Authorisation Subterfuge by Delegation in Decentralised Networks

Simon Foley and Hongbin Zhou

Department of Computer Science,
University College, Cork, Ireland.
s.foley@cs.ucc.ie, zhou@cs.ucc.ie

1 Introduction

Trust Management [1, 4, 10] is an approach to constructing and interpreting the trust relationships among public-keys that are used to mediate security-critical actions. Cryptographic credentials are used to specify delegation of authorisation among public keys. Existing trust management schemes are operational in nature, defining security in terms of specific controls such as delegation chains, threshold schemes, and so forth.

However, Trust Management approaches tend not to consider whether a particular authorisation policy is well designed in the sense that a principle cannot somehow bypass the intent of a complex series of authorisation delegations via some unexpected circuitous route. In an open system no individual has a complete picture of all the resources and services that are available. Unlike the administrator who 'sees everything' in a traditional closed system, the principals of an open system are often ordinary users and are open to confusion and subterfuge when interacting with resources and services. These users may inadvertently delegate un-intended authorisation to recipients.

In this paper we introduce the problem of *authorisation subterfuge*, whereby, in a poorly designed system, delegation chains that are used by principals to prove authorisation may not actually reflect the original intention of all of the participants in the chain.

2 Authorisation Subterfuge

2.1 Delegation of Authorisation

A simple model is used to represent delegation of authorisation between public keys. A signed credential, represented as $\{\!|\, K_B, p \,|\!\}_{K_A}$, indicates that K_A delegates to K_B, the authorisation permission p. Permissions are structured in terms of lattice $(PERM, \leq, \sqcap)$, whereby given $p \leq q$ means that permission q provides no less authorisation than p. A simple example is the powerset lattice of $\{\mathsf{read}, \mathsf{write}\}$, with ordering defined by subset, and greatest lower bound defined by intersection.

Given a credential $\{\!|\, K_B, q \,|\!\}_{K_A}$ and $p \leq q$ then there is an implicit delegation of p to K_B by K_A, written as $(\!|\, K_B, p \,|\!)_{K_A}$. Two reduction rules follow.

$$\frac{\{\!|\, K_B, p \,|\!\}_{K_A}}{(\!|\, K_B, p \,|\!)_{K_A}} \qquad \frac{(\!|\, K_B, p \,|\!)_{K_A}; p' \leq p}{(\!|\, K_B, p' \,|\!)_{K_A}}$$

If delegation is regarded as transitive, and if K_A delegates p to K_B, and K_B delegates p to K_C, then it follows that K_A implicitly delegates p to K_C.

$$\frac{(\!|\, K_C, p \,|\!)_{K_B}; (\!|\, K_B, p' \,|\!)_{K_A}}{(\!|\, K_C, p \sqcap p' \,|\!)_{K_A}}$$

This corresponds to SPKI certificate reduction [4] (greatest lower bound is equivalent to SPKI tuple intersection). It is not unlike a partial evaluation over a collection of KeyNote credentials, resulting in another credential. At this point, we do not consider permissions that cannot be further delegated.

2.2 Incompetence in Delegation

Bob (represented as public key K_B) holds credential $C_1 = \{\!|\, K_B, \texttt{buy100} \,|\!\}_{K_A}$ reflecting an authorisation to make purchases for his organisation (K_A) up to a limit of €100. Bob delegates this authority to another staff member Clare by signing $\{\!|\, K_C, \top \,|\!\}_{K_B}$, where $\top$ represents the highest value in the permission ordering and K_C is a public key owned by Clare. Bob believes that this is a reasonable strategy as,on the basis of his view of the world, certificate reduction gives $(\!|\, K_C, \texttt{buy100} \,|\!)_{K_A}$ (and no more).

In this case, Bob's delegation strategy is *incompetent* as in an open system it fails the principle of least privilege. Bob does not consider the possibility of the existence of other certificates, for example, $\{\!|\, K_B, \texttt{buy1000} \,|\!\}_{K_A}$ (where $\texttt{buy100} \leq \texttt{buy1000}$), indirectly authorising Clare for purchases up to €1,000 (and possibly unknown to Bob). Thus, when writing delegation certificates one must be careful to exactly specify the desired permission and no more. Bob should have signed $C_2 = \{\!|\, K_C, \texttt{buy100} \,|\!\}_{K_B}$. An implication of this is that a systematic naming scheme for permissions becomes critical in ensuring the principle of least privilege. For example, Keynote suggests a global registration scheme (for example, IANA/ICANN) to ensure uniqueness of permission attributes.

2.3 Confusion in Delegation

Continuing the example above, suppose that Bob also works for an organisation with public key K_Z and is unaware of the existence of the certificate $C_3 = \{\!|\, K_B, \texttt{buy100} \,|\!\}_{K_Z}$. Bob signs certificate $C_2 = \{\!|\, K_C, \texttt{buy100} \,|\!\}_{K_B}$, as recommended, believing that the resulting certificate chain (with $K_A \rightarrow K_B \rightarrow K_C$) provides the appropriate `buy100` authorisation for Clare (as an employee of K_A). Unknown to Bob, Clare could use certificate C_3 to provide an alternative chain $K_Z \rightarrow K_B \rightarrow K_C$ as proof of authorisation for `buy100`.

