Conduit Cascades and Secure Synchronization

Simon N. Foley,
Department of Computer Science,
University College, Cork, Ireland.
(s.foley@cs.ucc.ie)

Abstract

Synchronizing Personal Digital Assistants with host systems can result in indirect accesses that bypass security requirements. In this paper we propose a framework for analyzing the security vulnerabilities that can arise from synchronization. This framework provides us with the basis of a paradigm for analyzing the access-control vulnerabilities of systems comprised of secure and non-secure components.

1 Introduction

Personal Digital Assistants (PDAs) such as the Palm handheld are small hand-held computing devices that support a variety of applications, ranging from conventional electronic organizer programs to spreadsheets, electronic mail and web browser clients. A PDA is commonly viewed as an extension of a user's workstation (or server); carrying data and programs that often mirror data and programs from the workstation. Synchronization between the workstation and the PDA is performed on a regular basis, ensuring that changes made to data stored on the PDA are reflected on the workstation, and vice-versa.

Little consideration has been given to the security policy implications of using these devices as part of an application system. While PDAs are typically single-user systems supporting little or no access-control, they are expected to synchronize with multi-user host systems that do have access-control requirements. This synchronization may be used to bypass host system access-controls.

For example, an employee working in sales and engineering departments is subject to the security requirement that sales data may not be written to engineering datasets. If we are not confident about the employee's PDA upholding this requirement then synchronization must ensure that at any one time, either sales or engineering information is carried on the employee's PDA, but not both. Other scenarios are possible, for example, the PDA carries both engineering and sales datasets for information purposes. However, only sales data can be two-way synchronized with the host system.

In this paper we consider the analysis of access-control vulnerabilities that can arise from synchronizing host systems with PDAs. The approach first considers our confidence in the access constraints of the individual components and then analyzes whether that confidence can be maintained when the components synchronize. While a component such as a Palm does not have an access-control mechanism, we can still specify, albeit with low confidence, the access limitations that we believe the installed software implicitly provides. Our framework provides us with the basis of a paradigm for analyzing the security vulnerabilities of systems comprised of secure and non-secure components.

Arbitrary access policies can be abstractly represented in terms of directed graphs [7] or as reflexive orderings [5]. We use reflexive orderings to represent these policies and Section 2 provides some notation from [5] that is useful for specifying and reasoning about such policies. Section 3 extends these policies to include ratings that represent the degree of confidence in the policy being upheld. Sections 4 and 5 consider the additional accesses that can arise as a result of synchronization. A cascading effect can arise with multiple synchronization which we show to be a generalization of the network cascade vulnerability problem [10, 12].

The Z notation [14] is used to provide a consistent syntax for structuring and presenting the definitions and examples in this paper. We use only those parts of Z that can be intuitively understood and Appendix A gives a brief overview of the notation used.

2 Security Policies

Every system entity (principal, subject, object, etc.) is assumed to have an associated security label that encodes its security relevant characteristics. Labels may simply represent sensitivity levels such as $\mathsf{unclass}$ and secret, but they may also represent any security-relevant attribute, for example, a label representing sales information. Given a set of labels L then a security policy is defined as a reflexive relation $P : L \leftrightarrow L$. If $a \mapsto b \in P$ then information of type a may flow/interfere with information of type b. For example, sales information may flow (be read by) the program labeled budgets. In this paper we are not concerned with what is meant by information flow or interference: we use the flow relation as a simple abstraction of the security policy upheld by a system. It has been shown elsewhere [5] that this abstraction is expressive and can be used to characterize a wide variety of security policies, including Chinese Walls, Clark-Wilson access triples and user-group polices.

A standard Palm handheld does not have an access control mechanism. However, we can use a flow policy to represent the access limitations that we believe the installed software provides. For example, on a standard Palm, we are reasonably confident that the Giraffe game does not interfere with the mail database. Naturally, our confidence that the Palm will maintain this policy is far less than our confidence that a multilevel secure system can uphold a comparable policy.

