Denial of Service Attacks and How to Defend Against Them

Valentin Razmov
(valentin@cs.washington.edu)
Computer Science and Engineering Department
University of Washington

May 10, 2000

Abstract

This paper is a survey on the problem of denial-of-service (DoS) attacks and proposed ways to deal with it. We describe the nature of the problem and look for its root causes, further presenting brief insights and suggested approaches for defending against DoS. We point out both the positive and negative sides of each potential solution. Future work identifies and justifies open research issues. In conclusion we give a brief summary of what has realistically been achieved so far, as well as what the key missing components still are.

I. Introduction

Security threats are often classified into one of three main categories: breaches of confidentiality, failure of authenticity and unauthorized denial of service [Need94]. While there have been extensive studies into the details of the former two, until recently denial of service had been disregarded from being a serious threat and had seldom been the focus of attention. Since the number of successive attacks on major commercial Internet sites in February 2000 [CERT00], this topic has been gaining popularity in the research, commercial and even political circles.

In this paper we will explore the roots of the problem and will try to identify which of a set of proposed solutions provide the most promising directions for short-term and long-term remedy, both as alternatives and as complements to each other.

A denial of service attack on a network could take one of three possible forms. A malicious party (a.k.a. the attacker) could cause the network not to transmit messages it should be sending in order to offer service to a subset or all of its clients. On the other end of the spectrum, the network could be caused to send messages, which it should not be sending. By far the most common form of DoS in today's networks is causing excessive bogus traffic (a.k.a. flooding the network) in the direction of a particular server, which in the end will prevent legitimate users from getting the service they could otherwise be receiving from that server.

A simple example of denial of service attack is the popular SYN attack on the TCP protocol. A client sends a request (SYN) to a server announcing its intention to start a conversation. The server responds with an acknowledgement (SYN ACK), accepting the establishment of a connection to the client and simultaneously reserving an entry for the pending connection in its connection queue. Now it is the client's turn to acknowledge the start of the communication by sending its (SYN ACK ACK) packet. A malicious client may never do that, as a result the server ends up with its connection queue entry tied up (and unused) for a significant amount of time (at least as long as the timeout), before it can be released. If one imagines the above scenario repeating over multiple (almost) simultaneous client requests, it is easy to see how the server could be tricked into initiating bogus "communication" with one of more malicious client(s). Since the maximum processing power of a typical 100 MIPS class server is on the order of 1000 to 2000 connections per second [SPEC96] and the minimum standard server TCP connection queue is 2048 slots [DEC96], it becomes clear that overwhelming even a powerful server is within the capabilities of even a very small number of conspiring malicious clients. (Here we should note that a very similar in spirit attack could be mounted against SSL, in the latter case the server's limited computational rather than memory resources can be most successfully exploited.)

The above patterns are far from contrived, being instead existing weaknesses. An even more credible testimony to the seriousness of the threats these can pose are the real attacks having occurred in the Internet, reported in advisories by the Computer Emergency Response Team (CERT) over the past few years. One study has shown that the number of such attacks has been steadily growing by 50% per year over the past decade [How98].

This all should serve as a motivation for tackling denial of service attacks the best we can without future delay.

In the next section we look into the details of the problem and search for the root causes for it. Section III is a digest of proposed approaches for eliminating, mitigating or following up on attacks. Finally, in section IV we briefly identify directions where future work is needed and conclude by summarizing what the state-of-the-art in the research on denial of service attacks is and what still remains on our wish list.

II. Deeper into the Problem

1. Culprits

One obvious problem in the TCP SYN attack scenario is that all the preliminary communication takes place *before authentication*, so the server can not tell a legitimate request from a fake one. There is nothing much that can be done about this, since trying to put authentication from the very start would be a denial of service attack in its own right, since the server would be preoccupied verifying (digital) signatures. Regardless of whether the signatures are fake or not, the very action of doing the computationally intensive verification would use up all spare computational resources of the server.

A less obvious cause of denial of service as a phenomenon lies in the *lack of accounting* for the resources consumed by each client [SP99]. Spatscheck and Peterson argue that there are three key ingredients for protecting against attacks of that kind:

- *accounting* for all consumed resources per client;
- *detection* when the resources consumed by any given client exceed some limit;
- *containment* – the ability to reclaim the tied resources after detecting an attack by dedicating minimum additional server resources to the task and thus avoiding to fall for a follow-up denial of service attack.

