DENIAL OF SERVICE

R.M. Needham
University of Cambridge

Security threats are often divided into three categories: breach of confidentiality, failure of authenticity, and unauthorised denial of service. The former two have been very extensively studied; the first in particular has been pursued to extraordinary lengths. Some publications on confidentiality indeed recall mediaeval disputes about how many angels may stand on the point 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 has tended to be dismissed as a topic for serious enquiry (the present writer didso in [1]).

This is perhaps strange, because there are cases where the threat, which must be countered, is almost exclusively one of denial of service. If there is a burglar in my bullion vault, I do not care at all 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 baulked (no denial of service). One could quibble with the detail of this example, 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 (UK) insurance industry. It is not wholly typical, perhaps, of denial of service problems, but when there is no known type-example it may be helpful to start with this reasonably concrete case.

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* (in an alarm company's premises). There are also two non-mechanical 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 emphasise 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. This paper does not consider issues of responsibility.

Attacks on the server

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

Attacks on the network

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 which it should not - as for example the simulation of a disabled client. A third possibility is to flood the

1st Conf.- Computer & Comm. Security '93-11/93 -VA,USA
© 1993 ACM 0-89791-629-8/93/0011...$1.50

network with enough messages to impede its proper use

Attacks on the client

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

Defences

The best defences in respect of each class of attacl are (as usual) end-to-end defences. It is worth spending a little time on their scope and limitations here, since they do not deal with everything. The obvious manifestation of an end-to-end defence is a continuous regular handshake between the client and the server. Although the information conveyed by such a handshake can, and probably should, be designed to be symmetrical, the result is not. The result of such a handshake is to assure the contractor that the system is working properly, assuming that the contractor relies on the good behaviour of the server. The contractor can, if the handshake fails, send for the police without worrying about what is 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. A second point concerns frequency of handshakes. The contractor has (or hopes to have) a great many customers. There may be a limit to the frequency with which the handshakes may in practice be done relating to the capacity of the server or to the costs of communication. This point will be returned to.

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 detail of how this is to be done is influenced by a human aspect of system management, in a slightly unobvious 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 best ways to attack an alarmed vault is to cut the wire to it and retire a short distance. The police and the alarm company and the customer all turn out, find the system isn't working, and say "Oh bother, we'll fix it in the morning" , or words to that effect. The burglar then enters. This little tale relates to a potential attack on the communication network (it isn't unknown to blow up a telephone exchange to facilitate burglary, so sophisticated methods may come too) in which apparent failures are regularly produced for particular customers, generating the feeling that their systems are unreliable and not to be heeded too much.

To defend against this it is desirable that as far as possible the network should not know to which customer a particular handshake message pertains. This implies some things about both the network and, indirectly, about the messages.

1. The network
   As much as possible of the traffic over the network should be unidentifiable. Any part of the network which cannot be run that way should be physically secured. For example, if a number of clients are connected to a multiplexor-demultiplexor then it should be secure, because by the nature of the component it has to be evident which client a particular message is from or to. Between the multiplexor-demultiplexor and the server then messages can in principle be made unidentifiable and the security needs are different.

2. The messages
   Because of the point just made a message sent from a client to the server has two apparently contradictory requirements. It must be encrypted to prevent forgery but cannot be labelled with the sender identity in clear 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] which says which key to use to decrypt it. These should be periodically changed in some irrelevant way to prevent recognition.

A practical problem with the use of end-to-end techniques is that it may be impracticable to conduct the handshake often enough to give the desired assurance. In this case it is necessary to delegate some of the duty to a sub-server which is close enough to the clients to be able to poll them fast. The contractor needs to have confidence in the sub-server's integrity, and since it is almost by definition not on the contractor's premises there will be concern about its physical security and also about the integrity of any software in it. Such a sub-server is highly likely to coincide with the multiplexor-demultiplexor mentioned above. The server will have to conduct regular handshakes with it, and the contractor will have to have a plan as to what to do if the handshake fails. All the contractor knows is that all his customers in Manchester, say, are going unwatched. Not an enviable state, and one which shows that it is much better to proceed strictly end-to-end if it is at all possible.

Nothing has so far been said about network flooding, and this is something that has nothing to do with end-to-end methods. The danger from this attack can be mitigated by appropriate network design and implementation. The basic requirements appear to be twofold. One is that all end devices attached to the network should be able to receive, decrypt, and if necessary discard material arriving at line speeds while also performing its proper function. It can then not be snowed by inappropriate input, even if some way is found to send such input. Secondly, the network should be such that material may only be sent through predetermined paths which are either statically set up (as is likely in the burglar alarm example) or at any rate are negotiated as in networks made from ATM switches (Asynchronous Transfer, not Automatic Teller). These paths should ideally have guaranteed bandwidth; the exact meaning of this remark needs further investigation, however.

Denial of Service, revisited

What is going on in this discussion is to observe that the best protection against denial of a particular service is to render a continuous service dependent on the same resources as the service being protected. If the only way to deny service is to interfere with one of the resources in the chain you will notice that soemthing is wrong in a very timely manner. It is nowhere near as easy to detect that soemthing os wrong only when the substantive service is needed.

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, and possibly by means of such techniques as network flooding. In parenthesis, some types of selective denial can be done this way too, as for example by destroying the client or, semi-selectively, by destroying part of the network. However these attacks should be very noticeable, and are certainly detected 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 subject 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 awkward to do and to make them known to authority when they happen. In the case of denial of service the balance between the two tilts quite far towards 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 realise, 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 to prevent violation are perfect it is unnecessary to notice violations as they occur, because there are none to notice. This attitude leads straight to the class of problems set out by Anderson [3], and is inappropriate to serious engineering.

References

[1] Needham, R.M. *Cryptography and Secure Channels*, in Distributed Systems, edited S.J. Mullender, pp 531-541, Addison-Wesley 1993

[2] Davis, D. and Swick, R. *Network Security via Private-Key Certificates*, ACM Operating Systems Review 24(4), pp64-67, 1990

[3] Anderson, R. *Why Cryptosystems Fail*, First ACM Conference on Communications and Computing Security, 1993