2.1 Specifying Flow Policies

The set of all flow policies between labels of (generic) type L is defined by $\mathcal{R}[L]$, the set of all reflexive relations.

$$\mathcal{R}[L] == \{R : L \leftrightarrow L \mid \mathrm{id}(\mathrm{dom}\, R \cup \mathrm{ran}\, R) \subseteq R\}$$

Let the *alphabet* αR of policy R denote the set of labels that it is defined in terms of $(\mathrm{dom}\, R)$.

A flow policy may be specified using the $\leadsto$ operator: $A \leadsto B$ defines a policy such that all elements of A may flow to all elements of B. Relations $\bot A$ and $\top A$ define the least and most restrictive flow policies with alphabet A, that is, $\bot A$ permits all flows, while $\top A$ does not permit any flows (other than reflexivity).

$$
\begin{array}{|l}
[L] \\
\hline
\_ \leadsto \_ : ((\mathbb{P}\, L) \times (\mathbb{P}\, L)) \rightarrow \mathcal{R}[L] \\
\bot, \top : (\mathbb{P}\, L) \rightarrow \mathcal{R}[L] \\
\hline
A \leadsto B = \mathrm{id}\, A \cup \mathrm{id}\, B \cup (A \times B) \\
\bot A = A \times A \\
\top A = \mathrm{id}\, A \\
\hline
\end{array}
$$

Example 1 We are reasonably confident that the standard software installation on our Palm upholds the policy GPALM.

$$
\begin{aligned}
\mathsf{GPALM} &== \top\{\mathsf{giraffe}, \mathsf{email}\} \\
\mathsf{PALM} &== \{\mathsf{email}\} \leadsto \{\mathsf{abacus}\} \\
\mathsf{MLS} &== \{\mathsf{unclass}, \mathsf{secret}\} \leadsto \{\mathsf{secret}, \mathsf{topsecret}\}
\end{aligned}
$$

Policy PALM specifies that email information may flow to the (Abacus) spreadsheet database (but not vice-versa); MLS specifies the usual multilevel security policy. $\triangle$

2.2 A Policy Algebra

Reflexive policies may be constructed using the usual set and relation operators (set comprehension, union, and so forth). In this section an algebra is described that is useful for the specification of information flow policies.

The projection operator projects a relation R into relation $R@A$ with alphabet $\alpha R \cap A$ such that the relationships of R are preserved, for example, $\mathsf{mls}@\{\mathsf{secret}, \mathsf{topsecret}\} = \{\mathsf{secret}\} \leadsto \{\mathsf{topsecret}\}$. The policy extension operator extends R into a relation $R \uparrow A$ with alphabet $\alpha R \cup A$, such that all relationships are permitted so long as the restrictions on relationships in R are preserved.

$$
\begin{array}{l}
[L] \\
\hline
\_@\_, \_ \uparrow \_ : \mathcal{R}[L] \times \mathbb{P}\, L \to \mathcal{R}[L] \\
\hline
R@A = \{\, a, b : (A \cap \alpha R) \mid (a \mapsto b) \in R \,\} \\
R \uparrow A = \{\, a, b : (A \cup \alpha R) \\
\qquad\qquad\qquad \mid \{a, b\} \subseteq \alpha R \Rightarrow (a \mapsto b) \in R \,\}
\end{array}
$$

Example 2 A Palm is owned by a secret user and the overall policy can be specified as

$$\mathsf{SPALM} == \mathsf{PALM} \uparrow \{\mathsf{secret}\}$$

Note that the resulting policy is not transitive: while abacus may flow to secret which may flow to email, abacus may not flow to email, per the original policy. This policy reflects our belief that the secret user is trusted (at that level) and does not copy abacus data to email. That is, we believe that the user is trusted not to violate the flow restrictions of the device by creating transitivity where none should exist. $\triangle$

Flow policies may be compared, in a security sense, using relation $\sqsubseteq$.

