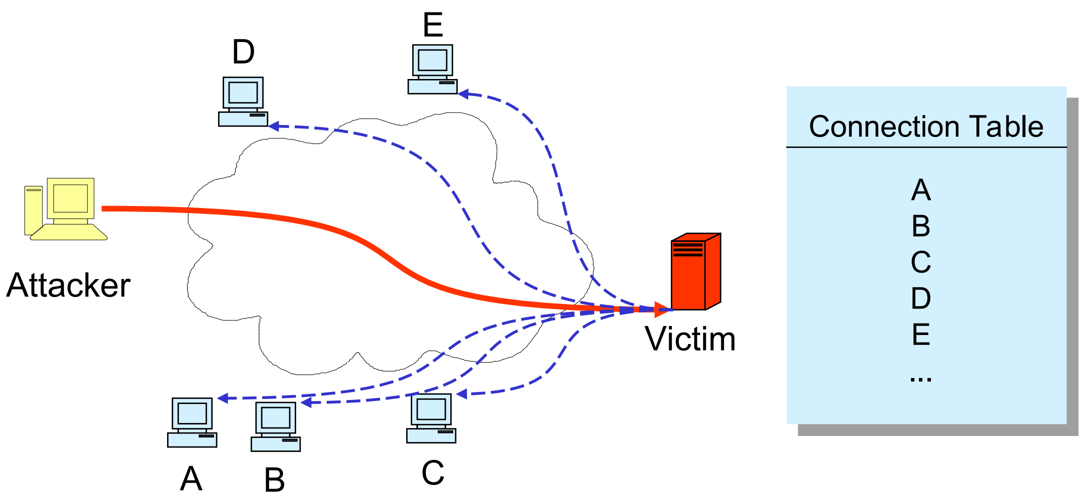
- After this dialogue, data can be exchanged in both directions
- Both peers may initiate termination of the connection (with a two-way-handshake)

Background: Reaction According to Protocol Specification

Reply Packets According to Protocol Specification if State not Available

Packet Send	Reaction of Receiver
TCP SYN (to open port)	TCP SYN ACK
TCP SYN (to closed port)	TCP RST (ACK)
TCP ACK	TCP RST (ACK)
TCP DATA	TCP RST (ACK)
TCP RST	no response
TCP NULL	TCP RST (ACK)
ICMP Echo Request	ICMP Echo Reply
ICMP TS Request	ICMP TS Reply
UDP Packet (to open port)	protocol dependent
UDP Packet (to closed port)	ICMP Port Unreachable

Examples: TCP-SYN flood attack



→ TCP SYN packets with forged source addresses (“SYN Flood”)

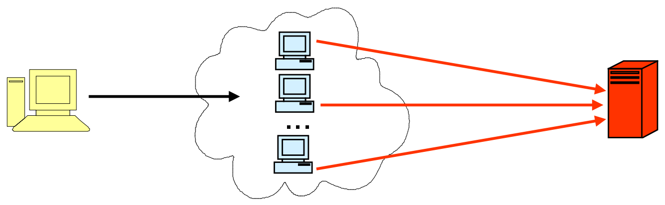
⇢ TCP SYN ACK packet to assumed initiator (“Backscatter”)

Background: Internet Control Message Protocol

- The Internet Control Message Protocol (ICMP) has been specified for communication of error conditions in the Internet
- ICMP PDUs are transported as IP packet payload and identified by value “1” in the protocol field of the IP header
- Some ICMP functions:
  - *Announce network errors:* e.g. a host or entire portion of the network being unreachable, or a TCP or UDP packet directed at a port number with no receiver attached (destination unreachable)
  - *Announce network congestion:* routers generate ICMP source quench messages, when they need to buffer too many packets
  - *Assist troubleshooting:* ICMP supports an Echo function, which just sends an ICMP echo packet on a roundtrip between two hosts
  - *Announce timeouts:* if an IP packet's TTL field drops to zero, the router discarding the packet may generate an ICMP packet (time exceeded)
  - *Announce routing detours:* if a router detects that it is not on the route between source and destination, it may generate an ICMP redirect packet

Example: Abusing ICMP for Malicious Activities

- Two main reasons make ICMP particular interesting for attackers:
  - It may be addressed to broadcast addresses
  - Routers respond to it
- The *Smurf* attack - ICMP echo request to broadcast:
  - An attacker sends an ICMP echo request to a broadcast address with the source addressed forged to refer to the victim
  - Routers (often) allow ICMP echo requests to broadcast addresses
  - All devices in the addressed network respond to the packet
  - The victim is flooded with replies to the echo request
  - With this technique, the network being abused as an (unaware) attack amplifier is also called a reflector network.



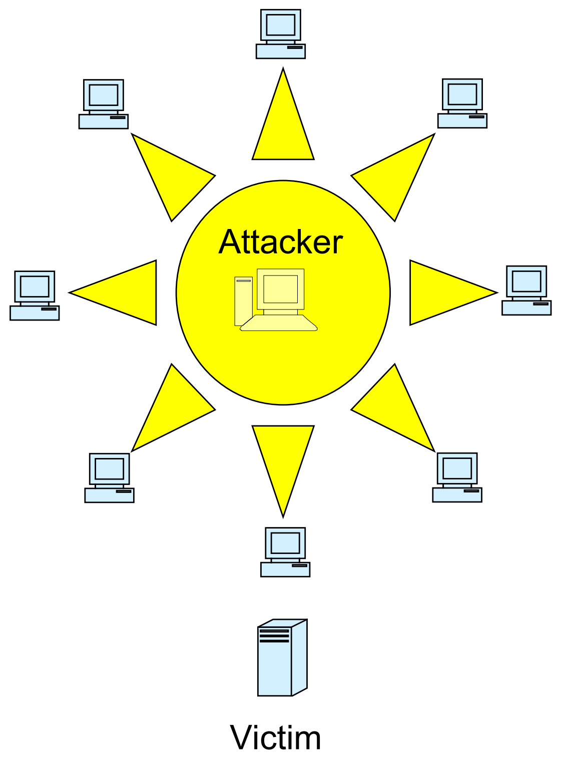
Example: TCP bang attack, DNS & NTP amplification

- TCP bang attack:
  - Smurf attack amplifies over space
  - Idea: amplify over time!
  - Attacker forges IP source address in TCP SYN packets
  - SYN-ACK packets from reflectors hit victim
  - If victim cannot respond with TCP-RST (due to overload, firewall etc), reflectors retransmit SYN-ACKs
- DNS & NTP amplification
  - Connection-less UDP-based protocols
  - Both: Simple request/reply scheme
  - Replies may be much larger than requests
  - Amplification by sending packets from forged source address

Example: Bounce attacks

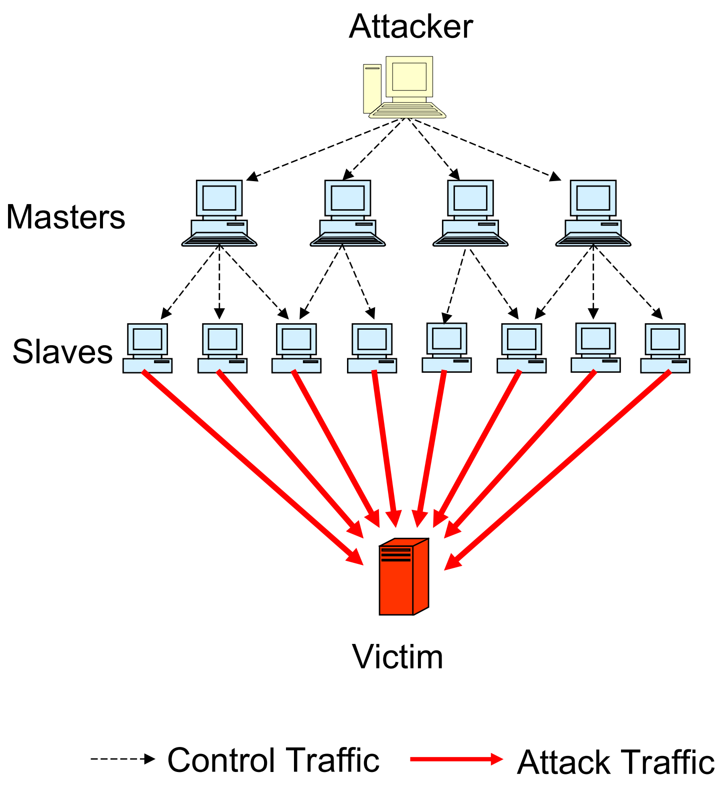
- DoS attacks so far do not require the victim to interact
- Sometimes the victim “cooperates” in amplification by bouncing packets itself
- Examples:
  - Misconfigured SMTP servers that reply to e-mail bounces with bounces
    - Attacker only needs to send a mail from a non-existing account to a different non-existing account
  - Mailing lists that are subscribed to each other (and do not filter properly)
  - UDP echo servers that answer to other echo servers
  - …

