Proc. 3rd RADC Workshop on Multilevel Database Security,
Castille, New York, June 1990, pages 7-12.

RESTRICTED POLYINSTANTIATION

or

How to Close Signaling Channels Without Duplicity

Ravi Sandhu and Sushil Jajodia*†*

Department of Information Systems
and Systems Engineering
George Mason University
Fairfax, VA 22030-4444

Abstract. We dispel the mistaken notion that polyinstantiation in multilevel secure databases amounts to lying and/or instilling confusion about the "true" values of data. On the contrary we show it is easy to polyinstantiate and be truthful, provided one is disciplined and sensible about it. Our conclusion is that polyinstantiation is effective if properly used, but it can be terribly misused if so desired. In this respect polyinstantiation is no different than most other useful mechanisms in computer systems. We also point out that the necessary discipline can be enforced using standard integrity concepts such as well-formed transactions, least privilege and strong discretionary access controls.

1 INTRODUCTION

What distinguishes a multilevel database from ordinary single level ones? In a multilevel world as we raise a user's clearance new facts emerge; conversely as we lower a user's clearance some facts get hidden. Therefore users with different clearances see different versions of reality. Moreover, these different versions must be kept coherent and consistent—both individually and relative to each other—without introducing any downward signaling channels.

*Both authors are also affiliated with the Center for Secure Information Systems at George Mason University.

†Sushil Jajodia is also affiliated with the Secure Technology Center, The MITRE Corporation, 7525 Coleshire Drive, McLean, VA 22102

The caveat of "no downward signaling channels" poses the major new problem in building multilevel secure database management systems (DBMSs) as compared to ordinary single-level DBMSs. This caveat is inescapable and absolute. We must reject outright "solutions" which tolerate signaling channels. Solutions with signaling channels, such as proposed in [1, 16], may well be acceptable as an engineering compromise in particular situations. But they are clearly not acceptable as general-purpose solutions. This point needs to be emphasized because security is usually the one to take the first hit in engineering trade-offs. It behooves us as security researchers to present solutions which avoid taking this hit while at the same time providing

- intuitively reasonable, practically useful, formally simple and complete update semantics,
- consistency and integrity of the database both within and across levels,
- flexibility for application semantics, and
- fine-grained classification of data, i.e. element-level labeling.

Practical considerations also dictate that all this be implemented with minimal trusted code.

The central point of this paper is to demonstrate how these diverse goals can be met in a multilevel relational DBMS without compromising security as part of the bargain. This is admittedly a difficult problem and one which has received a great deal of attention in recent times. Our proposal of restricted polyinstantiation, described in section 3, is simple in concept and almost obvious in retrospect. For the most part it uses standard concepts from the database arena. The one additional concept we need is polyinstantiation. Let us therefore begin by reviewing this concept. We assume the reader is familiar with basic relational notions and terminology.

2 POLYINSTANTIATION

The concept of polyinstantiation was explicitly introduced by Denning et al [2], although the roots of the idea can be traced back to Hinke-Schaefer [10] and perhaps earlier. Unfortunately SeaView researchers, in their formal articulation of this concept [2, 3, 4, 11], bundled in some unnecessary baggage [5] which has obscured the fundamental simplicity of this concept. Another unfortunate aspect of SeaView is the lack of a formal update semantics, recent attempts notwithstanding [12]. This has led to identification of some bizarre scenarios which have mistakenly been taken to be intrinsic to polyinstantiation [16, 17].

Fortunately the security community's understanding of polyinstantiation has advanced dramatically since our initial identification of the shortcomings of SeaView's

formal definitions in [5]. Our contributions to this progress have been described at considerable length and with the utmost formalism and rigor in [5, 6, 7, 8, 14]. The main points of these papers are briefly summarized below.

- The unnecessary baggage in SeaView's definition of polyinstantiation, *viz.* the multivalued dependency (mvd) component of polyinstantiation integrity (PI), should be dropped because it results in spurious tuples [5]. Instead PI should be defined to consist only of the functional dependency (fd) component. The fd requirement amounts to stating that the actual key of a multilevel relation is the apparent key (consisting only of data attributes) extended with all the classification attributes.

- For purpose of flexibility additional PI constraints should be supported by the DBMS on a relation-by-relation and database-by-database basis; perhaps even on a tuple-by-tuple basis. These additional constraints might include the one tuple per tuple-class concept of [9, 14], the interpreted propagation notion of [8], the original SeaView mvd-PI [11], the more recent dynamic mvd-PI of [12], and so on. These should not however be embedded in the data model and imposed uniformly on every application. Only the fd component of PI should be so hardwired into the DBMS and required of all applications.

