Covert Distributed Processing with Computer Viruses

Steve R. White
IBM Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Abstract. *Computer viruses can be used by their authors to harness the resources of infected machines for the author's computation. By doing so without the permission or knowledge of the machine owners, viruses can be used to perform covert distributed processing. We outline the class of problems for which covert distributed processing can be used. A brute-force attack on cryptosystems is one such problem, and we give estimates of the time required to complete such an attack covertly.*

1. Introduction

Given the large aggregate computing power in the world, harnessing it to work on a single problem is an attractive idea. Systems which use the idle processing power of a collection of machines have been built, and shown to work well [1]. Previously, these have operated under the assumption that the owners of machines must give explicit permission for this use of their computing power. Computer viruses [2] can be inherently covert programs, which perform their actions without any explicit permission or awareness on the part of the owners of the machines that they use. This raises the possibility that viruses can be used for distributed processing tasks, perhaps in spite of the desires of the owners of the machines being used.

2. Covert Distributed Processing

Computer viruses can carry virtually any kind of task with them [3], [4]. In particular, they can use their ability to hide within innocuous programs to spread a distributed computing task among many users and many computers. Processes and information can be distributed unwittingly by users in the normal process of sharing other information. This paper gives an example of how a large number of computers can be harnessed as a distributed processor. This can be done covertly, without the explicit cooperation of those involved.

Our distributed processing virus is written to work on a part of the problem to be solved, and to create offspring viruses that also work on the problem. De-

pending upon the communication topology of the distributed system, and on the virus' knowledge of it, they could cooperate in any number of ways to solve the problem.

Here, though, we make the fewest possible assumptions about the virus' knowledge of the system's connectivity. We assume that a virus v can create a virus v', and that v can give information to v' about the progress that v has made on the solution of the problem. We do not assume that v and v' can communicate thereafter. The computational work is spread only as the viruses themselves spread.

With these communication assumptions, our distributed processing virus lends itself to working on tree-structured algorithms, in which offspring processes are initialized by parent processes, and work independently thereafter. This style of computation resembles the Unix[1] "fork" operation, with the restriction that the forked tasks have no intertask communication. Markov chain calculations are an example of a problem that fits this paradigm.

One of the goals of the virus is to obtain as much resource as possible to work on its problem (CPU time, disk space, etc.), while avoiding detection. The proper strategy will depend upon the details of the operating system used. In the type of single-tasking operating system used in most personal computers, for instance, the best strategy may be to "wake up" periodically and examine the state of the processor. If no active application task is running (i.e. the machine is idle), the virus task can be started. It would give up the processor as soon as any other system action began, to avoid detection due to slow system response. In a typical multi-tasking operating system, the virus task may run as a low-priority background task, and let the operating system handle this kind of resource allocation.

3. Covert Information Distribution

All viruses carry information with them, so designing a virus to distribute information covertly is straightforward. The information could simply be contained in the body of the virus, and each user whose system becomes infected could recover the information by peeking at the object code of the virus. Alternatively, the virus could respond to a particular keyboard sequence by displaying the information, so it is available more readily.

This technique can be used to retrieve results obtained by the various offspring viruses. Once a particular virus arrives at a result, it can spawn an "information carrying" virus that propagates through the distributed system. Eventually, it will reach the author of the covert task. It is possible to hide the result from owners of intermediate machines as well. This is done by having the virus that arrives at the result encrypt it with a public key before spawning the "information

[1] Unix is a trademark of AT&T.

carrying" virus. Since the author of the covert task can keep the private key secret, only the author can decrypt the result.

4. Example: Brute-Force Attack on a Cryptosystem

Brute-force attacks on cryptosystems are attacks that require very large computational resources to mount. One such attack involves work by seeing if random keys can decrypt known ciphertext to known plaintext [4]. Another involves factoring large numbers [5], [6]. Several such attacks on cryptosystems have been proposed. They generally involve a substantial cost and/or a substantial engineering effort to build special-purpose hardware.

The idea of using computer viruses for such an attack was first proposed by Quisquater and Desmedt [7]. They suggested that a virus could both guess keys at random, and spawn other viruses that do the same thing. From the preceding discussion, it is clear that such a calculation is within the paradigm of covert distributed processing.

We consider implementing this attack by writing a virus that spreads between small computers which are not under any central control. Unlike previous methods of implementing such an attack, this one can be done (a) at virtually no cost (other than the cost of developing the appropriate virus); and (b) with a strong expectation of anonymity, since it is very difficult to trace a virus back to its author in this environment.

We can get a (very rough) idea of a lower bound on the time required for such an attack by making some estimates. We assume that we are attacking DES, which has a key space of size 2^{56}, and that the typical machine is a fairly fast personal computer.

$$\text{(Typical software DES rate)} \simeq 6.3 \times 10^4 \text{ Bytes/sec}$$

$$\therefore \text{(Typical rate of keys tried on one machines)} \simeq 2.5 \times 10^{10} \text{ keys/year}$$

$$\text{(Number of machines)} \simeq 10^7$$

$$\therefore \text{(Average rate of keys tried on all machines)} \simeq 2.5 \times 10^{17} \text{ keys/year}$$

$$\text{(Average number of guesses required)} = 2^{56} \text{ keys} \simeq 6.4 \times 10^{16} \text{ keys}$$

$$\therefore \text{(Time required)} \simeq 0.26 \text{ years}$$

This is a rather severe underestimate. We have assumed that the virus will infect every machine, that the virus will have exclusive use of the machine all the time, that the time required for the virus to propagate is negligible, and that this is the only such virus in circulation. These assumption are likely to be off by at least four orders of magnitude.

Nonetheless, it is instructive to have come this close! And, these estimates are based on comparatively simple 1989 technology. As the aggregate computing power in the world increases dramatically over the next decade, this covert use of it may become more of a threat.

Acknowledgements

The author thanks Yvo Desmedt, William Arnold, Steve Weingart, Frederica Darema, and Kevin McAuliffe for useful conversations.

Bibliography

[1] J.F. Shoch, J.A. Hupp, "The 'Worm' Programs - Early Experience with a Distributed Computation," CACM 25 (March 1982) pp. 172-180

[2] F. Cohen, "Computer Viruses: Theory and Experiment," Computers & Security 6 (1987) pp. 22-35

[3] F. Cohen, "On the Implications of Computer Viruses and Methods of Defense," Computers & Security 7 (1988) pp. 167-184

[4] W. Diffie, M.E. Hellman, "Exhaustive Cryptanalysis of the NBS Data Encryption Standard," Computer, Vol. 10, No. 6 (June 1977) pp. 74-84

[5] T.R. Caron, R.D. Silberman, "Parallel Implementation of the Quadratic Sieve," J. Supercomputing 1 (April 1988) pp. 273-290

[6] A.K. Lenstra, M.S. Manasse, "Factoring By Electronic Mail," Proc. Eurocrypt '89, Houthalen, Belgium (April 10-12, 1989) In press

[7] J.-J. Quisquater, Y. Desmedt, "Watch for the Chinese Loto and the Chinese Dragon," informal paper, Crypto '87 (To be published)