Resource Depletion with Distributed DoS (1)



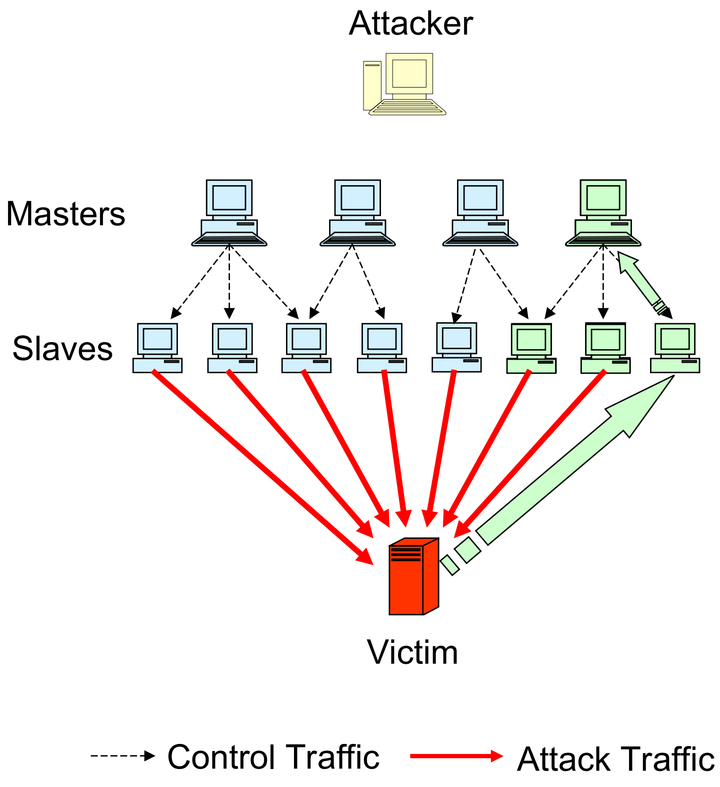
- Attacker intrudes multiple systems by exploiting known flaws
- Attacker installs DoS-software:
  - „Root Kits“ are used to hide the existence of this software
- DoS-software is used for:
  - Exchange of control commands
  - Launching an attack
  - Coordinating the attack

Resource Depletion with Distributed DoS (2)



- The attacker classifies the compromised systems in:
  - Master systems
  - Slave systems
- Master systems:
  - Receive command data from attacker
  - Control the slaves
- Slave systems:
  - Launch the proper attack against the victim
- During the attack there is no traffic from the attacker

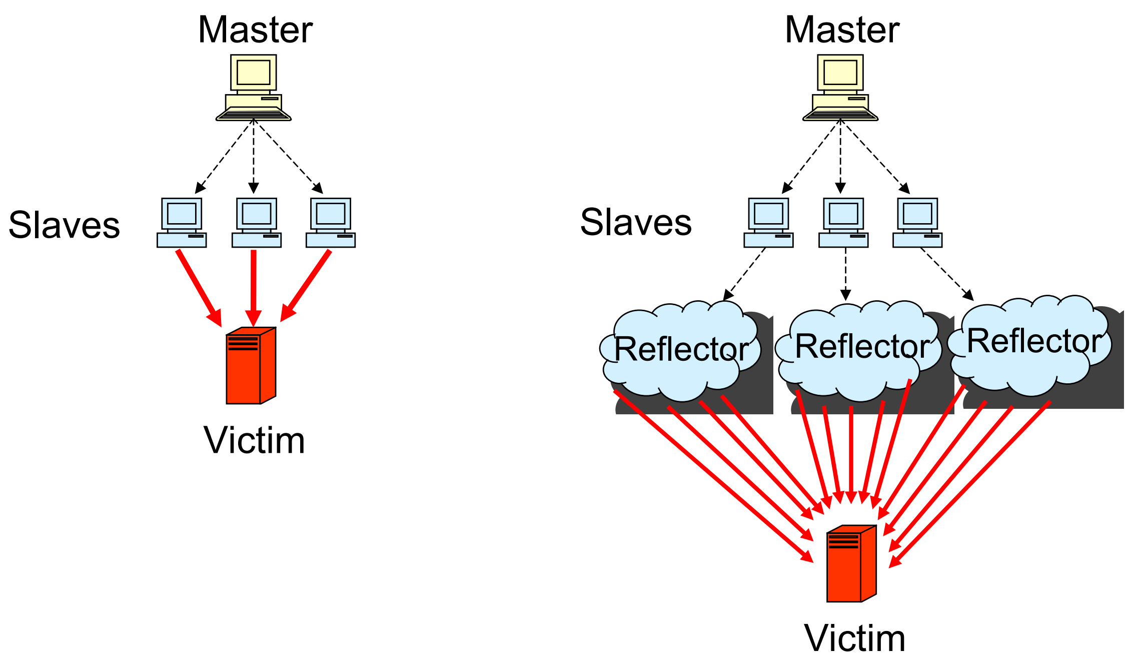
Resource Depletion with Distributed DoS (3)



- Each master system only knows some slave systems
- Therefore, the network can handle partial failure, caused by detection of some slaves or masters

Resource Depletion with Distributed DoS (4)

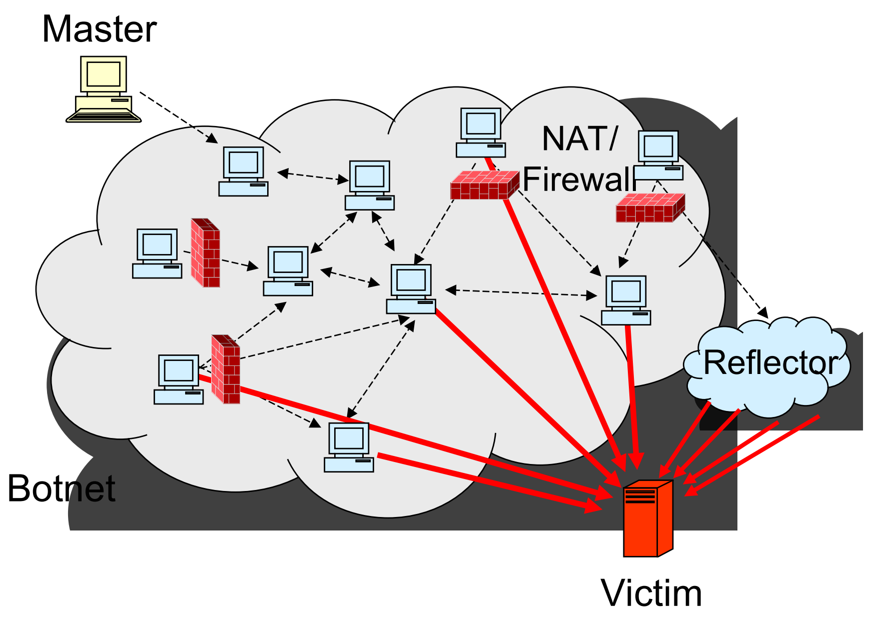
Different Attack Network Topologies



a.) Master-Slave-Victim b.) Master-Slave-Reflector-Victim

Resource Depletion with Distributed DoS (5)

Different Attack Network Topologies



c.) Peer-to-Peer-based Botnet (encrypted communication)

Defense Techniques Against DoS Attacks (1)

- Defenses against disabling services:
  - Hacking:
    - Good system administration
    - Firewalls, logging & intrusion detection systems
  - Implementation weakness:
    - Code reviews, stress testing, etc.
    - Software updates
  - Protocol deviation:
    - Fault tolerant protocol design
    - Error logging & intrusion detection systems
    - “DoS-aware protocol design”, e.g. be aware of possible DoS attacks when reassembling packets

Defense Techniques Against DoS Attacks (2)

- Defenses against resource depletion:
  - Generally:
    - Rate Control (ensures availability of other functions on same system)
    - Authentication & Accounting
  - Do not perform expensive operations, reserve memory, etc., before authentication
  - Expensive computations: careful protocol design, verifying the initiator's "willingness" to spend resources himself (e.g. "client puzzles" [JuBr99], details follow)
  - Memory exhaustion: stateless protocol operation (details follow)

Defense Techniques Against DoS Attacks (3)

- Concerning origin of malicious traffic:
  - Defenses against single source attacks:
    - Disabling of address ranges (helps if addresses are valid)
    - Might also be misused by forged addresses…
  - Defenses against forged source addresses:
    - Ingress Filtering at ISPs (if the world was an ideal one...)
    - "Verify" source of traffic (e.g. with exchange of "cookies" [TL00])
    - Tracing back the true source of packets with spoofed addresses
  - Widely distributed DoS:
    - Anycast infrastructure, like in DNS
    - Distributed data centers & content delivery networks
    - ISP filters with advanced methods to distinguish between bot and honest client (e.g. by verifying JavaScript is correctly executed etc.)
    - For individuals & smaller companies or intelligent attackers: ???

