# MANAGING TIME FOR SERVICE AND SECURITY

**Ruth Nelson**
*Information System Security*
*48 Hardy Ave.*
*Watertown, MA 02172*

**Elizabeth Schwartz**
*University of Massachusetts at Boston*
*Department of Math and Computer Science*
*100 Morrissey Blvd.*
*Boston MA 02125*

## Abstract

*Most security policies and mechanisms are concerned with controlling the dissemination of information or assuring its integrity. System management policies emphasize allocating resources fairly. Combining techniques from security and management can allow us to develop countermeasures against many of the network attacks on our systems. In particular, security policies and mechanisms need to be extended to manage resources quantitatively, as well as in a binary manner. The question is not whether access to service is allowed or not, but how to provide all the authorized services to the limit allowed by the capacity of the system. This paper explores the idea of allocating time-limited resources to deter or counter attacks that can cause the system to overload or crash. These overloads can cause not only loss of system availability to legitimate, well-behaved users, but also loss of system integrity and information security.*

## Introduction

Most security policies and mechanisms are concerned with controlling the dissemination of information or assuring its integrity. System managers are also concerned about protecting information, but they put a lot of their effort into managing time-consumable resources: processing capacity and communications access. Combining techniques from security and management can allow us to develop countermeasures against many of the network attacks on our systems. Security policies and mechanisms need to be extended to manage resources quantitatively, as well as in a binary manner. Avoidance of system overloads through quantitative resource management will provide better system service, regardless of whether the overloads are the result of deliberate attacks or accidents. Since denial of service attacks can exploit security features and mechanisms to shut the system down or even to compromise some of the system security, system security can be improved by including fair allocation of service as a security goal.

Access control policies usually are binary: a user either is allowed access to a given resource or not. The resource is treated as a unitary object, with no attribute of quantity. This binary, non-quantitative quality of access control persists even in systems with refined security policies. The objects may become smaller, and the conditions for granting access may include conditions such as time of day, but eventually, access is either granted or denied.

System management policies, on the other hand, emphasize allocating resources fairly. The processing and storage resources are treated as finite consumables, and their capacity is critical. The question is not whether access to service is allowed or not, but how to provide all the authorized services to the limit allowed by the capability of the system. These policies address denial of service prevention: they try to prevent resources from being consumed unfairly and made unavailable for authorized use.

Attacks on system availability typically consume limited system resources and deny their fair use. We will examine some of these unfair uses and some ways of detecting and limiting them. Attacks on limited system resources can also cause systems to lose control over their information resources because they can undermine the mechanisms that manage access.

In last year's paper, "Unhelpfulness as a Security Policy," Nelson talked about using time and quantity control as an aid to protecting access to information. The idea was that some information could be released in a slow and unhelpful manner, thus deterring users from acquiring too much of it. This year's paper explores the idea of allocating time-limited resources to deter or counter attacks that can cause the system to overload or crash. These overloads can cause not only loss of system availability to legitimate, well-behaved users, but also loss of system integrity and information security.

## Characterizing the Resources

Most security models are concerned with data or information as the relevant resource. Information has some unique characteristics that make it particularly difficult to control. The most interesting of these is that information in a system is not consumed or conserved. "Sending" information or "giving" information does not take it away from the sender or giver. Instead, the information is copied and both parties have it. If a user is given access to some information, and then the access is later revoked, there is no way to tell whether the user has kept some or all of the information. Once you know something, no one can be sure that you have forgotten it.

Processing and communications capacity have very different characteristics. These are functions of time. They are limited in any particular system by the configuration of the system. If they are not used, they are lost. They cannot be duplicated or copied, nor do they grow with system demand.

Physical storage is also a limited resource that is a function of the system. It is also not shareable. If it is in use, it is not available until it is released.

## Examples

The following examples illustrate the non-binary, complex nature of resource management. We have included examples of malicious actions and also examples of unintentional damage caused by legitimate use and unusual circumstances. We believe that the system must control the results of overuse, whether intentional or not. Analysis of the event after the fact may yield evidence of malicious intent, and this is helpful for prosecution and deterrence strategy. However, there generally is no inherent way to determine intent at the time of an event, and attackers have been known to exploit the same system limitations experienced by "normal" users and to mimic "normal" usage in their attacks to hide their intent.