- The operational semantics for update operations on multilevel relations should be as close to standard SQL as possible. Moreover, an update should result in polyinstantiation only when absolutely required for closing downward signaling channels (or optionally for deliberately establishing cover stories) and the fewest possible tuples should be introduced in such cases. The minimal propagation semantics given in [8] achieves these objectives. Other update semantics such as the interpreted propagation semantics of [8], one tuple per tuple-class of [9, 14], the original SeaView mvd-PI [11] and the dynamic mvd-PI of [12] should be made available as options to be used by the Database Administrator; but should certainly not be embedded as a fundamental property of a data model.

To summarize in a nutshell our position is that the data model for multilevel relations should be kept as simple and free of unnecessary constraints as possible, while allowing maximum flexibility to the Database Administrator in stating additional requirements to capture application semantics.

2.1 The Source of Polyinstantiation

Having made these general observations let us now forget about all this marvelous theory and formalism summarized above. Instead let us consider by means of examples how poyinstantiation arises and therefore how it might be controlled. There are basically two ways that polyinstantiation can occur.

1. *Polyhigh* occurs when a high user* attempts to insert data in a field which already contains low data. Overwriting the low data in place will result in a signaling channel. Therefore the high data can be inserted only by creating a new instance of the field to store the high data. We also have the option of rejecting the update altogether with the attendant possibility of denial-of-service to the high user.

2. *Polylow* occurs in the opposite situation where a low user attempts to insert data in a field which already contains high data. In this case rejecting the update is not a viable option because it establishes a signaling channel. That leaves us two alternatives. We can overwrite the high data in place which violates the integrity of the high data. Or we can create a new instance of the field to store the low data.

In both cases note that there are alternatives to polyinstantiation. Unfortunately the alternatives are not acceptable as general solutions. Both alternatives are secure in the sense of secrecy and information flow. However the alternative to polyhigh entails denial-of-service to high users by low users, while the alternative to polylow entails destruction of high data by low users which presents a serious integrity problem. Clearly a general purpose solution must allow both polyhigh and polylow to occur with the option of turning to the respective alternative in the context of specific applications. Let us now consider a concrete example to make this point clearer.

2.2 Polyhigh Example

Consider the following relation SOD where Starship is the apparent primary key.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	null	U	U

Here, as in all our examples, each attribute in a tuple not only has a value but also a classification. In addition there is a tuple-class or TC attribute. This attribute is computed to be the least upper bound of the classifications of the individual data elements in the tuple.

Now consider the following scenario.

1. A U-user updates the destination of the Enterprise to be Talos. The relation is therefore modified as follows.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U

*Strictly speaking we should be saying subject rather than user. For the most part we will loosely use these terms interchangably. Where the distinction is important we will be appropriately precise.

2. Next a S-user attempts to modify the destination of the Enterprise to be Rigel. We cannot overwrite the destination in place because that would create a downward signaling channel. We can reject the update at the risk of denying entry of legitimate secret data. Or we can polyinstantiate and modify the relation to appear as follows, respectively for U and S users. Note that U users see no change.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U
Enterprise	U	Exploration	U	Rigel	S	S

What are we to make of this last relation given above. There are at least two reasonable interpretations.

- The destination of Talos may be a cover story for the real destination of Rigel (in which case we are merely accurately mimicking the duplicity of the real world within the database).*
- Alternately we have a temporary inconsistency in the database which needs to be resolved. For example the S-user who inserted the Rigel destination may latter login at the U level and nullify the Talos value, so thereafter the relation appears respectively as follows to U and S users.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	null	U	U

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Rigel	S	S

  It is most important to understand that this scheme does not create a downward signaling channel from one process to another. The nullification of the destination at the U level is being done by a U subject. One might argue that there is a downward signaling channel with a human in the loop. The human is however trusted not to let the channel be exercised without good cause.

  Note that the U-user who executed step 1 of the scenario may again try to enter Talos as the destination which brings us within the scope of polylow.

*We know there are other ways of incorporating cover stories. For example we may have two attributes, one for cover-story destination and one for the real destination. Debate on the relative merits and demerits of these techniques is outside the scope of this paper.

Let us reiterate that we can leave the relation as shown after step 1 unchanged and simply reject the update of step 2. This is a reasonable option which should be supported by the DBMS at the risk of denial-of-service. However we see no reason to force this to be the only option for all applications. DBMSs must be flexible and must cater to situations where polyinstantiation with temporary inconsistency is preferred to denial-of-service.