Background on Authentication (1)

- <u>Definition:</u>

  A *cryptographic protocol* is defined as a series of steps and message exchanges between multiple entities in order to achieve a specific security objective
- Properties of a protocol (in general):
  - Everyone involved in the protocol must know the protocol and all of the steps to follow in advance
  - Everyone involved in the protocol must agree to follow it
  - The protocol must be unambiguous, that is every step is well defined and there is no chance of misunderstanding
  - The protocol must be complete, i.e. there is a specified action for every possible situation
- Additional property of a cryptographic protocol:
  - It should not be possible to do or learn more than what is specified in the protocol

Background on Authentication (2)

- Basic variants of authentication:
  - *Data origin authentication* is the security service that enables entities to verify that a message has been originated by a particular entity and that it has not been altered afterwards (synonym for this service: *data integrity*)
  - *Entity authentication* is the security service, that enables communication partners to verify the identity of their peer entities
- In general, entity authentication can be achieved with:
  - *Knowledge:* e.g. passwords
  - *Possession:* e.g. physical keys or cards
  - *Immutable characteristic:* e.g. biometric properties like fingerprint, etc.
  - *Location:* evidence is presented that an entity is at a specific place (example: people check rarely the authenticity of agents in a bank)
  - *Delegation of authenticity:* the verifying entity accepts, that somebody who is trusted has already established authentication
  - In communication networks, direct verification of the above means is difficult or insecure which motivates the need for cryptographic protocols

Background on Authentication (3)

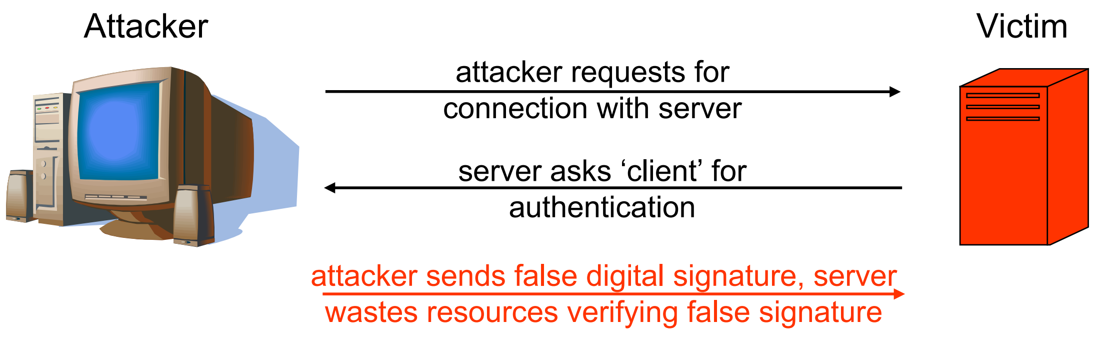
- The main reason, why entity authentication is more than an exchange of (data-origin-) authentic messages is *timeliness:*
  - Even if Bob receives authentic messages from Alice during a communication, he can not be sure, if:
    - Alice is actually participating in the communication *in this specific moment,* or if
    - Eve is *replaying* old messages from Alice
  - This is of specific significance, when authentication is only performed at connection-setup time:
    - Example: transmission of a (possibly encrypted) PIN when logging in
  - Two principle means to ensure timeliness in cryptographic protocols:
    - *Timestamps* (require more or less synchronized clocks)
    - *Random numbers* (challenge-response exchanges)
- Most authentication protocols do also establish a secret session key for securing the session following the authentication exchange

Background on Authentication (3)

- Two main categories of protocols for entity authentication:
  - *Arbitrated authentication:* an arbiter, also called *trusted third party (TTP)* is directly involved in every authentication exchange
    - Advantages:
      - This allows two parties A and B to authenticate to each other without knowing any pre-established secret
      - Even if A and B do not know each other, symmetric cryptography can be used
    - Drawbacks:
      - The TTP can become a bottleneck, availability of TTP is critical
      - The TTP can monitor all authentication activity
  - *Direct authentication:* A and B directly authenticate to each other
    - Advantages: no online participation of a third party is required and no possible performance bottleneck is introduced
    - Drawbacks: requires asymmetric cryptography or pre-established secret keys

DoS Examples: Resource Depletion with CPU Exhaustion

- Category *CPU exhaustion by expensive computations:*
  - Here: attacking with bogus authentication attempts

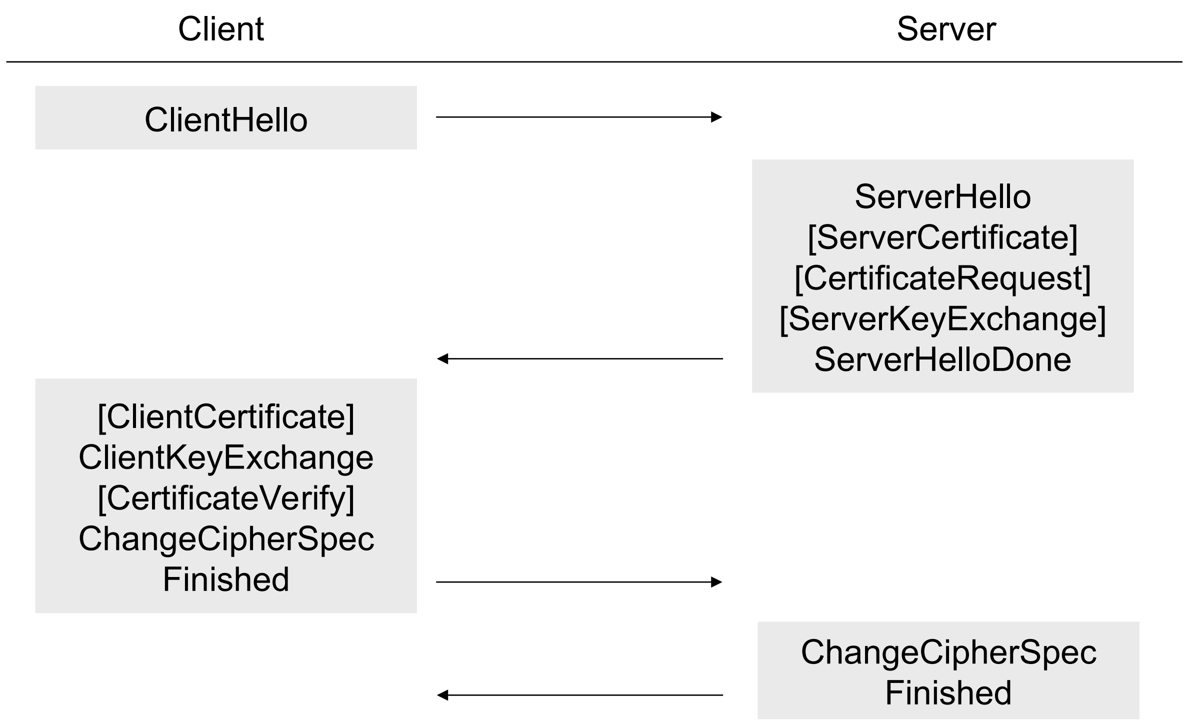

Attacker
Victim
attacker requests for connection with server
server asks 'client' for authentication
attacker sends false digital signature, server wastes resources verifying false signature

  - The attacker usually either needs to receive or guess some values of the second message, that have to be included in the third message for the attack to be successful
  - Also, the attacker, must trick the victim *repeatedly* to perform the expensive computation in order to cause significant damage

Background: Secure Socket Layer (SSL)

- SSL was designed in the early 1990's to primarily protect HTTP sessions and it provides the following security services:
  - *Peer entity authentication:*
    - Prior to any communications between a client and a server, an authentication protocol is run to authenticate the peer entities
    - Upon successful completion of the authentication dialogue an *SSL session* is established between the peer entities
  - *User data confidentiality:*
    - If negotiated upon session establishment, user data is encrypted
    - Different encryption algorithms can be negotiated: RC4, DES, 3DES, ...
  - *User data integrity:*
    - A MAC based on a cryptographic hash function is appended to user data
    - The MAC is computed with a negotiated secret in prefix-suffix mode
    - Either MD5 or SHA can be negotiated for MAC computation

SSL Authentication: Full Handshake



[...] denotes optional messages

SSL Handshake Protocol: Cryptographic Aspects (1)

- SSL supports three methods for establishing session keys:
  - *RSA:* a *pre-master-secret* is randomly generated by the client and sent to the server encrypted with the servers public key
  - *Diffie-Hellman:* a standard Diffie-Hellman exchange is performed and the established shared secret is taken as *pre-master-secret*
  - *Fortezza:* an unpublished security technology developed by the NSA
- As SSL was primarily designed to secure HTTP traffic, its “default application scenario” is a client wishing to access an authentic web-server:
  - In this case the web-server sends its public key certificate after the ServerHello message
  - The server certificate may contain the server’s public DH-values or the server may send them in the optional ServerKeyExchange message

→ Both methods, RSA and Diffie-Hellman enable an attacker to launch DoS attacks!