The intention of the examples is to highlight ways in which system service can be compromised due to overload, and to bring up some questions of appropriate policies and mechanisms for managing time-limited resources.

### 1. Death and the WELL

The WELL is an on-line conferencing system in Sausalito, California. It has received some attention in the security community recently in connection with the arrest of Kevin Mitnick. The WELL has many on-line "virtual communities," but it may be best known as the on-line home of the Grateful Dead community. The Grateful Dead conference is one of the most active conferences on the WELL.

In a conferencing system, users are allowed to access the system and post messages. On the WELL, messages are grouped into topics which are then grouped into conferences. Each topic is stored in a single file. As with any database system, multiple users can access the file to read, but only one user at a time can lock the file for write access. On a normal day, many users will be logged in simultaneously, but they will be distributed across various topics and conferences.

The day that Grateful Dead guitarist Jerry Garcia died was not a normal day at the WELL. Grateful Dead fans from all over the country heard rumors of Garcia's death and rushed to the WELL, first hoping for a denial, and then gathering to mourn and reminisce. As the news spread, hundreds of users tried to log on simultaneously, and all of them were accessing the same file. The WELL experienced a load far above normal, becoming almost unavailable because of delays. The overload of demand in one area nearly made the entire system almost unavailable for many users for many hours for any purpose at all.

A conferencing system does have some special requirements. It might be a reasonable policy decision for a conferencing system to continue to provide conferencing service up to the point where system capacity is swamped. This is an example of how a service that a system wants to provide (users accessing resources that they have a right to access) can cause a denial of services to other users.

### 2. Too Much Mail

E-mail allows users at any site to command network computing resources. It is a many-to-many service; users at any site on the Internet can by default send mail to any user on the receiving site. A remote machine makes a request to transmit a message; the receiving site runs a process to receive the message, and writes the message to one or to many files. The receiving machine may also send the message to another site on the internal network, or if the message is addressed to a mailing list, send the message on to a list of users, possibly requiring starting multiple processes to send the message on to multiple sites. Finally, increased mail activity can increase the size of the system logs. If the disk storage is full, system programs may either fail to log, or they may shut down if logging is essential to continued operation.

There are several ways that electronic mail can be exploited to deny service, and mail attacks are common on the Internet. Multiple large mailings can fill up the directory on which mail is stored, so that no new mail can be received. If users are given individual mail quotas, their quotas can be consumed, so that they can not receive wanted mail. Repeatedly sending a message to a large list can consume large amounts of CPU time and other per-process resources. It is possible for users on many systems to mail very large files, or to write scripts to mail a large number of files in a very short time. The following example illustrates the use of management to avoid mail overload.

#### Popular Presidency

The United States White House has created e-mail addresses for the President and Vice President of the United States. This makes their system vulnerable to all sorts of excesses, both deliberate and accidental. Very few on-line users have to pay for electronic mail by the message, and a tremendous number of people are often urged to send mail to the President. Any world event could lead to a flood of messages. Furthermore, some people find it amusing to attempt to interfere with such a prominent site.

The White House has a security problem: prevention of denial of service for its mail system. Failure of the White House mail system could be cause embarrassing media coverage, and it could frustrate and anger voters. They must protect themselves from attempts to drown them in email, while continuing to allow any user on any site to get a message through. If one user on a large site is causing trouble, the White House probably does not want to cut off access to all other users on that site.

The White house has announced that their solution is to keep track of how much mail is coming from each site, and gradually throttle mail from troublesome sites. No site will be completely cut off, but no one site will be allowed to monopolize the resources. They have not announced the details of how they are throttling but they have guaranteed that all mail from every site will eventually get through. This is a beautiful example of quantitative access control.

## 3. Giving Us the Finger

Finger was designed to offer useful information in a friendly network. It can provide a match of real name to user name or vice versa. Finger provides standard services like information about when a user has last logged on, whether they have mail and whether they have read their mail. It also allows the user to provide optional information in a .plan file.

Finger presents both technical and social problems. On the technical side, as with the other services discussed above, finger allows remote users to take up resources at network sites. Some users will run a script that repeatedly, or even constantly, connects to your site, looking to see whether a particular user has logged on. This can eat up CPU and other per-process resources.