This confusion may introduce problems if the certificate chains that are used to prove authorisation are also used to determine who should be billed for the transaction. Bob believes the chain $K_A \rightarrow K_B \rightarrow K_C$ provides the appropriate

accountability for Clare's authorisation; his view of the world has not considered the existence of C_3. In this case we think of Bob as more *confused* in his delegation actions rather than incompetent; the permission naming scheme influences his local beliefs and it was the inadequacy of this scheme that led to the confusion. Perhaps Bob has too many certificates to manage and in the confusion looses track of which permissions should be associated with which keys.

Such inadequacies can lead to vulnerabilities when certificate chains are used to provide evidence of accountability for an authorisation. For example,

- K_Z, collaborating with K_C conceals $\{\!|\, K_B, \texttt{buy100} \,|\!\}_{K_Z}$ from K_B so that Clare can order from an unintended supplier (authorised via K_Z).
- K_Z conceals $\{\!|\, K_B, \texttt{buy100} \,|\!\}_{K_Z}$ from K_B and K_C and then, unknown to Clare, cut-and-pastes certificate chain $[C_1; C_2]$ from Clare to $[C_3; C_2]$.
- K_Z, collaborating with K_A conceals $\{\!|\, K_B, \texttt{buy100} \,|\!\}_{K_Z}$ to facilitate plausible deniability (for K_A) on the validity of an order. A third party cannot confirm the intent of the original delegation, viz, whether it should be billed to K_A or K_Z.

Certificate chains have been used in the literature to support degrees of accountability of authorisation, for example, [3, 8, 2]. The micro-billing scheme [3] uses KeyNote to help determine whether a micro-check (a KeyNote credential, signed by a customer) should be trusted and accepted as payment by a merchant. In [8], delegation credentials are used to manage the transfer of micropayment contracts between public keys; delegation chains provide evidence of contract transfer and ensure accountability for double-spending. These systems are vulnerable to authorisation subterfuge (leading to a breakdown in accountability) if care is not taken to properly identify the 'permissions' indicating the payment authorisations when multiple banks and/or provisioning agents are possible.

2.4 Dishonesty in Delegation

Bob has a legitimate expectation that so long as he delegates competently then he should not be liable for any confusion that is a result of poor permission design. Bob can use this view to act dishonestly. In signing a certificate he can always deny knowledge of the existence of other certificates and the inadequacy of permission naming in order to avoid accountability. While Bob secretly owns company K_Z, he claims that he cannot be held accountable for the 'confusion' when Clare (an employee of K_A) uses the delegation chain $K_Z \rightarrow K_B \rightarrow K_C$ to place her order.

3 What's in a Name?

A number of ad-hoc strategies can be used to avoid the problems of incompetence, confusion and dishonesty. One strategy would be to ensure that Bob had different roles (public keys) corresponding to the different organisations (K_A and

K_Z) that he works for. This assumes competence on Bob's part to ensure he is in the right role when delegating authorisation.

Another strategy would be to ensure that each permission is sufficiently detailed to avoid ambiguity. For example, including a company name as part of the permission $\{\!|\, K_B, \texttt{<OrgA:buy100>} \,|\!\}_{K_A}$ may help avoid the vulnerabilities in the particular example above. However, at what point can a principal be sure that a reference to a permission is sufficiently complete? Achieving this requires an ability to be able to fix a permission within a global context, that is, to have some form of global identifier and/or reference for the permission.

In addition to global uniqueness, it is preferable that permission identifiers also be *non-transient*. Including just a globally unique organisation name within a permission may not be sufficient. Organisation names and their ownership can change. For example, the domain name registration mishap concerning `panix.com` [11] may result in subterfuge when delegating permission `<panix.com:buy100>`.

In Trust Management frameworks such as KeyNote and SPKI/SDSI, public keys provide globally unique identifiers that are tied to the owner of the key. These can also be used to avoid permission ambiguity within delegation chains. For example, given $\{\!|\, K_B, \texttt{<}K_A\texttt{:buy100>} \,|\!\}_{K_A}$ there can be no possibility of subterfuge when Bob delegates authority to Clare with $\{\!|\, K_C, \texttt{<}K_A\texttt{:buy100>} \,|\!\}_{K_B}$. In this case the certificate makes the intention of the delegation very clear and provides accountability in the delegation chain for the authorisation held by Clare.

Needless to say that this strategy does assume a high degree of competence on Bob's part to be able to properly distinguish between permissions $\texttt{<}K_A\texttt{:buy100>}$ and $\texttt{<}K_Z\texttt{:buy100>}$, where, for example, each public key could be 342 characters long (using a common ASCII encoding for a 2048 bit RSA key). One might be tempted to use SDSI-like local names to make this task more manageable for Bob. However, in order to prevent subterfuge, permissions require a name that is unique across all name spaces where it will be used, not just the local name space of Bob. In Bob's local name space the permission `<(Bob's Alice):buy100>` may refer to a different Alice to the Alice that Clare knows.