Countering CPU Exhaustion with Client Puzzles (1)

- Basic idea:
  - Upon a new request, a server generates a new task (“client puzzle”) that the client has to solve before it will be served
  - Client puzzles can be easily generated and verified by a server, while clients must use a significant amount of computational resources in order to solve them
  - Furthermore, the puzzles' difficulty can be easily scaled based on factors such as server load or server trust of the client
- Drawback:
  - Honest clients must also spend resources on solving client puzzles

Countering CPU Exhaustion with Client Puzzles (2)

- Example scheme:
  - The server generates two random numbers N_S and X' and computes a cryptographic hash value $h = H(N_S, X')$ of them
  - The server then provides the client with one of the random numbers N_S and k bits (for example 8 bit) of the hash value
  - The client must then guess random numbers and perform compute cryptographic hash values until k bit of a resulting hash value match the value that has been supplied by the server
  - As cryptographic hash functions can not be inverted, the client on the average has to try 2^{k-1} different random numbers until he finds one number X so that 8 bit of $H(N_S, X)$ match the value provided by the server
  - However, in order to generate and check the client puzzle, the server just needs to compute the cryptographic hash function two times
  - This effort on the server side can be further reduced by just generating and sending one random number N_S and the parameter k to the client and always requiring the first k bit of $H(N_S, X)$ to be of a fixed value, e.g. 0

Countering CPU Exhaustion with Client Puzzles (3)

- Basic properties of a client puzzle as required by Aura et. al.:
  - the creation and verification of a puzzle is inexpensive to a server,
  - the server can adjust the cost of solving a puzzle (from zero to impossible),
  - the puzzle can be solved on most type of client hardware,
  - the pre-computation of solutions is impossible,
  - the server does not need to store any client-specific data while client solves the puzzle,
  - the same puzzle may be given to several clients, while ensuring that knowing the solution of one or more clients does not help a new client in solving the puzzle,
  - a client can reuse a puzzle by creating several instances of it, however, the solution to a puzzle should not be reusable

Countering CPU Exhaustion with Client Puzzles (4)

- Reusable client puzzles according to Aura et. al:
  - Server periodically broadcasts random number N_S and difficulty level k
  - Every client can then create a solution to a new instance of this puzzle by:
    - Generating a fresh random number N_C
    - Determining with brute force search (= trying all possible values) an X such that:

$$H(C, N_S, N_C, X) \overset{!}{=} \underbrace{00000}_{k}Y$$

- Summary:
  - Client puzzles provide an effective means to slow down potential DoS attackers significantly
  - At the same time, the length of messages is only increased minimally (about one byte for parameter k and up to eight bytes for the solution X)
  - This may protect servers at the early stage of a normal authentication where the computations are the most CPU intensive

Integrating a Client Puzzle into an Authentication Protocol

Client		Server
	$S_S(T_S, k, N_S)$ $\longleftarrow$	S periodically decides k, generates N_s , and timestamp and signs the following message.
C verifies the timestamp T_S and signature S_S. C generates N_C and guesses a solution X so that: $h(C, N_S, N_C, X) = 0_1 0_2 \ldots 0_K Y$. C signs the following message	$S_C(S, C, N_S, N_C, X)$ $\longrightarrow$	
		S verifies that N_S is recent, C, N_S, N_C have not been used before, and $h(C, N_S, N_C, X) = 0_1 0_2 \ldots 0_K Y$. **S may now commit resources** . S stores C, N_S, N_C while N_S recent and verifies the signature S_C. **S has now authenticated C.** S signs the following message.
C verifies the signature S_S. **C has now authenticated S.**	$S_S(S, C, N_C)$ $\longleftarrow$	

Countering Memory Exhaustion: Stateless Protocol Design

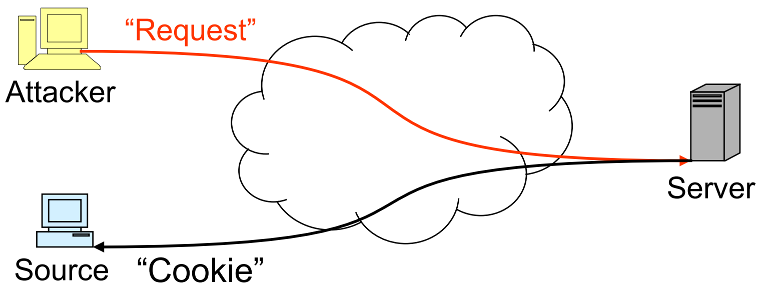
- Basic idea:
  - Avoid storing information at server, before DoS attack can be ruled out
  - So, as long as no assurance regarding the client has been reached all state is "stored" in the network (transferred back and forth)

Stateful Operation		Stateless Operation
1. C → S: Msg_1		1. C → S: Msg_1
2. S → C: Msg_2	S stores $State_{S1}$	2. S → C: Msg_2 , $State_{S1}$
3. C → S: Msg_3		3. C → S: Msg_3 , $State_{S1}$
4. S → C: Msg_4	S stores $State_{S2}$	4. S → C: Msg_4 , $State_{S2}$
...		...

- Attention: Integrity of the state needs to be checked (via a MAC)
- Drawback: requires higher bandwidth and more message processing

Verifying the Source of a Request

- Basic idea:
  - Before working on a new request, verify if the “initiator” can receive messages send to the claimed source of the request



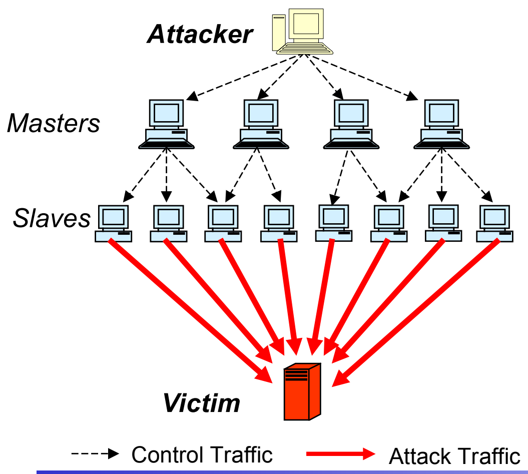
  - Only a legitimate client or an attacker which can receive the “cookie”, can send the cookie back to the server
  - Of course, an attacker must not be able to guess the content of a cookie
- Discussion:
  - Advantage: allows to counter simple spoofing attacks
  - Drawback: requires one additional message roundtrip

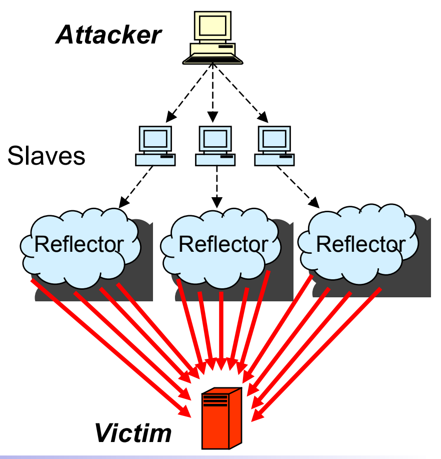
But...

- Verifying the source of a request with a cookie exchange can avoid spending significant computation or memory resources on a bogus request
- What if the attacker is only interested in exhausting the access or packet processing bandwidth of a victim?
  - Obviously, sending cookies to all incoming packets even aggravates the situation!
  - Such an attack situation, however, is quite easy to detect: there are simply too many packets coming in
- Problems in such a case:
  - Which packets come from genuine sources and which are bogus ones?
  - Even worse: source addresses given in the packets may be spoofed
  - Where do the spoofed packets come from?

