TOWARDS NETWORK DENIAL OF SERVICE RESISTANT PROTOCOLS

Jussipekka Leiwo
Vrije Universiteit, FIW, De Boelelaan 1081A, 1081 HV Amsterdam, The Netherlands
leiwo@cs.vu.nl

Tuomas Aura, Pekka Nikander
Helsinki University of Technology, FIN-02015 HUT, Finland
{Tuomas.Aura,Pekka.Nikander}@hut.fi

Abstract

Networked and distributed systems have introduced a new significant threat to the availability of data and services: network denial of service attacks. A well known example is the TCP SYN flooding. In general, any statefull handshake protocol is vulnerable to similar attacks. This paper examines the network denial of service in detail and surveys and compares different approaches towards preventing the attacks. As a conclusion, a number of protocol design principles are identified essential in designing network denial of service resistant protocols, and examples provided on applying the principles.

1. INTRODUCTION

The ITSEC [15] defines availability as a property of both data and services, whereas confidentiality and integrity are defined only as properties of data. Being broadly defined, only limited protection of availability can be achieved through technical security measures. Administrative and managerial measures, such as transfer of responsibility, are required for dealing with external accidental threats [23].

One means of causing a loss of availability is through a denial of service (DoS) attack. These attacks can occur either through resource allocation or resource destruction [26], or through random failures.

TCP SYN flooding [5, 36] is a typical attack through resource allocation. TCP/IP Ping attack [4] and MIME vulnerabilities [3] have demonstrated the severity of DoS through resource destruction.

The buffer overflow problem that causes Ping and MIME attacks can be dealt with safe string handling (e.g. [28]) and on-line monitoring [6]. Applications developed using languages with appropriate boundary checks, such as C++ and Java, are not vulnerable. Therefore, we shall focus on network DoS through resource allocation. Any protocol where the server commits to extensive computations (e.g. public key cryptography) or to memory allocation (e.g. storing the protocol state) prior to or as part of client authentication, is vulnerable to network DoS.

The 1985 US DoD workshop [10] concluded that no generic, mission-independent DoS conditions can be identified. In 1999, The Committee on Information Systems Trustworthiness [35] suggests that neither DoS prevention methods nor systematic design methods exist against DoS, and that ad-hoc measures used successfully for securing time sharing systems are invalid in networked information systems.

There are, however, a number of ways for dealing with network DoS. This paper surveys them and establishes a number of design principles for network DoS resistant communication protocols.

2. TERMINOLOGY

Availability is defined broadly as the property of data and services being accessible to an authorized party within a reasonable time of a request.

Defining availability in terms of response time enables rigorous definition of constraints within which the system is expected to operate. For example, [11] suggests, but doesn't further elaborate, temporal attributes being associated with the integrity constraints to deal with availability.

Difficulties with time in distributed systems, however, complicate definition of exact constraints for systems (e.g. [2]). The Internet, for example, does not provide delivery time guarantees, hence not providing infrastructural support for distributed applications with strict time constraints. Even in best cases, the sequence of events may be the only meaningful measure of time.

Availability is also used in other contexts in information security. Denning [9], for example, refers to the defensive and offensive information warfare in terms of increase and reduction of the availability of information to various parties.

Denial of Service (DoS) is a result of the realization of an intentional threat against availability. Accidental threats must be dealt with administrative and external measures [23]. In a network threat model this means, that an authorized user is prevented from accessing a service because of the intentional action of an attacker. The attacker can either destroy a certain resource or exhaust the resources of the service provider.

Network DoS occurs when a remote party successfully denies access to a service. A DoS attack may occur locally, but network DoS involves an attack through the communication interface.

In a TCP SYN flooding attack, the buffer that stores the initiated but not completed TCP session requests is exhausted. The attack can be combined with other attacks, such as IP spoofing, to prevent detection. The major difficulty is that the initial message is unauthentic, and therefore the attack hiding mechanisms, such as IP spoofing, are likely to succeed in hiding the attacker's identity.