This goes back to the priorities of how the Internet was originally designed – with very little provisioning for how resources in the network could be accounted for [Cla88]. In fact, accountability was reportedly the lowest priority goal in the design of the Internet (as compared to the ubiquitous phone network, where it was a major goal). No wonder that in the end one of the most serious threats in the network dwells exactly on this lack of accountability. Indeed, if servers could more effectively manage their available resources, they would unlikely fall into a situation where those resources are easily tied up, while more (possibly legitimate) incoming requests are knocking on the door. One solution for making it harder to deplete a server's resources and allowing graceful degradation of performance in the face of such a threat, is discussed in [JB99] as we will discuss in more detail in the following section.

A third contributor to the success and effectiveness of these types of attacks is the unsophisticated processing of incoming packets in the network servers under high load conditions [DB99]. Many modern operating systems incorporate *interrupt-driven network subsystem architectures*, which have been shown to lack both efficiency and stability under conditions of high network load. The problem comes from the fact that they give strictly highest priority to processing of incoming network packets, regardless of which application those packets belong to, whether or not this application is currently executing and whether or not this receiver application has lower priority that the currently executing one. As a result, a situation known as *receiver livelock* could potentially occur, where the network server spends all of its resources processing incoming packets, only to later discard them because no CPU time was left to service application programs. In short, under the current mechanisms packets are dropped only after server resources have been invested in them, which is clearly prone to denial of service attacks – untrusted application programs running on a client could cause the failure of a shared server. In their paper Druschel and Banga argue against the conventional interrupt-driven network subsystem architecture and the eager network processing as an approach and in turn propose a lazy receiver processing architecture which rests on early packet demultiplexing, early packet discard and processing of incoming network packets at the receiver's priority. The argument is that the new mechanism leads to improved fairness, throughput and stability under overload, while not suffering from degraded performance under normal load conditions.

2. Types of attacks

In terms of the number of malicious entities involved in an attack, we distinguish:

- *uni-source attacks* – launched by and originating from a single source;

- *distributed attacks* – originating from multiple coordinated sources, though not necessarily involving more than one malicious end user.

Distributed DoS attacks operate on a much broader scale (with practically limitless number of launch sites) and can considerably add to the severity, length and scale of an attack, making it possible to practically disable even very powerful servers over prolonged periods of time. Such was the case with the servers of large commercial sites like Yahoo!, eBay.com, etc. in early February 2000. Since then, distributed attacks have turned from a theoretical possibility to a major concern for Internet servers of any size and computing power.

An attacker breaks into multiple random (i.e. not necessarily related) ill-secured "*zombie*" (a.k.a. "*stepping stone*") end hosts. She then installs publicly available attacking tools (Trin00, Tribe Flood Network, Mstream [Ditt00]) – the fourth culprit for the proliferation of denial of service attacks – on these launch pads. Finally, in a moment of choice a massive coordinated attack is launched against a target server. Withstanding such an overwhelming burst of requests is hardly possible (on the order of a few thousand launch sites were estimated to have been used during the attacks mentioned above) and the server is surely disabled, denying legitimate users any service. A figurative real-world equivalent is the so called pizza delivery attack [Sch00]. Alice does not like Bob, so she calls multiple pizza delivery parlors and orders one pizza from each, to be delivered to Bob's house at some given time. When the moment comes, Bob is overwhelmed by the host of pizza deliverers arriving at his house and demanding their money. Simple yet very effective, if Alice had called from a public payphone (essentially disguising her identity), there is nothing Bob or the pizza parlors could do to even hope to find out who played the trick on them in the first place.

3. IP "spoofing"

When an attack is detected and hopefully recorded, the next thing the victim wants to do is to find out who the originator was. This turns out to be a really hard problem in the Internet. Much like in the case with Alice calling from a public payphone thereby not leaving a trace of who she was, it is possible (and easy) to disguise your identity in today's Internet. Ironically, it is the senders who put in their source address in an IP packet, so a malicious client could choose to stick any IP address in (a technique known as "*spoofing*") and pretend to be sending from somewhere else.