2.3 Polylow Example

Our example for polylow is similar to the polyhigh example with the difference that the two update operations occur in the opposite order. So again consider the following relation SOD where Starship is the apparent primary key.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	null	U	U

This time consider the following scenario.

1. A S-user modifies the destination of the Enterprise to be Rigel. The relation is modified to appear respectively as follows to U and S users. Note that U-users see no change in the relation.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	null	U	U

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Rigel	S	S

2. A U-user updates the destination of the Enterprise to be Talos. We cannot reject this update on the grounds that a secret destination for the Enterprise already exists, because that amounts to establishing a downward signaling channel. We can overwrite the destination field in place at the cost of destroying secret data. This would give us the following relation for both U and S users.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U

   For obvious reasons this alternative has not been seriously considered by most researchers. That leaves us the option of polyinstantiation which will modify the relation at the end of step 1 to the following for U and S users respectively.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U
Enterprise	U	Exploration	U	Rigel	S	S

This is exactly the same relation as obtained at the end of step 2 in our polyhigh example. The possible interpretations are therefore similar, i.e., we either have a temporary inconsistency or a cover story. The temporary inconsistency can be fixed by having a U subject (possibly created by a S user looged in at the U level) nullify the Talos destination. But the inconsistency may recur again and again.

3 RESTRICTED POLYINSTANTIATION

In the previous section we have examined the source of polyinstantiation and identified polyhigh and polylow as the two different ways in which polyinstantiation manifests itself. In this section we consider applications which have the following requirements.

- Downward signaling channels cannot be tolerated.
- Temporary inconsistencies cannot be tolerated.
- Denial of data entry service to high users cannot be tolerated.

Moreover each of these requirements has equal weightage and one cannot be sacrificed for another. The scenarios of the polyhigh and polylow examples of the previous section show that polyinstantiation by itself cannot meet these requirements simultaneously. One requirement or the other must give in some way.

In this section we describe a slight twist to the concept of polyinstantiation and show how all three requirements identified above can be simultaneously met. Our extension to polyinstantiation is called *restricted polyinstantiation*. To simplify the discussion let us also assume there are no cover stories, or equivalently, cover stories are not to be incorporated by polyinstantiation.

The basic idea is to introduce a special symbol denoted by "restricted" as the possible value of a data element. The value "restricted" is distinct from any other value for that element and is also different from "null." In other words the domain of a data element is its natural domain extended with "restricted" and "null." In accordance with our previous practise "null" is always classified at the level of the apparent key in a tuple. The value "restricted" may however be classified at any level which dominates the level of the apparent key.

Let us now play out the polyhigh and polylow scenarios of the previous section.

3.1 Polyhigh Example Revisited

Consider the following relation SOD where Starship is the apparent primary key.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	null	U	U

Consider the following scenario.

1. A U-user updates the destination of the Enterprise to be Talos. The relation is therefore modified as follows.

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	Talos	U	U

2. Next a S-user attempts to modify the destination of the Enterprise to be Rigel. We cannot polyinstantiate even temporarily, so we must reject this update. Do we have denial-of-service to the S-user? No, because the S-user can obtain service as follows.

   Step 2a. The S-user first logs in as a U-subject and marks the destination of the Enterprise as restricted giving us the following relation.*

Starship		Objective		Destination		TC
Enterprise	U	Exploration	U	restricted	U	U

   The meaning of restricted is that this field can no longer be updated by a U user. U users can therefore infer that the true value of Enterprise's destination is classified at the S level (assuming there are only two levels).

   Note that the signaling channel introduced by this mechanism is very similar to the one resulting from the nullification of Talos at the U-level in the example of section 2.2. Both involve a trusted S-user in the loop who presumably will ensure that the channel is not exercised wantonly, but rather that this inference is permitted only when the real world situation is actually so. Such a channel with trusted humans in the loop can be exercised only by Trojan Horses who are capable of manipulating the real world. This entails the manipulation of real trusted people making real decisions and not merely the manipulation of bits in a database.

   Step 2b. The S-user then logs in as a S-subject and enters the destination of the Enterprise as Rigel giving us the following relations at the U and S levels respectively.