3. PRELIMINARY OBSERVATIONS

We shall survey a number of approaches towards availability that have been suggested but fail to protect systems against network DoS.

Process scheduling (e.g. [37]) prevents a single process from unfairly allocating excessive resources. It only works if processes can be created on behalf of entities under the control of the operating systems. Demand for services should not exceed the available capacity. Under a systematic attack, process scheduling fails.

Low priority can be assigned to unauthentic or suspicious processes (as the Synkill[36] tool) but it remains difficult to allocate the priorities prior to the authentication of users as attacks often exploit unauthentic communication channels.

The dependable computing community [16] sees availability as a common requirement for each system, whereas the need for reliability, safety and security varies depending on the application. Dependability measures, however, assume a random and independent failure model, whereas in the network DoS attacks, one must assume the presence of an attacker that intentionally and systematically attempts to violate the availability of the system and services.

Keus and Ullman [21, 20] suggest that the study of DoS should be differentiated into concepts such as operability. The ITSEC [15] sets a requirement of continuity of service to achieve availability. Parker [31, 32, 33] has suggested a new definition of information security with a large number of intuitively appealing concepts, such as utility.

None of these , however, can answer the fundamental question of defining constraints under which a system is operable, or the service is continuous. Utility remains especially vaguely defined.

Gligor [12, 13] has demonstrated that DoS in operating systems must be approached through resource allocation, not through access control. Gligor introduces inter-user dependency and demonstrates that as a common cause of DoS problems. Yu and Gligor [40] further propose a model for dealing with DoS based on the notation of user agreement. Other similar models are proposed in [22, 26, 27].

The assumption with the resource allocation model is that there are appropriate resources to satisfy each request. While this may be the case at a single computer system, there are no meaningful ways to extend the approach to networked systems. Instead, the approach may lead to a highly undesirable ungraceful service level reduction when a certain threshold of incoming service requests is reached. This doesn't prevent DoS, only guarantees (at best) lack of random behavior under attack.

4. CONTEMPORARY APPROACHES

Contemporary approaches towards network DoS resistant protocol design shall be surveyed to justify the proposed protocol design principles.

Network inhibition [34] is a graph theoretical DoS attack where an attacker disables network elements in order to disconnect communicating parties. Quantifiable measures can be given for the security against DoS, namely that of the number of network components the attacker needs to disable – using limited resources only – in order to succeed.

The MIN CUT algorithm [39] enables attacker to find an optimal attack in a polynomial time. Partitioning network into separate components is also polynomial [7]. Disconnecting 3 or more given nodes from each other is NP complete [8], as is denying connection between two parties with the possibility of partial disconnection of an edge [34].

Needham [29, 30] suggests that protocol design methods are most appropriate to prevent network DoS. Stateless protocols [1] are a means of trading off the need for state storage space to an increasing protocol message size and cryptographic computations. Instead of storing the protocol state, the server sends an authenticated protocol state to a client and expects client to return the state with further messages. With fast MAC algorithms, the overhead is minimal. The server does not commit to the storage until the authenticity of the client has been established.

ISAKMP [24] requires client to return a server–generated cookie, derived from PHOTURIS [18, 19, 38], that can be bound to the identity of the client and verified by the server before commencing a costly authentication protocol. The cookie should be such that it is hard for a malicious client to alter but easy for the server to verify. Meadows [25] has formalized the cookie approach using a special cost function, and a measure of increasingly strengthening the authenticity of communicating parties.

Juels and Brainard [17] introduce client puzzles to commit a client into the protocol prior to authentication against what they call connection depletion attacks. Hirose and Matsuura [14] propose a hybrid of cryptographic measures and stateless protocols to prevent DoS attacks. The server gains assurance of the good intentions of a client is gained gradually through a series of light weight computations, mostly hash functions, prior to signature verifications.

