On Security in Capability-Based Systems*

Li Gong
University of Cambridge
Computer Laboratory
Cambridge CB2 3QG, England

November 1988

1 Introduction

Hogan in her recent paper [4] presented the requirements and the characteristics of operating systems to realize the principle of complete mediation. She states *"the principle of complete mediation requires that every access to every object be checked for authority. This implies that a secure system must utilize a foolproof method of identifying the source of every request"*. Wells argues in a later paper [13] that Hogan's discussion *"does not apply to contemporary capability-based system technology"*, and her statement is not true in such systems. Wells used the KeyKOS system [3,11] as an example.

In our opinion, Wells's argument holds for at most one special type of capability systems, which we refer as fully armed systems. In fact, one can argue that in KeyKOS the utilization of a foolproof mechanism for identifying the source is implicitly embedded throughout the system design, which is mixed with other issues and would naturally be more difficult to be foolproof. Moreover, it is hard to be convinced that an architecture like KeyKOS is suitable to support open system policy. A simple question would be who will maintain those *"discernible external communications"* [13] that run across geographical and organizational boundaries ? And do we trust them ? The aim of this note is to supply a more complete picture of security in capability-based systems.

2 Weakness in Classic Capability Systems

Boebert stated in [1] that an unmodified or classic capability system cannot enforce the $\star$-property in the Bell-LaPadula model [9] or solve the confinement problem [8]. The main pitfall of the classic capability system is that *"the right to exercise*

*To appear in the April issue of the *ACM Operating Systems Review*, 1989.

access carries with it the right to grant access". Therefore it is possible that a capability can propagate across a security domain without detection and cause an unauthorized access.

Concerning the underlying system architecture and its requirements for appropriate protection measures, there are mainly three types of capability systems.

3 Fully Armed Systems

In KeyKOS, *procession of a capability is sufficient authority to invoke the object it designates"* [13], and because capabilities are identity independent, i.e. whoever holds them has the right to use them, there is no need to identify the source of every request. However, this in turn requires for other ways to make sure that capabilities are in the right hands, if the system is meant to be secure in any sense. Since a capability is a string of bits, to prevent unauthorized propagation a solution of the kind in KeyKOS must ensure that a potential malicious user which in effect means any normal user must never see or manipulate capabilities and the system only accepts capabilities provided by some trusted hardware which cannot be tampered with. This can only be done in a hardware supported environment and the system cannot even be networked to other machines not of this nature. This approach has to securely manage the storing and the propagation of capabilities. It has to certify that there is no covet channel in the system design rather than in the access control policy which can later be modified relatively easily that can leak information about capabilities and there is no tampering with hardware including those external communication facilities. Because of all these, the complexity thus the difficulty would exceed that of supplying a simple independent source identifying mechanism that run on an appropriate server. The cost and the inability to networking will limit the applications of such designs. In particular, we think the KeyKOS approach that *"it is the connections you need to trust in a capability-based system, not the labels"* [11] is not suitable to the open system architecture. In this deemed to be hostile environment, the cost to maintain security would be too high, even if it is possible. In fact in dealing with connections, one has to consider the identities of the ends and to anticipate the possible connections. This would be naturally more difficult than dealing with identities directly.

4 Stand-Alone Systems

Although we agree that a fully armed system like KeyKOS is not impossible, one has to seek for other designs in a not fully armed system, normally a classic hardware supported stand-alone machine. In these systems, a user might be able to see or manipulate some capabilities which in effect means that the systems might also accept user supplied capabilities. This will cause many problems. For example, the *grant* and *take* capabilities in the Take-Grant model [12] can specify the possible capability propagations but cannot limit propagations because they

themselves are capabilities and can propagate in many ways without detection. Another example, the technique as the *unconfined right* in Hydra, which is meant to disable a capability propagation, has no effect. Some *soft* protection measures have to be implemented.

An effort was made in the SCAP design by Karger and Herbert [6,7] to support lattice security and access traceability. They took an approach where subjects can pass their capabilities freely as usual, but when a capability is used to request an access, the security kernel must check whether the access should be granted according to the security policy enforced. In other words, holding a capability is no longer both necessary and sufficient to access an object as in unmodified systems. It is now only necessary. The capability system at the lowest level supports the policy which is represented by access control list at a higher level, maybe outside of the kernel. Cache is used to reduce the number of policy checks.

5 Networked Systems

Things will be worse in a network environment since the system has to deal with requests from outside world which it may not trust. In a typical network capability system as Amoeba [10], capabilities are kept in user space. To prevents forgery capabilities are scrambled. This however does not limit propagation. It is clear that the principle in capability systems that whoever holds a capability has the right to use it must be accompanied by a method to identify the source of a request.