IP-Address Spoofing

- Reprise: DoS-/ DDoS-Attacks
  - Direct Attacks (Master – network of slaves)
    Problem of spoofed source addresses of attack packets sent by the slaves
  - Reflector Attacks (Master – (slaves –) reflecting nodes)
    Problem of address-spoofing: set victims' IP-address as source
- Main problem is the possibility to lie about the source address…





Possible Solutions to DDoS-Attacks

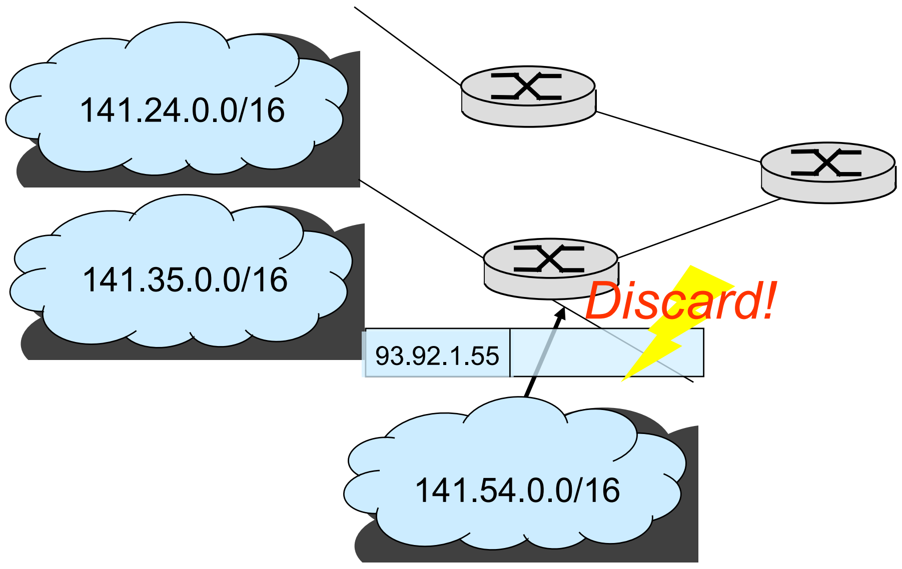
- Solutions to reflector attacks: secure available services
  - Balance effort of request and reply (no amplification through reflectors)
  - e.g.: Prohibit *ICMP-Echo-Request* to broadcast addresses
  - …
- Possible solutions to direct attacks:
  - Avoid IP-address spoofing
  - Live with spoofed addresses and restrain effect of attacks
    1. Locate source of attack-packets
    2. Filter traffic from attacking nodes
    3. Inform admin/root of attacking networks/nodes

    But: IP is connectionless! Necessary to find means to trace back the traffic to the original source / attacking node!

  Identify: zombie, spoofed address, ingress router, routers on path…

Inhibiting Spoofed Addresses: Ingress Filtering (RFC 2827)

- Routers block arriving packets with illegitimate source addresses.



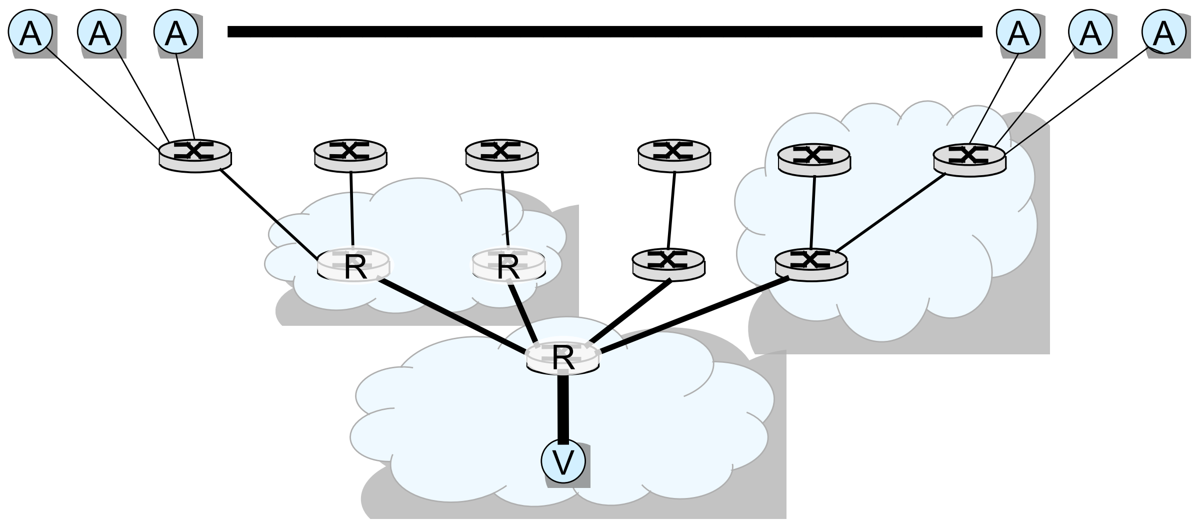
Ingress Filtering (2)

- (Almost) impossible in the backbone
- Only possible at access links → ISPs

- Problems occur:
  1. Issues with Mobile IP (users want to spoof to avoid “unnecessary” tunneling of outgoing traffic via home network!)
  2. Larger management overhead at router-level
  3. Potentially big processing overhead at access routers (e.g. big ISP running a large AS with numerous IP ranges and DHCP)
  4. Universal deployment needed

- And: ISPs do not really have an incentive in blocking any traffic…

Identifying Malicious Nodes: DDoS Attack Tree



- Rooted Tree with
  - Victim *(V)* (root of the tree)
  - Routers *(R)*
  - Attackers *(A_i)*
- Questions with forged IP addresses:
  - Where are malicious nodes?
  - Which router (ISP) is on attack path?

Identifying Malicious Nodes: Assumptions

- Attackers may generate any packet
- Multiple attackers can act in collusion
- Attackers are aware of tracing
- Packets are subject to reordering and loss
- Multitude of attacking packets (Usually many)
- Routes between A and V are stable (in the order of seconds)
- Resources at routers are limited
- Routers are usually not compromised

Identifying Malicious Nodes: Proposed Solutions

- Simple Classification of solutions:
  - Network Logging
    - Log Information on processed packets and path
  - Attack Path Traceback
    - Trace attack path through network
  - Other / Related
    - Attack Mitigation/Avoidance

Requirements / Evaluation Metrics

- Involvement of ISP (required or not)
- Amount of necessary packets to trace attack
- Effect of partial deployment
- Resource overhead
  - Processing overhead at routers
  - Memory requirements
  - Bandwidth overhead
- Ease of Evasion
- Protection
- Scalability
- Performance in case of Distributed DoS
- Performance in case of packet transformations

Involvement of ISP

- ISPs do not really have an incentive in preventing „attack traffic“:
  - Paid by number of transmitted bytes
  - Which traffic is „malicious“ and which is not?
  - „Malicious“ for whom?

- Infrastructure is expensive
- Management-/ down times are expensive
- Administrators are expensive

Amount of Packets Needed to Track Source

- Different types of attacks:
  - Bandwidth resource exhaustion
  - Continuous stream of packets for the time span of the attack
  - Packet flood to bring link/host down
  - One attacker / multiple attackers (multiple attack paths)

  - Well targeted packets (resource destruction, e.g. Teardrop attack)

- Which attacker can be traced?

Effect of Partial Deployment

- What if only a few ISPs deploy the mechanism (at first)?
- Still *some* benefit?
  - Attackers in the deploying ISPs traceable?
  - Ingress of attack packets traceable?
  - Cooperation of „islands“ possible – gain in knowledge if two ISPs deploy mechanism which are connected through a third transit domain?

Resource Overhead

- Resources in the network are scarce (memory, processing)!
- How much processing overhead is implied for the routers
  - Additional packet analysis
  - Additional functions
- How much information has to be stored at routers / in the network
  - Log of all processed packets?
- If mechanism needs communication:
  - In band / out of band?
  - How much extra bandwidth is needed to distribute information?

Scalability, Ease of Evasion & Graceful Degradation

- Scalability:
  - Does the mechanism scale with growing network sizes?
  - How much extra configuration is needed (only at new, or at all devices?)
  - How much do the elements depend on each other?
- Ease of Evasion:
  - How easy is it for an attacker to evade the mechanism?
  - Can the attacker send special packets which mislead the mechanism?
    - To stay transparent
    - To put an investigator off the scent
    - Attack the mechanism itself
- Graceful Degradation:
  - What if an attacker subverts one or many network elements on the path: Can the mechanism still produce meaningful results?

