Roger M. Needham

Denial of Service: *An Example*

Security threats are often divided into three categories: breach of confidentiality, failure of authenticity, and unauthorized denial of service. The first two have been very extensively studied; confidentiality in particular has been pursued to extraordinary lengths. Indeed, some publications on confidentiality recall medieval disputes about how many angels could stand on the head of a pin. The second has been the subject of inquiry for many years, and is remarkable for the extent to which it is easy to devise wrong protocols. The third has been much less studied, and indeed, the tendency has been to dismiss it as a topic for serious inquiry (I did so in [3]). The objective of the present article is to consider a particular instance of a denial of service problem and to look at engineering considerations relevant to an appropriate defense. A major aspect is the complexity and danger that result from unthinking use of what seem to be simple cost-saving measures.

There are cases where the security threat that must be countered is almost exclusively one of denial of service. If there is a burglar in my vault, I do not care who tells me (no need for authenticity), I don't much care who else finds out (not much need for confidentiality), but I care very much that attempts to inform me are not balked (no denial of service). One could quibble with the detail of this example, in particular by discussing how one might defend against false alarms, but it seems incontrovertible that denial of service is the main threat. Much of the present discussion was in fact stimulated by a study of the infrastructure needed by alarm companies, undertaken for the U.K. insurance industry. The examples given do not relate to any specific product or service. The context for discussion is more structured than (say) the Internet, which is probably helpful.

In the context of an alarm system, we have three mechanical components to deal with, namely a *client* (a controller in the vault), a *network*, and a *server* (on an alarm company's premises). There are also two nonmechanical parts to the system—the *customer* and the *contractor*. The contractor uses the client, the network, and the server to give a service to the customer. We put it this way to emphasize that the denial of service against which we seek to protect is the denial of service to the customer, not to the client. The attack may indeed consist of disabling or destroying the client, just as it may consist of interfering with the network or with the server.

Attacks on the Server, the Network, or the Client

It is clearly possible to cause interruption of service by physical destruction of the server, and the means of making this less likely are mostly outside the field of interest of computer people. However, it is important to observe that the contractor may be presumed to know this has happened and, perhaps less plausibly, to have plans to deal with the contingency. It is clearly the responsibility of the contractor to assure itself of the integrity of the server, in particular by checking against unauthorized changes in its software. Such changes could, in principle, cause the server to decline, illegitimately, to give service to a particular client.

The obvious attack on the network is to cause it not to transmit messages necessary to give the required service either to all clients or to a class of clients. A less obvious attack is to cause it to send messages it should not send—as, for example, the simulation of a disabled client. A third possibility is to flood the network with enough messages to impede its proper functioning. If there is a place in the network that connects multiple clients, destruction of that facility should cause a great number of alarm signals, which may clog the network later on, overload the server if the messages reach it satisfactorily, or overload the means of physical response (for example, the police).

The main attacks on a client are destruction, with obvious consequences, and substitution. Substitution involves replacement of the client with an apparently similar one that will not give the service the customer believes has been purchased—for example, always reporting "all's well," even when there is a burglar.

Defenses

There are ready defenses against some attacks, and it is necessary to consider their scope, their limits, and most particularly, their objectives. Customers want to get the service they pay for, and if they do not, and are robbed, they will file claims with their insurance companies. Insurance companies do not wish to receive claims, and therefore want to be satisfied that the defenses are adequate. Manufacturers and vendors of clients do not want to be sued by insurers or contractors, and neither do the network providers. They all want to pass liability on in the direction of contractors, which themselves wish to have demonstrably received the alarm and called the police.

The best defenses with respect to each class of attack are (as usual) end-to-end defenses, though they do not deal with everything. The obvious manifestation of an end-to-end defense is a continuous regular handshake between the client and the server, which I will discuss in some detail. The purpose of such a handshake is to assure the contractor that the system is working properly, assuming that the contractor can rely on the good behavior of the server.

The contractor's computer will doubtless carry out many corporate functions; good practice would segregate the functions having to do with handshaking in a manner more secure than the rest. Although the information conveyed by such a handshake can, and probably should, be designed to be symmetrical, the result is not. The contractor can, if the handshake fails, send for the police without worrying about the cause of the problem. The case is very different for the customer; it should not be necessary to explain why it is a bad idea to have a visible indication on a burglar alarm control panel saying whether or not it is properly connected to the control room—or to make it emit a loud noise if it is not.