To make matters even worse, in the case of a distributed attack the zombie sources are many and their addresses likely to be spoofed. In order to get the real attacker, three additional important steps are necessary, all of which need to be satisfactorily resolved and none of which (to date) have an engineered and deployed solution:

- *tracing back to the sources* (zombies). Savage et al. [SWKA00] have proposed a scheme for doing this which we will briefly discuss in the next section. It needs to be recognized that zombies are unlikely to be reused by an attacker – it would be too risky for her and there are sufficiently many ill-secured zombie-to-be candidate end hosts out there. Hence, relying on correlating recorded traces from many different attacks may prove to be an all-too-optimistic expectation;
- *tracing back to the original source* of an attack. Finding out where the seeds were initially planted is another challenge. This can be achieved by examining the logs of the corresponding ISPs. A drawback of that approach is that it is bound to take much human effort and is therefore unlikely to be a viable solution, except for only a few critical cases of attacks against victim customers willing to invest in pursuing the attacker. In addition, innocent launch-pad end users may be unwilling to cooperate and be monitored, especially over the course of a longer period of time;
- *tracking down the attacker* herself. This last step involves not so much technical challenges, but rather successful coordination of efforts between ISPs, telecommunication companies and law enforcement agencies. In [Need94] Needham points out that in all security matters there are two objectives – to make violations difficult and to make them known to authority when they happen. With the case of denial of service, meeting the first objective is somewhat limited (at least based on the current state of the art), so in the meantime it is essential to be effective in meeting the second objective. However, as we mentioned before the management overhead in connection to this goal may be just too much to cope with, unless there is an overwhelmingly good (non-technical) reason to pursue it. On the technical side, supporting no non-repudiation (which is the case for the traffic causing denial of service) means that there is no way to prove who the actual initiator was.

4. Significance

Finally, let us consider the importance of coping with denial of service as a security phenomenon.

Denial of service attacks could disable servers for potentially long periods of time. During that time between the onset of such an attack and the time when the breach is actually detected and recovered from, the victim server is unable to handle any requests by legitimate non-malicious users. For large commercial servers this translates to a significant loss of income, and which they consider even more serious – a loss of reputation. To be concrete, one report estimated the total loss due to the distributed attacks we mentioned in the range of $1.2 billion.

There is also another important dimension for assessing the damage from such attacks. Although denial of service itself does not directly affect data residing on the server under attack, there is no reason not to anticipate it being used as a first step to another follow-up attack, which actually does steal, alter or somehow manipulate that data. And the data could be mission- or life-critical in some occasions. This kind of chained attack can realistically happen if the protocol the server executes is not fail-stop (or at least fail-safe) [GS98]. In a little more detail, a protocol is considered fail-stop if it automatically halts in response to any active attack that interferes with the protocol execution. The notions of fail-stop and fail-safe protocols are quite useful for formal verification of certain properties of security protocols, as discussed at length by Gong and Syverson. For instance, Bro [Pax98] is an intrusion detection system in use, which is fail-open (due to the need to operate in real-time). Under normal traffic conditions the system looks at packets (headers) and decides which ones to let within a designated administrative boundary it serves to protect. However, if Bro becomes overloaded, it simply lets packets through while trying to catch up on processing a queue of outstanding ones. This can be used by skillful attackers to penetrate into the domain and do further (more serious) damage there.

III. Suggested Remedial Approaches

Naturally, as with every kind of security breach, there are three general approaches for dealing with the attack – eliminating it completely, mitigating the effects of the attack on the victim, and discouraging the attacker. Those do not have to be exclusive to one another, but can and should be used as complementary, whenever possible. We will look at each strategy separately in more detail, considering the variety of solutions proposed in literature and pointing out what we think are the strong and weak sides of each.

1. Eliminating the possibility of attack

This is by far the most desirable strategy for "defending" against any kind of security attack. Unfortunately, the problems are rather complicated and very seldom can a threat be completely eliminated. More often that not, this would be the case with scams, which really only have transient effects.