*Alternately the S-user logs in at the U-level and requests some properly authorized U-user to carry out this step. Communication of this request from the S-user to the U-user may also occur outside of the computer system, by say a telephone call.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	restricted U	U

Starship	Objective	Destination	TC
Enterprise U	Exploration U	restricted U	U
Enterprise U	Exploration U	Rigel S	S

How does this differ from the scenario of section 2.2? The main difference is that U-users are no longer able to update the destination of the Enterprise.† In particular, attempts by U-users to reenter Talos as the destination of Enterprise will be rejected on the grounds that the field is restricted. Therefore the relation can be guaranteed to be consistent till such time as the restricted value is eliminated. Consideration of who should be allowed to enter and remove the restricted value is deferred till section 4.

3.2 Polylow Example Revisited

Now consider the two update operations in the opposite order. So again we begin with the following relation SOD where Starship is the apparent primary key.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	null U	U

This time consider the following scenario.

1. A S-user modifies the destination of the Enterprise to be Rigel. This update is rejected! Instead the S-user is asked to go through steps 2a and 2b of section 3.1 giving us the following relations at the U and S levels respectively.

Starship	Objective	Destination	TC
Enterprise U	Exploration U	restricted U	U

Starship	Objective	Destination	TC
Enterprise U	Exploration U	restricted U	U
Enterprise U	Exploration U	Rigel S	S

2. A U-user updates the destination of the Enterprise to be Talos. The update is rejected on the grounds that the field is restricted.

†If we so choose the tuple with the restricted value can be automatically eliminated at the S level. Doing so will highlight the close connection between this situation and the relations resulting after the Talos destination has been nullified to restore integrity in the scenario of section 2.2.

Note that there is no denial-of-service to the S-user. What is happening is a denial of improper service. The denial-of-service to the U-user is only appropriate in this situation.

4 APPROPRIATE ASSURANCE

We can summarize the previous section as requiring the following protocol for insertion of high data.

> *Prevent Protocol.* Data with classification C can be entered by a C-subject in attribute A_i for tuple t with apparent key AK and key class CK only if for all $CK \leq C' < C$ there exists a tuple t' such that $t'[AK, CK] = t[AK, CK]$,$t'[A_i] =$ restricted, and $t'[C_i] = C'$. In other words the attribute A_i in tuple t is restricted at all levels below C where t is visible.

We call this the prevent protocol because it prevents polyinstantiation due to either polyhigh or polylow from occurring.

The question arises of how do we enforce this protocol. The above protocol adheres to both simple security and the $\star$-property. It can therefore be enforced to A1 assurance standards without the use of trusted subjects.

There are however additional components to this protocol which need to be specified and enforced. Note that assigning a restricted value to an attribute with classification C is dangerous in that it can cause denial of service to C-users and loss of integrity in the database. So when the destinations of all our flights are made restricted, when they should not be, we might end up grounding the entire fleet! Therefore the ability to mark a filed as restricted should be a carefully controlled privilege. But note that this problem essentially exists whether or not we recognize restricted as a special value. For suppose a malicious program running at the U level, and obeying simple security and $\star$-property, sets the destination of all flights to be Dayton, Ohio. Does the entire fleet converge on Wright Patterson Air Force Base?

Clearly in order to maintain integrity we need to control updates, not only across security levels, but also within a single security level. Conventional single-level DBMSs have addressed the integrity problem using concepts such as well-formed transactions, least privilege and strong discretionary access controls. Although existing systems have major shortcomings [13, 15] they nevertheless have a number of mechanisms which are used to build integrity into one's applications.

Our use of restricted does however have a special characteristic which deserves mention. Once a field has become restricted there must be some way it can be made unrestricted in future. Now, within the confines of the $\star$-property, a restricted value at the U level can be made unrestricted only by a U-subject. Clearly if this privilege is not carefully controlled then we will continue to have the scope for the

polylow variation of polyinstantiation. That is restricted can be nullified and then some actual value can be entered.

It follows that we must strictly control the ability to nullify a restricted value or enter a restricted value. It is beyond the scope of this paper to discuss the details of how this might be done. However there are solutions whose assurance can span the scale from A1 quality assurance to absolutely no assurance. For instance suppose that a restricted field can be made unrestricted only by use of a trusted path and only by a special subject. Clearly this approach can have A1 assurance.

5 CONCLUSION

In this paper we have shown how both the polyhigh and polylow variations of polyinstantiation can be eliminated by our solution of restricted polyinstantiation. This allows us to avoid downward signaling channels, temporary inconsistencies, denial of data entry to high users and the overwriting of high data by low subjects. This is the first solution to meet all these requirements simultaneously.