Performance: Towards DDoS and Packet Transformation

- Ability to handle DDoS:
  - Can the mechanism produce meaningful results, if a victim is attacked on different paths?
- Ability to handle packet transformation:
  - Does the mechanism produce meaningful results (results at all) if the packets are transformed due to:
    - Network Address Translation (NAT)
    - Packet Fragmentation
    - Packet Duplication (Multicast)
    - Tunneling

Identifying Malicious Nodes: Proposed Solutions

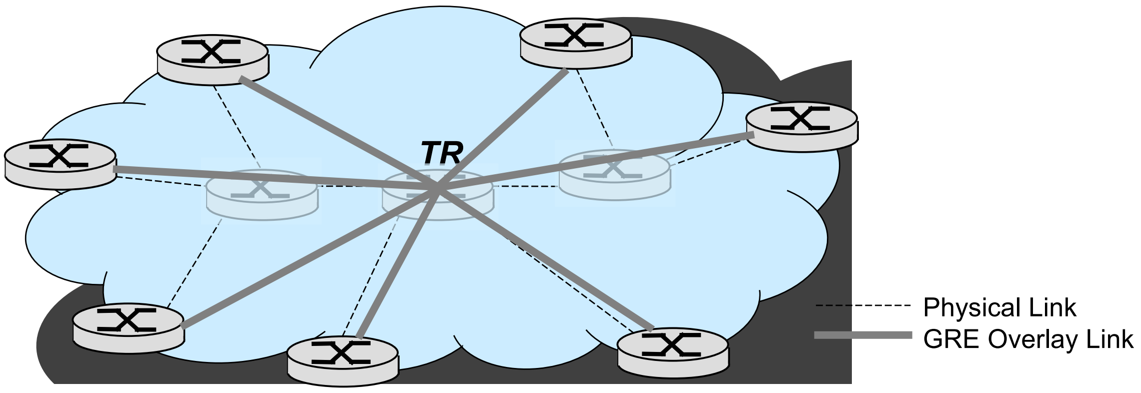
- Network Logging
  - Local network logging
  - Aggregated network logging
  - Source Path Identification („Hash-based IP-Traceback")
- Attack Path Traceback
  - Input Debugging
  - Controlled Flooding
  - ICMP Traceback
  - Probabilistic Packet Marking („IP-Traceback")
- Other / Related
  - Hop-Count Filtering
  - Aggregate Based Congestion Control (ACC)
  - Secure Overlay Services

Logging Approaches

- Log information on processed packets and path
- Network logging
  - Local network logging:
    - All routers log all traffic
    - *Too much overhead!*
    - *Does **not** scale*
  - Aggregated network logging
  - Source Path Identification („Hash-based IP-Traceback")

Aggregated Network Logging

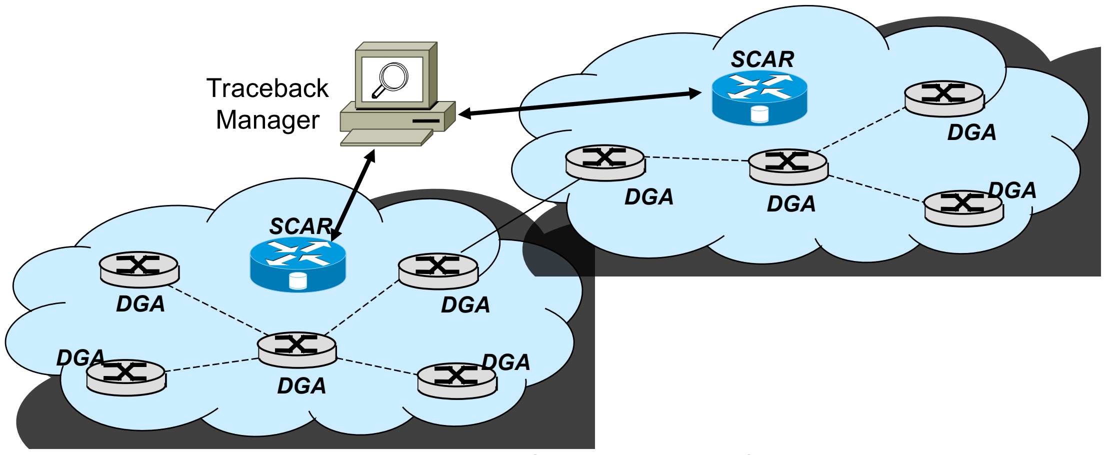
- Centralized Approach:
  - Introduction of „Tracking Router“ (TR)
  - Forwarding all traffic through TR (*Generic Routing Encapsulation*, GRE)
  - TR used to analyze “interesting” traffic and to identify edge router quickly
  - *Creates a single point of failure! Does not really scale!*



[Stone: „Centertrack: An IP Overlay Network for Tracking DoS Floods“]

Source Path Identification

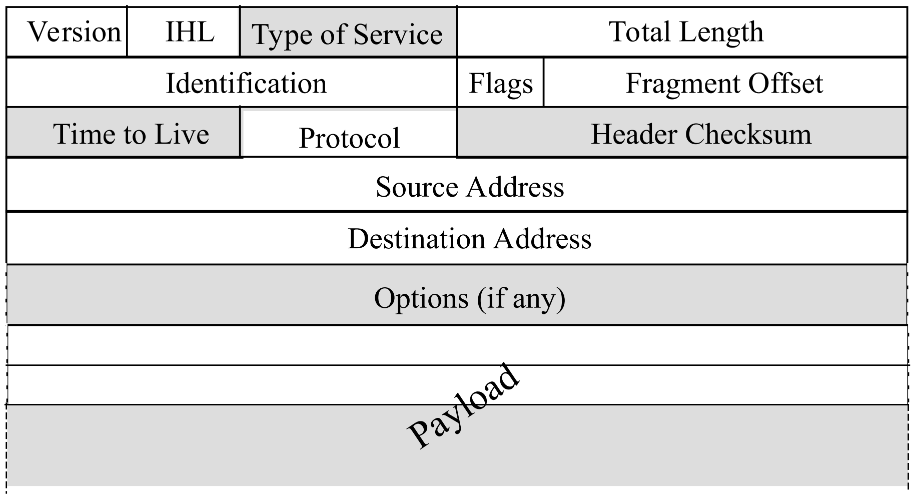
- Source Path Identification Engine *(SPIE, aka Hash-based IP Traceback)*
- Storage of compressed data in specialized devices
  - DGA generate digests of data *(Data Generation Agent)*
  - SCAR for storage and retrieval *(SPIE Collection & Reduction Agents)*
  - STM for central management *(SPIE Traceback Manager)*



[Snoeren et al.: „Single-Packet IP-Traceback“]

Source Path Identification (2)

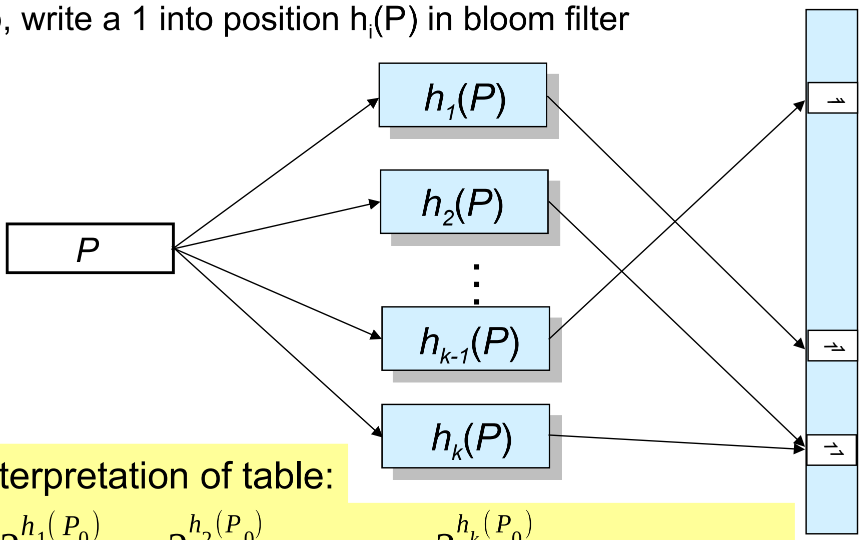
- *„Store all information on traversed packets?“*
- No! Store digests of:
  - Constant fields in IP Header (16 bytes)
  - First 8 bytes of Payload



- Hashed in so-called *Bloom Filters*

Source Path Identification: *Bloom Filters* (1)

- 24 bytes of each packet hashed with *k* hash functions h_i
- Hash values stored in filter:
  - To store p, write a 1 into position $h_i(P)$ in bloom filter



Numeric interpretation of table:

$$BF(P_0) = 2^{h_1(P_0)} \text{ or } 2^{h_2(P_0)} \text{ or } \ldots \text{ or } 2^{h_k(P_0)}$$

$$BF(P_n) = BF(P_{n-1}) \text{ or } 2^{h_1(P_n)} \text{ or } 2^{h_2(P_n)} \text{ or } \ldots \text{ or } 2^{h_k(P_n)}$$

Source Path Identification: *Bloom Filters* (2)