With a little user cooperation, finger can become a serious bandwidth problem. Users can create a .plan file to display information to any requester. There are not limits on the size or contents of the .plan file. A user who decides to include, for example, a nude made up of ASCII art, or a pornography story, and offer it to the world, can create a situation where the server will be swamped with requests. Users can also, at least on UNIX, exploit a feature called "named pipes" to get information about where the finger request is coming from, and to run a program whenever finger is run against their .plan file. Some users have devised ways to deliver large amounts of information, enough to affect network traffic bandwidth and CPU load.

### Finger is Too Helpful

The default UNIX finger , with no parameters, will offer a complete list of who is logged on, where they are logged on from, and whether they are idle. This is much too helpful [see "Unhelpfulness..."] and can pose serious security problems. Over time, a would-be intruder can build a complete picture of who is using the system, what their normal hours are, and when they are likely to be idle. This also give a would-be cracker a list of user names, whose passwords might be guessable (especially if user passwords contain information from their .plan file!). The intruder can also tell whether the system administrator is at work, at home, or idle. There are also some off-line considerations: knowing exactly when a user has logged off and might be about to leave the building could be useful to a stalker or mugger.

Finally, allowing users to offer data off-site, to any unverified requester, can have data security implications.

### Limiting Fingering

Many sites disable finger entirely as a risk, or disable viewing of user information files off-site. Other sites run modified programs that allow matching user name with real name and office phone number but give out no other information. Finger can be run from a wrapper that logs connections. However, finger is a useful program. It is useful for many sites to allow the outside world to be able to check user name - real name pairings. It is useful, perhaps, to see user activity in moderation. To preserve this useful service, the system needs a quantitative solution, which allows SOME access to user information, SOME number of accesses to each user's information, SOME amount of data to be sent in return.

## Questions for the designer

Designing a system that manages its time-and-space limited resources effectively requires consideration of appropriate sizing, priorities, limitation of access, detecting overuse, throttling, cutting off inappropriate or malicious use, etc. Many of these considerations require balancing the openness of the system against the effectiveness of the security controls. Users of a university system, especially faculty and research staff, may not tolerate mechanisms that restrict their use at all. Corporate and government users may tolerate more limitations, but probably not peacefully. Designers and policy-makers must consider the social questions as well as the technical issues. If the service definition for the system can be examined and defines, then the service/security tradeoffs can be made in a fairer, clearer, and less confining manner. Here are some of the resource management issues:

What services is the system providing? To whom? How much and at what rate?

What are the priorities of the system? What services and which users need to be guaranteed and which can be sacrificed when the system becomes overloaded?

What mechanisms does the system (usually the operating system) provide for measuring usage and detecting overload or probable attack conditions? What mechanisms does it have for detecting the source of an overload? Can it recognize that the overuse is from a particular user? A particular site on the network?

What mechanisms are available for slowing down or stopping service in times of overload or probable attack? Can these be applied to particular sources of trouble? Can they be applied to all services including those that do not require password login?

Can the system record usage for auditing? Is the audit and recording mechanism subject to overuse attacks (whether or not users can access it directly)? How does the system behave if the log file space fills up?

## Conclusions

Management policies for information need to concentrate on its propagation properties. Policies for managing limited resources need to define how these resources are allocated to users in a fair manner. This allocation is not a matter of yes-or-no privilege; it involves priorities, ordering, and quantity. The quantitative security policies must be supported by system mechanisms that can detect and respond to overloads in a consistent and secure manner.

Attacks on systems often exploit vulnerabilities caused by the system's finite resources. It is crucial that the security policies and models recognize this fact. These policies cannot be effective if they are limited to yes/no determination of privilege; they must consider the amount of usage and the allocation of time-limited resources among active users of a system. Too much of a good thing can be a security problem; overuse may signal a denial of service attack or a fishing expedition to gain information for breaking into the system.

It is important that security policies and mechanisms address both access to information and use of the physical resources of the system. The policies cannot be perfect or rigid; they must deal with complexities of system usage as well as more static characteristics of data or users. The mechanisms that detect problems and enforce security policies cannot be perfect either. As we have learned in intrusion detection, almost any pattern of unusual usage can represent a deliberate attack, unintentional misuse, or a legitimate but unanticipated use of the system. Our security concerns cause us to stop the offending usage, but sometimes we have to apologize later. With clear policies and effective mechanisms for quantitative control, we will be able to offer secure service, not just security of information. We will be able to explain rather than apologize.