1.1. Allowing connections *only* to trusted clients. This is clearly the most conservative approach to communication and as such it has the highest degree of averting security threats. Such a solution is justifiable for deployment only in closed and special-purpose (e.g. military) environments. It is inherently inapplicable and incompatible to an open communication system such as the Internet. A known problem with closed environments is that outside intrusions are both not expected and commonly not anticipated. So, the level of preparedness for a security breach, should it ever occur, is very low and the damage grows proportionally high.

1.2. Out-of-band signaling. This is not a novel idea, especially in the phone network field. The idea is that control and data signals would travel physically on separate wires and thus any interference and possible confusion is excluded. This was not the case with the phone networks in 1960s, when in-band signaling was in use. One could whistle into the phone receiver and under certain favorable circumstances (the right wavelength and amplitude) the signal, which really was just data, could be interpreted as a control signal (e.g. a free call, etc.).

Schneier claims that out-of-band signaling would not only alleviate the existing problems with denial of service attacks, but also aid in defeating other inherent known security problems in the Internet. Since the claim was not supported by any detailed explanations, to us the question remains open as to whether and to what extent this is really true. In any case, a significant re-engineering of the Internet is not in sight, so this idea may have more theoretical than practical value.

As it should have become clear from the above discussion, none of the strategies for eliminating the possibility of attacks are practically reasonable or deployable for the case of an open wide-area network.

2. Mitigating the effect of the attack on the victim

While the utmost goal is to avoid being attacked, when and if this happens it is highly desirable to be able to sustain some level of (degraded) performance during the high load, even prior to the actual detection of an attack. The following approaches try to achieve this goal in different ways.

2.1. Securing all computers on a network. Achieving that would render the existence of zombies impossible and hence an attacker would be reduced to being able to mount only a uni-source attack. In addition, IP traceback schemes would directly lead to the attacker's weapon machine, which in turn would both reduce the management overhead in the post-mortem tracing process and serve as a disincentive for the attacker to start in the first place. The most significant drawback of this approach is that it is unimaginable to first deploy for a rapidly growing wide-area environment (such as the Internet), and then to continually upgrade and maintain the same level of security everywhere.

2.2. Ingress filtering. This approach is targeted at reducing or completely eliminating the ability to forge source addresses, which if accomplished would ultimately result in much easier tracing back to the true source of an attack and as such would serve as a significant deterrent for attackers. Ingress filtering is to be done in the routers, which ideally should not allow packets with illegitimate source addresses through [FS98]. Obviously, the router closest to the source has the greatest chance of being able to decide whether a packet *could* have a fake source address or not. Note, that it may be the case that an address is forged, but still a valid address of another customer of the same network.

A minus of the approach is that on high speed links, comparison of packet headers against filters becomes an expensive operation and reduces network bandwidth, so it is likely to be not out of favor for both the customers and the service providers (ISPs). Another downside is that ingress filtering relies heavily on widespread deployment in the routers in order to be effective. Mobile IP too would suffer from additional complications arising from the deployment of ingress filtering.

2.3. Client "puzzles" prior to committing resources. The idea, recently proposed by two RSA researchers, is to distribute cryptographic puzzles to clients (whether genuine of fake) when the server comes under pressure from high load [JB99]. Resources are only committed to connections, for which the clients have successfully solved and submitted their puzzles within a timeout period. This strategy would serve two purposes: damping (i.e. spacing) the client requests and allowing the server to control the number of clients establishing connections with it by sending out harder puzzles (depending on the load conditions and actual number of resource slots available), and thus experiencing *graceful degradation* in performance. A good feature to note is that client puzzles are resistant against IP spoofing.

Into more details, a puzzle consists of a number of sub-puzzles, chosen carefully by the server. Each sub-puzzle itself is computed by hashing (using a known one-way hash function) a bitstring and sending the client the result of that hash along with the partially revealed bitstring. Assuming that the hash function is computationally irreversible (and it will be chosen to have that property), the client will need to perform brute-force search on the space of all possible bitstrings in order to discover the original bitstring, which he then sends back to the server. The original paper claims that to compute the puzzle and then to verify it, the server would only spend on the order of 1%-4% of the time it would otherwise take it to establish a TCP connection.

The main disadvantage of client puzzles, admitted by the authors themselves is the requirement for special client-side software (either built into the browser or distributed in some different way). Another drawback, remarked by Schneier [Sch00] is that client puzzles although reasonable as an approach for dealing with uni-source attacks, fall to distributed denial of service attacks, since generation and distribution of client puzzles is a denial of service attack in its own right. Overall, it seems that the approach is viable, but should only be used with caution and in combination with other such promising approaches.