- Table size: hash function of length 32 bit leads to ½ GByte table size (2^{32} Bit = 2^{29} Byte)
- During normal operation DGAs maintain bloom filters, if bloom filter more than 70% “full” (70% of the bits are set to “1”), send it to SCAR
- Detection if a specific packet was processed:
  - Hash packet with *k* hash functions h_i
  - If any of the corresponding bits in all stored bloom filters is 0: Packet has *not* been processed
  - All bits of a bloom filter are 1: Packet *most probably* traversed the DGA
- Retrieval:
  - Victim contacts STM with pattern “P” of attack packet
  - STM distributes pattern “P” to SCARs
  - SCARs perform k hashes $h_1(P)$.. $h_k(P)$ to test which DGA forwarded matching packet
    - If there is one stored bloom filter with all bits at positions $2^{h_i(P)}$ set to one, then the respective DGA most probably has forwarded the packet

Traceback Approaches

- Trace attack path backwards through network
- Attack path traceback
  - Input Debugging
  - Controlled Flooding
  - ICMP Traceback
  - Probabilistic Packet Marking („IP-Traceback“)

Input Debugging

- During attack:
  - Trace attack-path „by hand“
  - Contact administrator / ISP
  - Admin matches ingress port for a given packet pattern of egress port
  - Repeat until source is found…
- Disadvantages:
  - Cumbersome (what if admin X is not available?)
  - Slow
  - Expensive (manual intervention)
  - Not scalable

Controlled Flooding

- During single source DoS-Attacks, traversed backbone links on the attack path are (heavily) loaded
- Traceback attack path by testing links:
  - Measure incoming attack traffic
  - From victim to approximate source:
    - Create load on suspect links in the backbone
    - Measure difference in incoming attack traffic: if less attack packets arrive, the link is on the attack path…
- Need for the possibility to create load on targeted links with access on multiple end-hosts around the backbone (available test-hosts use chargen-service on multitude of foreign end-hosts)
- ☹ DoS of the backbone in itself
- Almost impossible to test (high speed) backbone links using end-hosts (how many DSL-links do you need to saturate one CISCO-12000-Link (10Gbps)?

[Burch & Cheswick: „Tracing Anonymous Packets to Their Approximate Source“]

ICMP Traceback

- Routers give destination information about path of packets

- For 1 in 20k IP packets routers send additional ICMP *ITRACE* to destination
- Information in the *ITRACE* Packet:
  - TTL → 255 (number of hops between router and destination)
  - Timestamp
  - Address of router
  - Ingress (previous hop) and Egress ports (next hop on path)
  - Copy of payload of traced packet (for identification)

[Bellovin: „ICMP Traceback Messages“]

ICMP Traceback: Open Issues

- Signaling out of band → additional traffic (even with low probability)
- Large amount of packets needed to reconstruct the full attack path (Amount of ICMP packets vs. speed of path detection)
- Victim needs to analyze large amount of ITRACE messages
- Firewalls (often) drop ICMP messages
- Possibility to create fake ITRACE messages
  - Limited due to TTL value
  - Potential better solution:
    - Set up a PKI and let each router sign ITRACE messages
    - Use symmetric MACs and reveal key later on
  - But: Effort for creating and checking signatures???

Probabilistic Packet Marking (aka „IP Traceback“)

- ❑ Approach similar to ICMP Traceback:
- ❑ Mark forwarded packets with a very low probability
- ❑ In-band signaling to avoid additional bandwidth needs (mark packets directly)

- ❑ Different marking methods possible
- ❑ Different signaling (encoding) methods possible

[Savage et al.: „Network Support for IP Traceback“]

PPM Marking: Node Append

- ❑ Similar to IP Record Route: append each node‘s address to IP packet
- → Complete attack path in every received packet

```
Marking Procedure at router R:
   For each packet w, append R to w

Path Reconstruction Procedure at victim v:
  for any packet w from attacker
  extract path (R_1,..,R_j) from the suffix of w
```

- ❑ Converges quickly, easy to implement
- ❑ High bandwidth overhead (especially for small packets)
- ❑ Possible additional fragmentation of IP packets in every router

PPM Marking: Node Sampling (1)

- Similar to ICMP Traceback, but use additional IP header field

```
Marking Procedure at router R:
    For each packet w, with probability p write R into w.node

Path Reconstruction Procedure at victim v with
   additional node table NodeTbl (node, count):
   For each packet w from attacker z ← w.node
   if z in NodeTbl
        increment z.count
   else
        insert (z,1) in NodeTbl
   sort NodeTbl by count
   extract path ($R_1$,..,$R_j$) from ordered fields in NodeTbl
```

- Routers close to victim have higher probability of marking: the higher the count in NodeTbl the closer the router

PPM Marking: Node Sampling (2)

- Issues of node sampling:
  - Additional IP header field needed
  - Routers far away from victim contribute only few samples (marks are overwritten) and very large number of packets is needed to recover complete path *(p=0.51, d=15: > 42k attack packets needed to completely reconstruct attack path)*
  - In DDoS with multiple attackers different paths can not easily be distinguished

PPM Marking: Edge Sampling, Marking

- Mark packets with backbone edge e (u,w) (start router u, end router w) and distance (d(u,v))
- Victim v can deduct graph of edges e and reconstruct attack tree

```
Marking Procedure at router R:
   For each packet w, with probability p
     write R into w.start and 0 into w.distance
   else // probability 1-p
     if w.distance = 0 then
       write R into w.end
     increment w.distance
```

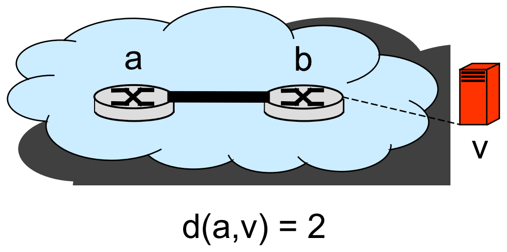
PPM Marking: Edge Sampling, Reconstruction

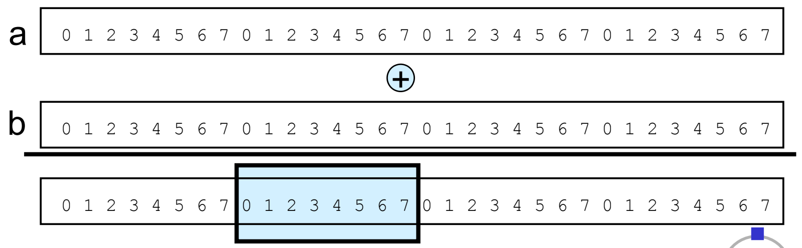
- In order to reconstruct the attack tree

```
Path Reconstruction Procedure at victim v with
  additional attack tree t:
  for each packet w from attacker
      if w.distance = 0 then
            insert edge (w.start, v, 0) into t
      else
            insert edge (w.start, w.end, w.distance) into t
  remove all edges (x,y,d) with d ≠ d(x,v) in t
  extract path (R_1,..,R_j) enumerating acyclic paths in t
```

PPM Encoding

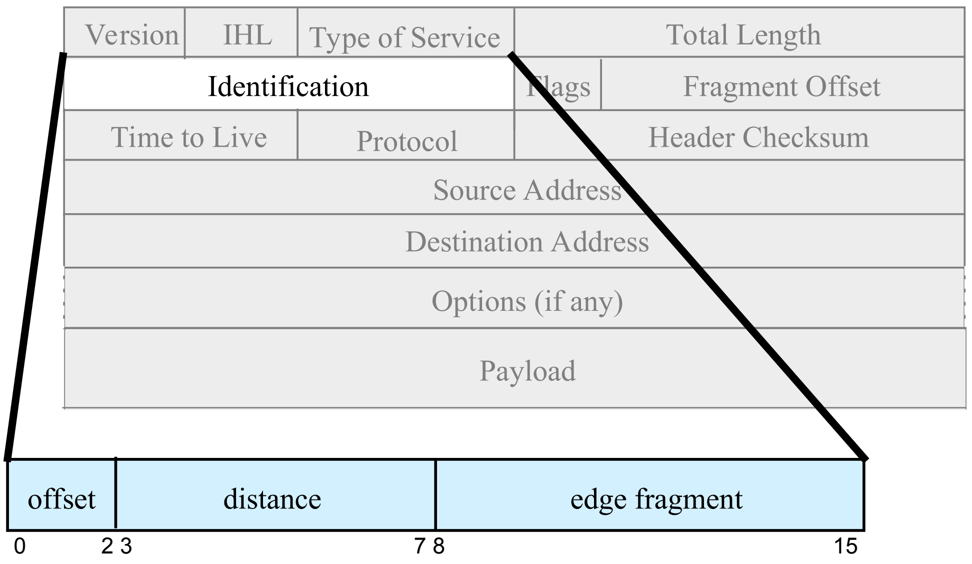
- With IP routers using IP addresses, marking of w.start, w.end, w.distance needs 32 + 32 + x bits.
- But: transmission of marks in IP header preferred!
- Solution: coding edge as IP(w.start) XOR IP(w.end)
  (last hop known (w.distance = 0), others determined through XOR at victim)
  → 32 bit („edge-id") + x bits (distance)
- Transmit only fragment of edge-ids with every packet and mark with higher probability *(actually, bit-interleaved with hashed values of the router's edge IP address to distinguish edges → 64 bit per edge)*
- Edge-ID-fragment 8 bits, offset 3 bits, distance 5 bits → 16 bits





