Summary Part 1

Architectural Support for Copy and Tamper Resistant Software

Mrinal Nath (ID: 3307043)

February 14, 2005

What are the problems solved by this paper?

People can copy the Software Intellectual Property illegally by tracing the instructions and the data produced by copyrighted programs (assuming that the binary cannot be hacked). To prevent such IP rights violations, the authors are proposing architectural (hardware) support so that the instructions and data produced by a protected program are not visible to anyone in the outside world.

What are the approaches attempted by this paper?

The authors propose a machine called 'XOM' or Execute Only Memory. In this model, they do not trust any storage location outside the machine. So, if data and/or instructions are to be stored outside the machine, they are encrypted using cryptographic techniques. When these data or instructions are read into the machine they are suitably decrypted. They use the concept of *compartments* to isolate different 'principals' or processes that may be running concurrently on the machine. Each principal will have its session key, and thus no process can access the data and/or instructions of another process. (The session key itself is encrypted using asymmetric cipher while transmitting it to the machine. Each machine has its own private key embedded in the hardware, which is used to decrypt the asymmetric cipher). Each principal will have an identifier which is used to tag instructions and data belonging to that principal. To allow movement of data between processes, and also to allow a XOM program to be interrupted, the authors have proposed various instructions. These instructions vary greatly in their complexity and the amount of work that they do.

Security against spoofing and splicing attacks is guaranteed by the use of MAC which is dependent on the position of the data/instruction it encrypts. However, the proposed architecture is not immune to replay attacks.

A XVMM (XOM virtual machine monitor) is used to provide a suitable execution environment for the execution of the XOM code. To improve performance of the encryption/decryption processes, they propose that special hardware be built for these purposes.

What are the main conclusions of this paper?

Apparently, the overall execution time does not increase much (only about 5%). I find this result surprising, since the XOM requirements of encryption and decryption seem to require significant computation. Also, storing the MACs will cost memory space, and this overhead is not mentioned.

However, the XOM machine model is capable of preventing people from unauthorized access to code and data, thus preventing copying and/or tampering of software IP.