Even though most tools to deal with TCP SYN floodings were integrated into firewalls, tools, such as Synkill [36] were developed. Synkill is a background process that monitors the state of TCP session establishment and, depending on clients' trustworthiness, may prevent certain connections under suspicious circumstances. This is the only approach that can be implemented without extra networking equipment. However, the situation where such tools are required, i.e. the presence of the threat of network DoS in standard protocols, is highly undesirable. New protocols should be designed to be network DoS resistant. In the following, the network DoS attack scenarios are described, and protocol design principles identified.

5. ATTACKS AND ATTACK METHODS

Tolerable attacks can be prevented by proper protocol design. The weakest attack is **normal protocol execution only**. Clients can only trigger correct protocol executions. However, a single client can allocate all the resources if it is not prevented by the protocol design.

Deviation from protocol message sequence is where clients can intentionally or accidentally deviate from the sequence of protocol execution. If combined with masquerade, such as IP spoofing, the TCP SYN attacks demonstrates that these attacks may become very serious. The most likely scenario is the exhaustion of server memory, as in the TCP SYN flooding due to the failure to complete the handshake protocol.

Deviation from protocol message syntax is where incorrect protocol messages can be fabricated by the client to harm the server, for example as in the TCP Ping attack. This may cause resource destruction but can also be used in other attacks to fully or partially alter the server state, for example by inserting malicious function calls to the program execution stack.

Deviation from protocol message semantics enables attackers to fabricate the content of protocol fields by will. It is merely a tool to hide the identity of the client, e.g. by IP spoofing, when combined with other methods of attack. To prevent these attacks, it is important that the server does not commit into intensive computation or memory allocation until the clients are properly authenticated. Network bandwidth exhaustion is also possible if the attacker succeeds in, for example, modifying the routing information of packets and generates large quantities of looping traffic with high time-to-live value.

Fabrication of protocol messages can be used for attacking the server or the network infrastructure. By successfully falsifying the routing information or creating error notification messages the server throughput can be reduced by reducing the bandwidth and processing capacity left for legitimate clients. Compromise of network components leads easily to fatal network DoS scenarios.

Fatal attacks lead to the destruction of resources in an uncontrollable manner. A typical example is a server intrusion. If an attacker can gain access to the server being attacked or any crucial network element, especially with administrative privileges, there are more effective means of denying the service than through communication protocols.

If server penetrations are successful, more severe threats than network DoS can be expected. Therefore, different from the cryptographic attack models, the adversary can not be assumed to have full control over the network. Physically controlling the communication line, a simple physical attack will have fatal consequences to the availability of services. In a broadcast network, any attacker can jam the radio frequencies and prevent any communications.

6. PROTOCOL DESIGN PRINCIPLES

In addition to trivial implementation decisions, such as increasing buffer sizes, a number of design principles can be established to aid in designing network DoS resistant protocols.

Memory should only be allocated after the client has been authenticated. As most network DoS attacks are supported by masquerading, allocating memory at the request of arbitrary clients should be avoided.

Client authentication should only take place after easier means of detecting attacks are completed. There are often more effective means to detect attempts to replay protocol messages in order to hide the source of an attack than committing into the computationally intensive authentication procedure.

1. $C \rightarrow S : N_C$
2. $C \leftarrow S : h(N_S, H), t, H$
3. $C \rightarrow S : S_C(r_1, N_c, N'_S, t, C, S)$
4. $C \leftarrow S : S_S(r_2, N_c, N'_S, t, C, S)$

Figure 1 Network DoS resistant modification of the X.509 authentication protocol

The work load of a client should be higher than that of the server. In order to prevent a client from launching multiple attacks, the client workload should exhaust the resources required to carry out the attack. The work load of the client should also increase at least linearly whereas the work load of the server should remain as constant as possible and be independent from the client's work load.