2.4. Use progressively stronger authentication. The idea here is to again avoid committing server resources early, trying to instead incrementally gain confidence in the identity of the client and only "promising" resources, proportionate to the level of assurance the server has at any one point during the communication [Mea99]. Starting with weak authentication first (e.g. a cookie) and upon receiving positive feedback (i.e. client responding and following the requested steps), the server progressively chooses

stronger authentication up to the point of doing an expensive authentication (e.g. digital signature), at which point the real conversation may start. Note that "strong authentication from the start would be a hook for denial of service attacks". This approach in itself does not prevent malicious parties from launching attacks, but it significantly raises the bar for doing that, making the attackers work harder by first having to dispose of the weaker authentication.

Unlike the approaches for completely eliminating the threat of attacks, the ones for actually dealing with it are in our opinion quite reasonable and realistic, albeit none of them offers a complete set of strategy on how to mitigate denial of service attacks.
Other such possible approaches are discussed and compared in [SWKA00].

3. Discouraging the attacker

The strategies discussed in this subsection should not be observed as separate from the rest in this and other papers. They are merely meant to augment existing other techniques for mitigating denial of service attacks. Only with the joint effort on all fronts can such a massive breach be ultimately defeated.

3.1. IP traceback by coordinating between ISPs. Coordination between ISPs is a feature that is highly desirable for tracking down security problems of all kinds. If accomplished, it could significantly raise the bar, forcing attackers to be even more inventive and resourceful (e.g. forcing them to break into and take over other zombie end hosts). However it is one of the hardest things to manage and in addition requires that (most of) the ISPs keep accurate logs and willingness to share them among each other in order to effectively coordinate a traceback. It seems that this could only be achieved if the ISPs themselves have some incentive in participating in such a joint effort. One such thing could be a form of administrative control and mandating them to cooperate, much like governments cooperate in cracking down on international crime. An even more convincing incentive would be to keep them off the list of ISPs, recommended to avoid for being insecure and careless.

Another drawback of the general approach (other than there being no management inter-ISP infrastructure) is that if a tool for cracking passwords is made publicly available (and this could happen) then the effect of raising the bar for attackers would be nullified.

3.2. IP traceback by probabilistically marking packets. This last approach is allegedly the most promising direction for discouraging attackers. It is both robust (using randomizing), incrementally deployable and backwards compatible [SWKA00]. The method is resistant against IP spoofing and against distributed denial of service attacks.

The key idea is to probabilistically mark packets at routers (over which the attackers obviously can have no control) with partial path information, which packets will carry. The entire path could be reconstructed post-mortem at the victim server. Assuming that a sufficient number of packets per source are received (simulations show that on the order of a few thousand packets will suffice), there is a very good chance of being able to reconstruct the path back to that source. Statistics show that most flooding denial of service attacks send many more packets over a short interval of time.

A drawback of the approach (as with the IP traceback approach in general) is that the attacker may be physically in a country with "few computer crime laws and bribable police", which will render all the effort of tracking her down useless.

The contribution of this approach is clearly an achievement in the effort to defend against denial of service attacks, given that no per-flow state in the routers or other changes in the infrastructure are necessary.

IV. Conclusions and Future Work

Complete elimination of denial of service threats is infeasible given the current Internet infrastructure. Internet, being an open environment with no limits set in stone on the number of users, is inherently vulnerable to attacks of the denial of service type. There is no way to predict the parameters of the largest possible flood. In the phone network the infrastructure is set and it is known what the provisions should be in order to reduce the risks to acceptable levels. Such a provision is hardly imaginable in the Internet, as it is.

Discussed approaches and strategies could be combined to offer various levels of mitigation of attacks and disincentive for the attackers, but complete set of tools for defense are currently not available both in

the academic and industrial communities. One possible high-investment solution might be in a new Internet where accountability has higher value. Another key idea is to improve the way network servers are implemented, e.g. using lazy receiver processing. A third possible direction to look at is how to discover and mitigate the effects of denial of service attacks when they do not completely flood a server, but still significantly constrain its effective use of resources, overwhelming it with bogus packets. Such an approach would be highly desirable and popular among the companies doing commerce on the Internet.