$$
\begin{array}{l}
[L] \\
\hline
\_ \sqsubseteq \_ : \mathcal{R}[L] \leftrightarrow \mathcal{R}[L] \\
\_ \sqcap \_ : (\mathcal{R}[L] \times \mathcal{R}[L]) \to \mathcal{R}[L] \\
\mathbf{not} : \mathcal{R}[L] \to \mathcal{R}[L] \\
\hline
R \sqsubseteq Q \Leftrightarrow (\alpha R \subseteq \alpha Q) \wedge (Q@\alpha R) \subseteq R \\
R \sqcap Q = (R \uparrow \alpha Q) \cap (Q \uparrow \alpha R) \\
\mathbf{not}\, R = (\top(\alpha R)) \cup ((\bot(\alpha R)) \setminus R)
\end{array}
$$

If $R \sqsubseteq Q$, then Q is said to be no less restrictive than R in that any flow that is not allowed by R will also not be allowed by Q. We view a $R \sqsubseteq Q$ relation as a refinement relation: the policy defined by Q is, in a security sense, an acceptable replacement for the policy R. Intuitively, this means that a system secure by policy Q is also secure by policy R. The set $\mathcal{R}[L]$ forms a lattice under partial order $\sqsubseteq$, with a lowest upper bound operator defined by $\sqcap$. The lowest upper bound operator is useful for constructing complex policies from simpler policies: $R \sqcap Q$ is a policy that enforces the flow restrictions of R and Q. Since $R \sqcap Q$ is a lowest upper bound on R and Q, then it is, in a security sense, an acceptable replacement for R and Q. The complement of a policy R is given as $\mathbf{not}\, R$.

Example 3 The policy complement operator is useful for constructing policies in terms of the flows that are *not* permitted. For example, we might have a Palm that does not allow spreadsheet data to be 'beamed' via the infra-red port to another.

$$
\begin{array}{lcl}
\mathsf{NOBE} & == & \mathbf{not}(\{\mathsf{abacus}\} \leadsto \{\mathsf{beam}\}) \\
\mathsf{BEAMPOL} & == & \mathsf{SPALM} \sqcap \mathsf{NOBE}
\end{array}
$$

The overall policy, BEAMPOL, depicted in Figure 1, upholds the constraints of the individual policies that compose it. $\triangle$

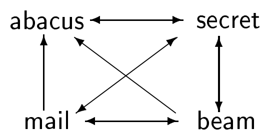

Figure 1: Possible Flows in policy BEAMPOL

The inclusion of class secret in policy BEAMPOL indicates that it is owned/operated by a user cleared at secret. For the purposes of this paper we assume that the PDA will not be used by any user cleared to a different class.

3 Confidence Rated Policies

The policy PALM (Example 1) specifies that we are confident, to some degree, that it is not possible to email Abacus spreadsheet data from a particular Palm handheld. A sophisticated user could bypass this by developing and installing a new Palm program that performs the necessary copying. Policy PALM reflects our belief that this compromise is unlikely and/or we are willing to accept the risks. In [6] we describe a PalmOS extension that enforces a limited type-enforcement security policy. While the extension is not protected and can be bypassed by determined malicious code, we have more confidence in this operating system (HanTE) upholding policy PALM than standard PalmOS. Similarly, we have far greater confidence in a multilevel secure system upholding the policy than either PalmOS or HanTE.

Let the type $[RT]$ represent the set of all possible confidence ratings that we might associate with a system and/or policy. We assume that this set forms a lattice ordering over $\_ \leq \_$, where $s \leq t$ means that we have more confidence in a system rated t than a system rated s.

EXAMPLE 4 Figure 2 gives sample confidence orderings. Since the Palm does not support hardware memory management and winCE does, Palm and winCE ratings are not comparable. △

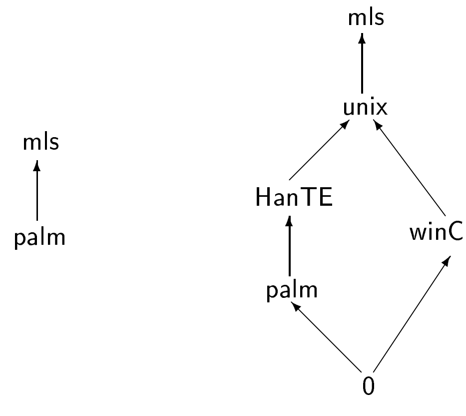

Figure 2: Confidence Rating Orderings R1 and R2.