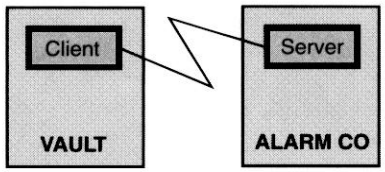

Figure 1. Overall structure

The handshake itself needs to be done with a little care. It must not be easy to interfere with the network so as to cause it to simulate either end of the handshake; the proper technique here is to use an encrypted serial number or the equivalent. The details of how to do this are influenced by a human aspect of system management, in a not very obvious way. One of the best ways to subvert a security system is to bring it into disrepute with the people who have to work it. Apparently, one of the most effective ways a burglar can attack an alarmed vault is to cut the wire to it and retreat a short distance. The police, the alarm company, and the customer all arrive, find the system isn't working, and say "Oh, well, we'll fix it in the morning," or words to that effect. The burglar then enters. This little anecdote applies to a potential attack on the communication network in which apparent failures are often produced for particular customers to generate the feeling that their systems are unreliable and not to be heeded too much. (It isn't unknown to blow up a telephone exchange to facilitate burglary, so sophisticated methods may be developed too.)

To defend against this, it is desirable that as far as possible the network not know to which customer a particular handshake message pertains, so that it is not easy to damage it selectively. This implies some things both about the network and, indirectly, about the messages. In both cases there turn out to be apparently conflicting requirements that need serious design consideration.

- ***The network.*** As much as possible of the traffic over the network should be unidentifiable, to make it as difficult as possible to generate selective unreliability. Any part of the network that cannot be run that way should be physically secured with great care. It is very desirable for the provider, however, to be able to monitor its network to see whether everything is satisfactory at its level, or so as to be able to resolve disputes between insurers and alarm companies, which initially seems incompatible with the first requirement. There appears to be a case for the use of an arrangement sometimes mentioned in principle but without much apparent use—allowing the network operator to record all messages but not to be able to interpret them until 24 hours later.
- ***The messages.*** Because of this, a message sent from a client to the server also has apparently contradictory requirements. It must be encrypted to prevent forgery but cannot be labeled with a highly visible sender identity for the reason just mentioned. Since the server will receive many messages from many origins, each message had better be accompanied by a certificate in the sense of Davis and Swick [2], telling which key to use to decrypt it. These certificates should be periodically changed in some suitable way to prevent recognition.

Capacity Problems

The contractor has (or hopes to have) a great many customers, and wishes

to provide for them appropriately but not too much. Presumably most of the vaults are burglar-free most of the time, and the contractor may not want to pay for a computer powerful enough to handle alarms from all of them at once (which translates into not being able to conduct fast handshakes with all of them). Similarly, the contractor may not want to pay the network provider for enough capacity to conduct a universal rapid handshake.

The former matter is between the contractor and its insurer. The insurer will presumably specify that the contractor should provide for a given number of simultaneous alarms per thousand customers.

The latter question is rather different, having what seems an obvious solution: delegate some of the duty to a subserver close enough to the clients to be able to poll them rapidly. This is a major step and one that should be taken only after due thought, because it has a serious effect on the trust relationships on which the system's security depends. If the handshake is genuinely end-to-end, the whole operation depends only on the reliability of the operation of the network, not on its honesty. This may seem a fine distinction, but it is important. Consider a case in which the ability of the network operator to monitor the traffic in detail is sacrificed in favor of the need for anonymity of messages. A dishonest employee of the network operator is not in a much better position to interfere with the reporting from a particular client than a member of the general public, being unable to recognize the client's messages inside the network, where he or she has privileged access. We may express the situation by saying the network operator is outside the trust envelope.

Suppose now that we do introduce a subserver, which will probably be essentially a concentrator and will reside on the network operator's premises—presumably in a switching office. The task of the subserver is to maintain the handshake and to report to the contractor's server only if something is wrong. The bandwidth from the network to the main server is much reduced—but the subserver and the people who operate and maintain it have been brought inside the trust envelope. If the subserver is operated by the network provider, then it is inside the trust envelope, with consequent needs to evaluate employees, audit hiring practices, and so forth. What began as an economy measure results in a security danger. It is an optimization in the classic sense—replacing something that is expensive but works with something that is cheap and "sort of" works.