To control capability propagations, some kind of check against security policy has to be carried out somewhere, if not everywhere, in the life time of the capabilities. This is reflected in a taxonomy for capability systems [5]. Note to check policy implies to identify the source of a request. In theory Karger's approach can be adopted to a network system, i.e., when a request arrives, the kernel enters a trap to check the validity of the capability as well as the source identity to see whether the security policy would allow it. However, intuition tells that the number of capability propagations is usually much less than the number of their uses so that it seems more economic to check the security policy at propagation time than at access time; moreover, the real time response will be better if the security policy, which may be complex and expensive to check, is checked at propagation time. In some situations this can be done well in advance of access time. This observation leads to a completely different system, the ICAP system [2].

In ICAP, the identities are incorporated into capabilities by slightly changing the semantics of a classic capability. When a capability is to propagate, an *access control server* checks the security policy to see whether to allow it. The security policy is not checked when a capability is later used for access but the source has to be identified. An access control list is to support the capability system, which is the other way round in Karger's SCAP. Kain and Landwehr's taxonomy [5] has to be expanded in order to cover the ICAP design. Please refer to the concerned papers for design details and security and performance analysis, as well as the

various reasons of why the access control list is adopted. For example, access control list can support the notion of specific denial of access which capability systems apparently cannot.

6 Conclusion

We assert that unless in a fully armed system like KeyKOS where the manipulation of capabilities must be done by trusted hardware and must be invisible to any suspicious users or system components, in order to realize the principle of complete mediation identifying the source of every request is necessary. The security policy enforced has to be checked as well. This is demonstrated in the SCAP design [6,7]. Moreover, the approach in KeyKOS to deal with connections rather than end points may increase the complexity and difficulty to design the system and to certify that the system is secure. The cost may be too high to adopt this approach to the open system architecture.

In an open system architecture where requests from untrusted or even unknown remote machines must be dealt with, identifying the source would be very difficult. It might even be impossible because it is well known that some network addresses can be easily forged. In this case, to identify the identities and present them together with the capabilities is necessary. The ICAP design [2] which incorporates and integrates the identities into capabilities aims to counter this problem in a networked environment.

References

[1] W.E. Boebert, *On the Inability of An Unmodified Capability Machine to Enforce the $\star$-Property*, Proceedings of the 7th DoD/NBS Computer Security Conference, September, 1984.

[2] L. Gong, *An Identity-Based Capability System that Can Enforce Security Policies*, submitted to the 1989 IEEE Symposium on Security and Privacy, available from the author, October, 1988.

[3] N. Hardy, *KeyKOS Architecture*, ACM Operating Systems Review, Vol.19, No.4, October, 1985.

[4] C.B. Hogan, *Protection Imperfect: The Security of Some Computing Environments*, Operating Systems Review, Vol.22, No.3, July, 1988.

[5] R.Y. Kain and C.E. Landwehr, *On Access Checking in Capability-Based Systems*, IEEE Transactions on Software Engineering, Vol. SE-13, No.2, February, 1987.

[6] P.A. Karger and A.J. Herbert, *An Augmented Capability Architecture to Support Lattice Security and Traceability of Access*, Proceedings of the 1984 IEEE Symposium on Security and Privacy, April, 1984.

[7] P.A. Karger, *Improving Security and Performance for Capability Systems*, Ph.D. thesis, also available as Technical Report No.149, University of Cambridge Computer Laboratory, Oct., 1988.

[8] B.W. Lampson, *A Note on the Confinement Problem*, Communications of the ACM on Operating Systems, Vol.16, No.10, October, 1973.

[9] C.E. Landwehr, *Formal Models for Computer Security*, ACM Computing Surveys, Vol.13, No.3, September, 1981.

[10] S.J. Mullender, A.S. Tanenbaum, and R. van Renesse, *Using Sparse Capabilities in Distributed Operating System*, Proceedings of the 6th International Conference on Distributed Computing Systems, May 1986.

[11] S.A. Rajunas, N. Hardy, A.C. Bomberger, W.S. Frantz, and C.R. Landau, *Security in KeyKOS*, Proceedings of the 1986 IEEE Symposium on Security and Privacy, April, 1986.

[12] L. Snyder, *Formal Models of Capability-Based Protection Systems*, IEEE Transactions on Computers, Vol. C3, No.3, March, 1981.

[13] C. Wells, *A Note on "Protection Imperfect"*, Operating Systems Review, Vol.22, No.4, October, 1988.