We include these confidence ratings when specifying flow policies. A *rated policy* is a flow policy over rating/label pairs, whereby $(r, x) \mapsto (r, y) \notin P$ means that one's level of confidence that x does not interfere with y is r. This generalizes to: given $P : \mathcal{R}[RT \times L]$, ratings r, s and $x, y \in \alpha P$, then $(r, x) \mapsto (s, y) \notin P$ means that we are confident that x information on an r-rated system cannot interfere with y information on a s-rated information.

EXAMPLE 5 Given policy PALM and rating policy R1, then Figure 3 gives palm and mls rated versions of this policy. Levels and ratings are abbreviated

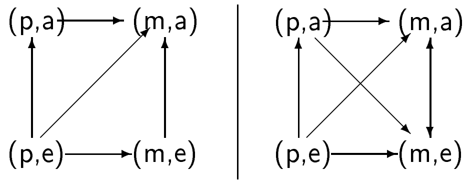

Figure 3: Policy PALM rated at mls and palm .

to their first character: abacus is given as a, and so forth. The mls rated version of policy PALM does not allow any flow from abacus to email under any circumstances. However, while the palm-rated version of the policy (right-hand side of Figure 3) does not allow palm-rated flow from abacus to email ((p,a)$\not\mapsto$(p,e) in Figure 3), we lack mls confidence that abacus does not flow to email, that is, ((m,a)$\mapsto$(m,e)) in Figure 3. △

If one's level of confidence is r that policy P is upheld, then this gives rise to a rated policy $r \fatsemi P$ where,

$$
\begin{array}{l}
[L] \\
\hline
\_ \fatsemi \_ : RT \times \mathcal{R}[L] \to \mathcal{R}[RT \times L] \\
\hline
r \fatsemi P = \{\, s, t : RT;\ x, y : L \\
\qquad\qquad \mid s \leq t \wedge \\
\qquad\qquad\ (s \leq r \wedge t \leq r \Rightarrow x \mapsto y \in P) \\
\qquad\quad \bullet (s, x) \mapsto (t, y) \,\}
\end{array}
$$

It follows from this definition that if my confidence is r that P is upheld then the same policy can be upheld if I decrease my confidence level to $s \leq r$.

EXAMPLE 6 Figure 3 gives the possible flows in the rated policies mls $\fatsemi$ PALM and palm $\fatsemi$ PALM based on the rating ordering R1 from Figure 2. △

Note that a policy rated at '0' (from Figure 2) does not represent complete uncertainty, rather it represents uncertainty at all levels *except* the lowest level '0'.

LEMMA 1 Given ratings r, s, and policies $P, Q : \mathcal{R}[L]$ then it follows from the definitions of $⨾$ and $\sqsubseteq$ that

$$r \leq s \land P \sqsubseteq Q \Rightarrow r ⨾ P \sqsubseteq s ⨾ Q$$

That is, the rated policy $r : P$ can be replaced (refined) by the higher rated policy $s : Q$ without any loss of confidence. □

4 Secure Synchronization

The purpose of synchronization is to ensure data consistency between PDA and host system databases. Changes to data on one platform need to be reflected on the other, and vice-versa. During a Palm 'hot-sync', a Synchronization Manager running on the host system calls a series of *conduits*. Each conduit is responsible for checking and updating the consistency of certain application databases. For example, the Oracle Lite relational DBMS for the Palm provides a conduit that runs on the host, synchronizing selected host/server databases with the (Oracle) application databases on the Palm.

Thus, conduits can be designed to control the flow of information between the handheld and the host system, helping to ensure that the overall system policy is upheld. For example, a conduit might be designed that allows secret and unclassified information to be down-loaded to a Palm (owned by a secret user), but only secret data may be uploaded. We use a rated policy to describe the flow controls enforced by the conduit.