While short-term defenses could be found in the literature, there is a call for longer-term strategies against denial of service attacks.

References

[BC99] Hal Burch and Bill Cheswick, "Tracing anonymous packets to their approximate source", unpublished paper, Dec 1999.

[CERT96] Computer Emergency Response Team, "CERT advisory CA-96.26: Denial of service attack via pings", Dec 1996. (http://www.cert.org/advisories/CA-96.26.ping.html)

[CERT97] Computer Emergency Response Team, "CERT advisory CA-97.28: IP denial of service attacks", Dec 1997. (http://www.cert.org/advisories/CA-97-28.smurf.html)

[CERT98] Computer Emergency Response Team, "CERT advisory CA-98.01: 'smurf' IP denial of service attacks", Jan 1998. (http://www.cert.org/advisories/CA-98-01.smurf.html)

[CERT00] Computer Emergency Response Team, "CERT advisory CA-2000.01 Denial of service developments", Jan 2000. (http://www.cert.org/advisories/CA-2000-01.html)

[Cla88] David Clark, "The design philosophy of the DARPA Internet protocols", In proceedings of Communications architectures and protocols, pp.106-114, Stanford, CA, Aug 1988.

[DB96] Peter Druschel and Gaurav Banga, "Lazy receiver processing (LRP): a network subsystem architecture for server systems", In Proceedings of 2nd USENIX Symposium on OSDI, Seattle, Oct 1996.

[DEC96] Digital Equipment Corporation, "Performance tuning tips for Digital Unix", June 1996. (http://www.service.digital.com/manual/misc/perf-dec.html)

[Ditt00] Dave Dittrich, "The 'mstream' distributed denial of service attack tool", May 2000, (http://staff.washington.edu/dittrich/misc/mstream.analysis.txt); "The DoS project's 'trinoo' distributed denial of service attack tool" (http://staff.washington.edu/dittrich/misc/trinoo.analysis); "The "Tribe Flood Network" distributed denial of service attack tool" (http://staff.washington.edu/dittrich/misc/tfn.analysis)

[FS98] P. Ferguson and D. Senie, "Network ingress filtering: defeating denial of service attacks which employ IP source address spoofing", RFC 2267, Jan 1998.

[GS98] Li Gong and Paul Syverson, "Fail-stop protocols: An approach to designing secure protocols", 1998.

[How98] John D. Howard, "An analysis of security incidents in the Internet", PhD thesis, Carnegie Mellon University, 1998.

[JB99] Ari Juels and John Brainard, "Client puzzles: A cryptographic countermeasure against connection depletion attacks", In proceedings of the 1999 Networks and distributed system security symposium (NDSS'99), Internet society, Mar 1999. (http://www.isoc.org/ndss99/proceedings)

[Mea99] Catherine Meadows, "A formal framework and evaluation method for network denial of service", In Proceedings of the 1999 IEEE Computer Security Foundations Workshop, Mordano, Italy, June 1999.

[Pax98] Vern Paxson, "Bro: a system for detecting network intruders in real-time", In proceedings of the 7th USENIX Security Symposium, San Antonio, TX, Jan 1998.

[Sch00] Bruce Schneier, "Distributed denial of service attacks", Crypto-gram newsletter, Feb 2000. (http://www.counterpane.com/crypto-gram-0002.html#DistributedDenial-of-ServiceAttacks)

[SP99] Oliver Spatscheck and Larry Peterson, "Defending against denial of service in Scout", In proceedings of 3rd USENIX/ACM Symposium on OSDI, pp.59-72, Feb 1999.

[Need94] Roger M. Needham, "Denial of service: an example", Communications of the ACM, vol.37, No.11, pp.42-46, Nov 1994.

[SPEC96] The Standard Performance Evaluation Corporation, "SpecWeb96 benchmark results", 1998. (http://www.specbench.org/osg/web96/results)

[SWKA00] Stefan Savage, David Wetherall, Anna Karlin and Tom Anderson, "Practical network support for IP traceback", Technical report UW-CSE-00/02/01, In submission to SIGCOMM'00, Feb 2000.