The work load of the client should be parameterized and easily modifiable by the server. First, this enables modification of protocols for different application scenarios and client hardware. Second, modifiability of the client's work load enables reactiveness to the network traffic. In case of a suspected attack, the difficulty of protocol participation can be gradually increased to guarantee the survivability of the system under a suspected attack without significantly reducing the availability of the service to legitimate clients.

6.1. EXAMPLE

Figure 1 illustrates a network DoS resistant variant of the X.509 authentication protocol designed using the above principles. The client initiates the authentication by sending a nonce N_C. The server then generates a nonce N_S and calculates a hash value $H = h(C, S, t, K_S, N_S, N_C)$ where h is a collision free hash function, t is a time stamp, and K_S is a secret key of the server. The server sends the hash value H to the client together with the time stamp and a hash $h(N_S, H)$.

The client receives a partial solution to the hash $h(N_S, H)$ and can calculate, by brute force, a candidate N'_S. In normal circumstances, the entire hash value can be disclosed to the client but, under a suspected attack, the willingness of clients to commit to the protocol can be easily tested by increasing their work load. The server only needs to store the nonce N_S for each authentication request.

The client calculates the candidate N'_S and sends it signed to the server. Confidence of the server to the client's authenticity can be increased by five steps, the third one of which is the first to require extensive computations:

1 Check whether t is recent. This prevents replay attacks.

2 Check whether the pair $\langle N_S, t \rangle$ is unused. This detects attempts of multiple authentications per one reply by server.

3 Check whether $H = h(C, S, t, K_S, N'_S, N_C)$ to test client's willingness to solve the hash challenge issued by the server, i.e. the commitment of the client to the protocol.

4 Verify signature $S_C(r_1, N_c, N'_S, t, C, S)$ to gain full confidence on the identity of the client. This is only executed if the previous steps are successful, i.e. the probability of an attack is already considerably small.

5 Store a pair $\langle H_S, t \rangle$ to the memory. The server only commits to the memory allocation once the client has been properly authenticated.

For mutual authentication, the server can send similar signed message to the client that can proceed with the similar verifications.

7. CONCLUSIONS AND FUTURE WORK

A number of protocol design principles have been established to aid in the designing of network security protocols capable of withstanding network DoS attacks.

Obviously, the avenues for future work are numerous. Work is currently being carried out in implementing a number of modifications of well known security protocols to improve their resistance to network DoS. Furthermore, formulation of attack scenarios and types of attacks into a more comprehensive contemporary theory of availability is under way. As network DoS is a significant new type of a threat to networked and distributed information systems, fundamentally new ways of dealing with it are required. Active research is currently being carried out by the authors to further deepen the understanding of network DoS and possible countermeasures.

References

[1] T. Aura and P. Nikander. Stateless protocols. In *ICICS'97*, LNCS 1334. Springer–Verlag, 1997.

[2] K. P. Birman. *Building Secure and Reliable Applications.* Manning Publications Corporation, Greenvich, CT, USA, 1996.

[3] Buffer overflow in MIME-aware mail and news clients. CA-98.10.

[4] Denial-of-service attack via ping. CA-96.26.

[5] TCP SYN flooding and IP spoofing attacks. CA-96.21.

[6] C. Cowan *et al.* StackGuard: Automatic adaptive detection and prevention of buffer-overflow attacks. In *USENIX/Sec'98*.

[7] W. H. Cunningham. Optimal attack and reinforcement of a network. *J. ACM*, 32(3):549–561, 1985.

[8] E. Dahlhaus *et al.* The complexity of multiway cuts. In *24th ACM STOC*, 1992.

[9] D. E. Denning. *Information warfare and security.* Addison–Wesley Longman, Inc., Reading, MA, USA, 1999.

[10] Proc. DoD computer security center invitational workshop on network security. New Orleans, LA, USA, March 1985.