In conclusion we wish to note that restricted polyinstantiation makes a particular trade-off among conflicting objectives. It may be eminently suitable to most applications. Yet we would advise against having this as the only option. Databases are long lived and develop a great deal of inertia over their life. Moreover different applications may call for different trade-offs. For example temporary inconsistencies may be preferred to inconvenience in data entry. Our general purpose multilevel secure DBMSs must cater to such applications too. Therefore our recommendation is that restricted polyinstantiation be available as one of several options that a multilevel secure DBMS supports.

Acknowledgement

We are indebted to John Campbell, Joe Giordano, and Howard Stainer for their support and encouragement, making this work possible. The opinions expressed in this paper are of course our own and should not be taken to represent the views of these individuals.

References

[1] Burns, R.K. "Referential Secrecy." *IEEE Symposium on Security and Privacy*, Oakland, California, May 1990, 133-142.

[2] Denning, D.E., Lunt, T.F., Schell, R.R., Heckman, M., and Shockley, W.R. "A Multilevel Relational Data Model." *IEEE Symposium on Security and Privacy*, 220-234 (1987).

[3] Denning, D.E., Lunt, T.F., Schell, R.R., Shockley, W.R. and Heckman, M. "The SeaView Security Model." *IEEE Symposium on Security and Privacy*, 218-233 (1988).

[4] Denning D.E. "Lessons Learned from Modeling a Secure Multilevel Relational Database System." In *Database Security: Status and Prospects,* (Landwehr, C.E., editor), North-Holland, 35-43 (1988).

[5] Jajodia, S. and Sandhu, R.S. "Polyinstantiation Integrity in Multilevel Relations." *IEEE Symposium on Security and Privacy*, Oakland, California, May 1990, 104-115.

[6] Jajodia, S. and Sandhu, R.S. "A Formal Framework for Single Level Decomposition of Multilevel Relations." *IEEE Workshop on Computer Security Foundations*, Franconia, New Hampshire, June 1990, 152-158.

[7] Jajodia, S. and Sandhu, R.S. "Polyinstantiation Integrity in Multilevel Relations Revisited." *IFIP WG11.3 Workshop on Database Security* (1990). To be published as *Database Security IV: Status and Prospects*, (Jajodia, S. and Landwehr, C.E., editors), North-Holland.

[8] Jajodia, S., Sandhu, R.S. and Sibley, E. "Update Semantics for Multilevel Relations." *Sixth Annual Computer Security Applications Conference*, Tucson, Arizona, December 1990, to appear.

[9] Haigh, J.T., O'Brien, R.C. and Thomsen, D.J. "The LDV Secure Relational DBMS Model." *IFIP WG11.3 Workshop on Database Security* (1990). To be published as *Database Security IV: Status and Prospects*, (Jajodia, S. and Landwehr, C.E., editors), North-Holland.

[10] Hinke T.H. and Schaefer M. "Secure Data Management System." Technical Report RADC-TR-75-266, System Development Corporation (1975).

[11] Lunt, T.F., Denning, D.E., Schell, R.R., Heckman, M. and Shockley, W.R. "The SeaView Security Model." *IEEE Transactions on Software Engineering*, 16(6):593-607 (1990).

[12] Lunt, T. and Hsieh, D. "Update Semantics for a Multilevel Relational Database." *IFIP WG11.3 Workshop on Database Security* (1990). To be published as *Database Security IV: Status and Prospects*, (Jajodia, S. and Landwehr, C.E., editors), North-Holland.

[13] Sandhu, R.S. "Mandatory Controls for Database Integrity." *Proceedings of the Third IFIP WG 11.3 Workshop on Database Security*, Monterey, California, September 1989. Published as *Database Security III: Status and Prospects*, (Spooner, D.L. and Landwehr, C.E., editors), North-Holland, pages 143-150.

[14] Sandhu, R.S., Jajodia, S. and Lunt, T. "A New Polyinstantiation Integrity Constraint for Multilevel Relations." *IEEE Workshop on Computer Security Foundations*, Franconia, New Hampshire, June 1990, 159-165.

[15] Sandhu, R.S. and Jajodia, S. "Integrity Mechanisms in Database Management Systems." *13th NIST-NCSC National Computer Security Conference*, Washington, D.C., October 1990, 526-540.

[16] Wiseman, S.R. "On the Problem of Security in Data Bases." In *Database Security III: Status and Prospects*, (Spooner, D.L. and Landwehr, C.E., editors), North-Holland, pages 143-150 (1990). Also available as Royal Signal and Radar Establishment, U.K., Memo 4263.

[17] Wood, A.W. "The SWORD Model of Multilevel Secure Databases." Royal Signal and Radar Establishment, U.K., Report 90008, June 1990.