Another possible source of suitable identifiers is a global X500-style naming service (if it could be built) that would tie global identities to real world entities, which would in turn be used within permissions. However, X500-style approaches suffer from a variety of practical problems [5] when used to keep track of the identities of principals. In the context of subterfuge, a principal might easily be confused between the (non-unique) common name and the global distinguished name contained within a permission that used such identifiers.

One practical difficulty when relying on public keys as global identifiers is that their use is often *transitory*. A public key serves as an identifier (for its owner) for as long as the key is regarded as valid. If the (private) key is compromised, or if the owner decides to re-key then authorisation certificates will have to be re-issued by all participants on delegation chains involving the permission. If K_A re-keys to K'_A, and issues a new certificate $\{\!|\, K_B, \texttt{<}K'_A\texttt{:buy100>} \,|\!\}_{K'_A}$ then Bob

(and everyone else) will have to issue new certificates $\{\!|\, K_B, \texttt{<}K'_A\texttt{:buy100>} \,|\!\}_{K_A}$, and so forth. This is contrary to the trust management strategy [10] whereby role memberships can be maintained independent of the permissions that are delegated to them. This contrasts with the use of X500-like global names. In this case, we assume that the name is non-transitory while the key is transitory. A re-keying results in the issuing of a new identity certificate. The owner uses their new key to re-issue existing authorisation certificates, whose permissions refer to the name of the principal rather than the public key. Other authorisation certificates signed by other principals remain valid as their permissions are based on non-transitory global names rather than transitory keys.

4 Conclusion

In this paper we described how poorly characterised permissions within cryptographic credentials can lead to authorisation subterfuge during delegation operations. This subterfuge results in a vulnerability concerning the accountability of the authorisation provided by a delegation chain: does the delegation operations in the chain reflect the true intent of the participants?

The challenge here is to ensure that permissions can be referred to in a manner that properly reflects their context. Since permissions are intended to be shared across local name spaces then their references must be global in nature. In the paper we discuss some ad-hoc strategies to ensure globalisation of permissions. In particular, we consider the use of global name services and public keys as the sources of global identifiers. In general we are interested in systematic ways of determining whether a particular delegation scheme using particular permissions is sufficiently robust to be able to withstand attempts at subterfuge. For the example above, Bob would like to know whether is is safe for him to delegate permission `<panix.com:buy100>` to Clare.

Trust Management, like many other protection techniques, provide operations that are used to control access. As with any protection mechanism the challenge is to make sure that the mechanisms are configured in such a way that they ensure some useful and consistent notion of security. We would like some assurance that a principal cannot bypass security via some unexpected but authorised route. It is argued in [6] that verifying whether a particular configuration of access controls is effective can be achieved by analysing its consistency, that is, whether it is possible for a malicious principle to interfere with the the normal operation of the system. This type of analysis [7, 9] is not unlike the analysis carried out on authentication protocols. In the case of mechanisms based on trust management it is a question of determining consistency between potential delegation chains. Developing a suitable verification framework for the consistency of delegation chains is a topic of ongoing research [12].

5 Acknowledgements

This work is supported by the UCC Centre for Unified Computing under the Science Foundation Ireland WebCom-G project and by Enterprise Ireland Basic Research Grant Scheme (SC/2003/007).

References

1. M Blaze et al. The keynote trust-management system version 2. September 1999. Internet Request For Comments 2704.
2. M. Blaze, J. Ioannidis, S. Ionnidis, A. Keromytis P. Nikander, and V. Prevelakis. Tapi: Transactions for accessing public infrastructure. submitted for publication, 2002.
3. Matt Blaze, John Ioannidis, and Angelos D. Keromytis. Offline micropayments without trusted hardware. In *Financial Cryptography*, Grand Cayman, February 2001.
4. C Ellison et al. SPKI certificate theory. September 1999. Internet Request for Comments: 2693.
5. C.M. Ellison. The nature of a usable PKI. *Computer Networks*, 31:823–830, 1999.
6. S.N. Foley. Evaluating system integrity. In *Proceedings of the ACM New Security Paradigms Workshop*, 1998.
7. S.N. Foley. A non-functional approach to system integrity. *Journal on Selected Areas in Communications*, 21(1), Jan 2003.
8. S.N. Foley. Using trust management to support transferable hash-based micropayments. In *Proceedings of the 7th International Financial Cryptography Conference*, Gosier, Guadeloupe, FWI, January 2003.
9. S.N. Foley. Believing in the integrity of a system. In *IJCAR Workshop on Automated Reasoning for Security Protocol Analysis*. Springer Verlag Electronic Notes in Computer Science, 2004.
10. R Rivest and B Lampson. SDSI - a simple distributed security infrastructure. In *DIMACS Workshop on Trust Management in Networks*, 1996.
11. T. Zeller. New York Times, January 18 2005.
12. H. Zhou and S.N. Foley. A logic for analysing authorisation subterfuge in delegation chains. In *Submitted for publication*, 2005.