[11] J. Glasgow, G. MacEwen, and P. Panangaden. A logic for reasoning about security. *ACM TOCS*, 10(3):226–264, 1992.

[12] V. Gligor. A note on the denial-of-service problem. In *IEEE S&P*, 1983.

[13] V. D. Gligor. A note on denial-of-service in operating systems. *IEEE TOSE*, 10(3):320–324, 1984.

[14] S. Hirose and K. Matsuura. Enhancing the resistance of a provably secure key agreement protocol to a denial-of-service attack. In *ICICS'99*, LNCS 1726. Springer–Verlag, 1999.

[15] Information technology security evaluation criteria (ITSEC), version 1.2. COM(92) 298 final, Brussels, Sept., 1992.

[16] L. J.C., editor. *Dependability: Basic concepts and terminology.* Springer–Verlag, Vienna, AT, 1992.

[17] A. Juels and J. Brainard. Client puzzles: A cryptographic countermeasure against connection depletion attacks. In *1999 ISOC NDSS*.

[18] P. Karn and W. Simpson. Photuris: A session-key management protocol. IETF RFC 2522, 1999.

[19] P. Karn and W. Simpson. Photuris: Design criteria. In *SAC'99*, LNCS, Springer–Verlag, 1999.

[20] K. J. Keus and M. Ullman. Availability: A central and up-to-date user requirement for it security. In *10th ACSAC*, 1994.

[21] K. J. Keus and M. Ullman. Availability: Theory and fundamentals for practical evaluation and use. In *10th ACSAC*, 1994.

[22] J. Leiwo, C. Gamage, and Y. Zheng. A method to implement a denial of service protection base. In *ACISP'97*, LNCS 1270, Springer–Verlag, 1997.

[23] J. Leiwo and Y. Zheng. Layered protection of availability. In *1997 Pacific Asia Conference on Information Systems.*

[24] D. Maughan, M. Schertler, M. Schneider, and J. Turner. Internet security association and key management protocol (ISAKMP). IETF RFC 2408, 1998.

[25] C. Meadows. A formal framework and evaluation method for network denial of service. In *12th IEEE CSFW*, 1999.

[26] J. K. Millen. A resource allocation model for denial of service. In *IEEE S&P*, 1992.

[27] J. K. Millen. Denial of service: A perspective. In *Dependable Computing for Critical Applications 4*. Springer–Verlag, 1995.

[28] T. C. Miller. strlcpy and strlcat – consistent, safe, string copy and concatenation. OpenBSD project, http://www.openbsd.org, 1996.

[29] R. Needham. Denial of service. In *1st ACM CCS*, 1994.

[30] R. M. Needham. Denial of service: An example. *CACM*, 37(11):42–46, 1994.

[31] D. B. Parker. Restating the foundation of information security. In *IFIP/Sec'92.*

[32] D. B. Parker. A new framework for information security to avoid information anarchy. In *IFIP/Sec'95.*

[33] D. B. Parker. *Fighting Computer Crime: A New Framework for Protecting Information.* John Wiley & Sons, 1998.

[34] C. A. Phillips. The network inhibition problem. In *25th ACM STOC*, 1993.

[35] F. B. Schneider, ed. *Trust in Cyberspace.* NAP, 1999.

[36] C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A. Sundaram, and D. Zamboni. Analysis of a denial of service attack on TCP. In *IEEE S&P*, 1997.

[37] A. Silberschatz and P. B. Galvin. *Operating Systems Concepts.* Addison–Wesley, Reading, MA, USA, 1994.

[38] W. A. Simpson. IKE/ISAKMP considered harmful. *;login;*, 24(6):48–58, 1999.

[39] M. Stoer and F. Wagner. A simple min-cut algorithm. *J.ACM*, 44(4):585–591, 1997.

[40] C.-F. Yu and V. D. Gligor. A specification and verification method for preventing denial of service. *IEEE TOSE*, 16(6):581–592, 1990.