PPM Encoding: Encapsulation in IP header

- Using the „Identification" field for in-band signaling (16 bit)



- But the ID-Field is needed!? In case of fragmentation:
  - Downstream marking: send ICMP Error („PMTU-D")
  - Upstream marking: set „don't fragment" flag

PPM Advantages and Disadvantages

- ☺ Stable
- ☺ Meaningful results under partial deployment
- ☺ No bandwidth overhead
- ☺ Low processing overhead

- ☹ Works mainly for bandwidth exhaustion attacks
  - ❑ Many packets needed for reconstructing attack path
  - ❑ Fragmented packets can not be traced (e.g. Teardrop attack, however, Teardrop is not bandwidth exhaustion anyway)
- ☹ Victim under attack needs rather high amount of memory (many packets!) and processing time
- ☹ In order to avoid spoofing, authentication needed (PKI, signatures)

Requirements Revisited

	ISP Involvement	Packet #	Partial Deployment	Overhead	Ease of Evasion	Protection	Scalability	DDoS	Packet-Transformation
Network-Logging	Large	1	No	High	Low	Fair	Poor	Good	Good
Source-Path Identif.	Fair	1	Yes	None (Memory:Fair)	Low	Fair	Fair	Good	Good
Input Debugging	High	Huge	No	None	Low	High	Low	Good	Good
Controlled Flooding	None	Huge	Not applicable	Huge	N/A	N/A	Low	Unable	Good
ICMP-Traceback	Low	Thousands	Yes	Low	High	High	High	Poor	Good
IP-Traceback (PPM)	Low	Thousands	Yes	Low	Low	High	High	Poor	Good

(According to A. Belenky, N. Ansari:"On IP Traceback“ [BA03])

Related Techniques for Mitigation / Avoidance

- Hop-Count Filtering
- Aggregate Based Congestion Control (ACC)
- Secure Overlay Services

Hop Count Filtering

- Can spoofed traffic be filtered based on contained data?

- Attacker can forge nearly any field in the IP header, but:
- TTL cannot be forged (is decremented by routers)
- Sanity check at ingress of ISP: does the distance to IP address of assumed sender leads to matching (sensible) TTL?
  - Needs to guess, what TTL is set by genuine system owning the IP address

- To avoid reflector attacks:
- Every node could perform sanity check before replying to assumed sender of packet

- But: Sender (attacker) can set initial TTL to any desired value…

[Jing, Wang & Shin: „Hop-Count Filtering: An Effective Defense Against Spoofed DDoS Traffic“]

Aggregate Based Congestion Control

- Is it possible, to restrain attack traffic in the backbone?
- Traffic is very diverse in the backbone, in general
- However, attack traffic forms an *aggregate* of similar traffic that can identified by:
  - Analyzing locally dropped traffic (due to full output queue),
  - Selecting the destination addresses with more than twice the mean number of drops, and
  - Clustering these destination addresses to 24bit prefixes
- ACC/pushback is a reactive approach:
  - If router/link is congested, can an aggregate be identified?
  - If there is an aggregate, limit the rate of aggregate traffic
  - If the aggregate persists, perform „*pushback*“: inform upstream routers to limit rate of the aggregate

[Mahajan, Bellovin & Floyd: „Controlling High Bandwidth Aggregates in the Network “]

Secure Overlay Services / Onion Routing

- Avoid attacks by hiding the service („*application hiding*“)

- Create hierarchy / Layers around servers (possible victims)
- Group nodes into the layers/hierarchy by degree of trust
- Forward all traffic through the hierarchy to the service
- Filter the traffic at each forwarding step

[Keromyits & Misra & Rubenstein: „SOS: Secure Overlay Services“]
[Reed, Syverson & Goldschlag: „Anonymous Connections and Onion Routing“]

Recapitulation: Source Identification of IP Traffic

- Problem: nodes may lie about their IP address
- Spoofing enables attackers to perform DoS/DDoS attacks

- If the source of an attack can be identified, attack traffic can be restrained

- Different approaches to identify attacker / routers / ISP on attack path:
  - Logging in the network
    - „Aggregated network logging“
    - Source Path Isolation („Hash-based IP Traceback“)
  - Traceback of packet flow
    - Controlled Flooding
    - ICMP Traceback
    - Probabilistic Packet Marking („IP Traceback“)
  - Other Means (Mitigation/Avoidance of attacks)

Some Upcoming Challenges

- The introduction of Internet protocols in classical and mobile telecommunication networks also introduces the Internet’s DoS vulnerabilities to these networks
- Programmable end-devices (smart phones, IoT devices, etc.) may constitute a large base of possible slave nodes for DDoS attacks on mobile networks
- Software defined radio implementation may even allow new attacking techniques:
  - Hacked smart phones answer to arbitrary paging requests
  - Unfair / malicious MAC protocol behavior
  - ...
- The ongoing integration of communications and automation may enable completely new DoS threats

Conclusion

- Increasing dependence of modern information society on availability of communication services
- While some DoS attacking techniques can be countered with “standard” methods, some can not:
  - Hacking, exploiting implementation weaknesses, etc. may be countered with firewalls, testing, monitoring etc.
  - Malicious protocol deviation & resource depletion is harder to defend against
- Designing DoS-resistant protocols emerges as a crucial task for network engineering:
  - Network protocol functions and architecture will have to be (re-)designed with the general risk of DoS in mind
  - Base techniques: stateless protocol design, cryptographic measures like authentication, cookies, client puzzles, etc.

Additional Reading (1)

[CSI00] Computer Security Institute and Federal Bureau of Investigation. *2000 CSI/FBI Computer Crime and Security Survey.* Computer Security Institute Publication, March 2000.

[Dar00] T. Darmohray, R. Oliver. *Hot Spares For DoS Attacks.* ;login:, 25(7), July 2000.

[JuBr99] A. Juels und J. Brainard. *Client Puzzles: A Cryptographic Countermeasure Against Connection Depletion Attacks.* In Proceedings of the 1999 Network and Distributed System Security Symposium (NDSS’99), Internet Society, March 1999.

[Mea00] C. Meadows. *A Cost-Based Framework for the Analysis of Denial of Service in Networks.* 2000.

[MVS01] D. Moore, G. M. Voelker, S. Savage. *Inferring Internet Denial-of-Service Activity.* University of California, San Diago, USA, 2001.

[NN01] S. Northcutt, J. Novak. *Network Intrusion Detection - An Analyst’s Handbook.* second edition, New Riders, 2001.

[TL00] P. Nikander, T. Aura, J. Leiwo. *Towards Network Denial of Service Resistant Protocols.* In Proceedings of the 15th International Information Security Conference (IFIP/SEC 2000) Beijing, China, 2000.

[BA03] A. Belenky, N. Ansari: "On IP Traceback", in IEEE Communications Magazine, July 2003

Additional Reading (2)

[BC00] Burch & Cheswick: „Tracing Anonymous Packets to Their Approximate Source“, Proceedings of the 14th USENIX conference on System administration, 2000

[Bel03] Bellovin, S.; Leech, M.; Taylor, T.: „ICMP Traceback Messages“, Internet-Draft http://tools.ietf.org/html/draft-ietf-itrace-04, 2003

[JWS03] Jing & Wang & Shin: „Hop-Count Filtering: An Effective Defense Against Spoofed DDoS Traffic“, Proceedings of the 10th ACM conference on Computer and communications security, 2003

[KMR02] Keromyits & Misra & Rubenstein: „SOS: Secure Overlay Services“, Proceedings of ACM SIGCOMM, 2002

[MBF01] Mahajan & Bellovin & Floyd: „Controlling High Bandwidth Aggregates in the Network“, Technical report, 2001

[RSG98] Reed, Syverson & Goldschlag: „Anonymous Connections and Onion Routing“, IEEE Journal on Selected Areas in Communications, 1998

[Sav01] Savage et al.: „Network Support for IP Traceback“, IEEE/ACM Transactions on Networking (TON), 2001

[Sto00] Stone: „Centertrack: An IP Overlay Network for Tracking DoS Floods“, Proceedings of 9th USENIX Security Symposium, 2000.

[Sno02] Snoeren et al.: „Single-Packet IP-Traceback“, IEEE/ACM Transactions on Networking (TON), 2002