A classic attack on network services is to cause excessive traffic so that legitimate traffic may be obstructed, lost, or rendered unprocessable by an overloaded server. As a general proposition this is very hard to defend against, but in a specific circumstance, such as our example, the problem is more tractable. The basic requirements appear to be twofold. One is that all end devices attached to the network be able to receive, decrypt, and if necessary, discard material arriving at line speeds while also performing their primary functions. They can then not be deceived by inappropriate input, even if some way is found to send such input. The other is that the network be set up in such a way that material may be sent only through predetermined paths, either set up statically, as is likely in the burglar alarm example, or negotiated, as in networks made from asynchronous transfer mode technology (ATM) switches. These paths should ideally have guaranteed bandwidth.

It is possible to see denial of service by network flooding as another instance of the bad effects of optimization. If insufficient bandwidth is provided on the grounds that not every sensor will call for service at once, a vulnerability to flooding attacks is present from the start. Network flooding is also related to the subserver issue: the subserver itself, if attacked, will cause the contractor to see alarms for all relevant customers, which may generate overload.

Denial of Service, Revisited

The foregoing discussion has been centered on a very particular application, in which it is possible, and necessary, to do a fairly complete job. What particular aspects make this problem easier than, for example, applications on the Internet?

The leading aspect is that in the present example we know what we are trying to achieve, the extent to which we are trying to achieve it, and why. The contractor wishes to provide a service that customers will pay for and that their insurers will accept as prudent protection of their vaults. The contractor also wishes not to lose lawsuits with its customers when things go wrong, and wants *its* insurance premiums to be reasonable. The network provider does not wish to be sued either, but to be able to pass liability along to the contractor. The important thing about the involvement of insurers is that they are arbiters of risk—that is their business. Within the Internet or similar contexts the matter tends, so far, to be much less concrete.

The next aspect is that the structure of the whole application is early bound. We know at any time how many customers a contractor has, where they are, and how the communication is meant to flow. If adequate provisions have been made, we know with some precision what communication demands there will be and how many transactions a computer will be called upon to handle. We know, in short, the parameters of the largest possible flood; we may not provide for it, but we can discuss rationally the amount of provision there should be to reduce risk to acceptable levels. In the Internet or similar contexts none of this knowledge is present; we do not even know how many potential clients there are, let alone how they will behave. The structure is not merely late bound—it is not bound at all.

At a different level, a clear distinction emerges between *selective* and *unselective* denial of service, though careful phrasing is needed here. Unselective denial of service is always possible by means of explosives. (Some types of selective or semi-selective denial can be accomplished this way too, as by destroying the client or by destroying part of the network.) These attacks should be very noticeable, however, and are certainly de-

tected by end-to-end checks. Selective denial of service in which a particular customer is attacked without it being evident that something is wrong is more insidious but fortunately also more amenable to protection. The most crucial requirement seems to be anonymity of communication, which makes it difficult to attack one customer without attacking all customers.

In all security matters there are two objectives—to make violations difficult and to make them known to authority when they happen. In the case of denial of service, the balance between the two objectives swings quite far toward the latter for the simple reason that dynamite denies service quite effectively but only rarely causes, for example, failure of authenticity. It is important to realize, though, that the balance is always there. In the case of confidentiality and authenticity, there has been a tendency to assume that since the measures taken to prevent violation are perfect, it is unnecessary to notice violations as they occur, because there are, as a matter of policy, none to notice. This attitude leads straight to the class of problems set out by Anderson [1], and is inappropriate to serious engineering. ◘

References

1. Anderson, R. Why cryptosystems fail. In *Proceedings of the First ACM Conference on Communications and Computing Security*, 1993.
2. Davis, D., and Swick, R. Network security via private-key certificates. *ACM Operat. Syst. Rev. 24*, 4 (1990), 64–67.
3. Needham, R.M. Cryptography and secure channels. In *Distributed Systems*, S.J. Mullender, Ed. Addison-Wesley, Reading, Mass., 1993, pp. 531–541.

About the Author:

ROGER M. NEEDHAM is Professor of Computer Systems and Head of the Computer Laboratory at the University of Cambridge, UK. Current research interests include security, especially cryptographic protocols, and the design of ATM networks and switches. **Author's Present Address:** Computer Laboratory, University of Cambridge, Pembroke Street, Cambridge CB2 3QG, United Kingdom; email: rmn@cl.cam.ac.uk