EXAMPLE 7 An email conduit synchronizes unclassified data with the email database on the Palm.

$$\mathsf{C0} == \mathsf{mls} ⨾ \bot\{\mathsf{unclass}, \mathsf{email}\}$$

A spreadsheet conduit synchronizes secret data with spreadsheet database on the Palm.

$$\mathsf{C1} == \mathsf{mls} ⨾ \bot\{\mathsf{secret}, \mathsf{abacus}\}$$

Another email conduit allows only one-way synchronization of unclassified email.

$$\mathsf{C2} == \mathsf{mls} ⨾ \{\mathsf{unclass}\} \rightsquigarrow \{\mathsf{email}\}$$

These conduits are all rated as mls since they are assumed to run on an MLS system. Note that our confidence is based on the (believed) flow-controls of the conduit *and* the behavior of the Synchronization Manager. △

Given rated policies H, P of a host system and a Palm, respectively, and conduit rated policy C, then when the Palm synchronizes with the host the following flows are possible:

- Flows described by H or P.
- Consider that a Host (with $a \mapsto b, g \mapsto h \in H$) is connected to a Palm (with $c \mapsto f \in P$) by a conduit that connects b with c and f with g ($b \mapsto c, f \mapsto g \in C$) then the synchronization results in an indirect flow from a via b, c, f, g to h on the host. These indirect flows may be defined by relational composition $H ⨾ C ⨾ P ⨾ C ⨾ H$.
- Similarly, if Palm (with $c \mapsto d, e \mapsto f \in P$) is connected to a Host (with $g \mapsto b \in H$) by a conduit that connects b with c and f with g ($b \mapsto c, f \mapsto g \in C$) then the synchronization results in an indirect flow from e via f, g, b, c to d on the Palm. These indirect flows may be defined by relational composition $P ⨾ C ⨾ H ⨾ C ⨾ P$.

The composition by synchronization of host policy H with Palm policy P using conduit C is thus defined by $H \,|[\, C \,]|\, P$.

$$[L]$$
$$\_\,|[\,\_\,]|\,\_ : \mathcal{R}[RT \times L] \times \mathcal{R}[RT \times L] \times \mathcal{R}[RT \times L] \rightarrow \mathcal{R}[RT \times L]$$
$$\begin{aligned} H \,|[\, C \,]|\, P = {} & H \cup P \cup \\ & H ⨾ C ⨾ P ⨾ C ⨾ H \cup \\ & P ⨾ C ⨾ H ⨾ C ⨾ P \end{aligned}$$

Note that since policies are reflexive, then $a \mapsto b \in P$ and $b \mapsto c \in P$ does not necessarily imply that $a \mapsto$

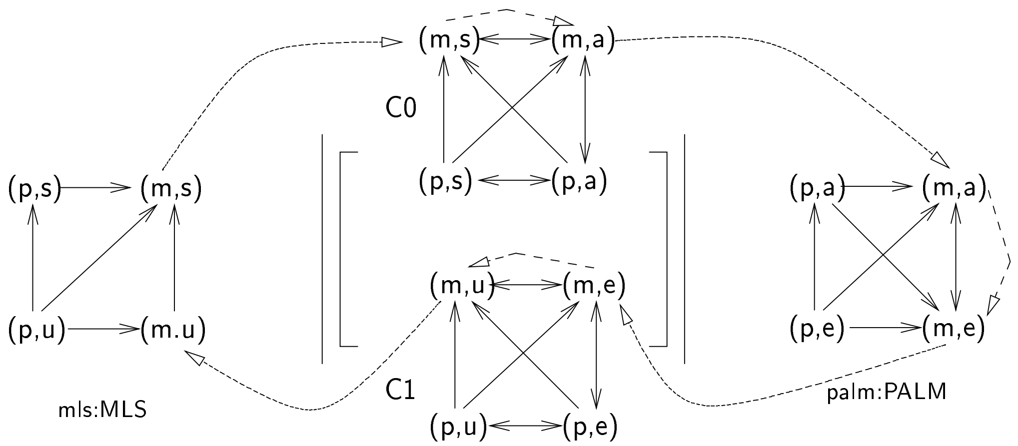

Figure 4: Indirect Conduit Flows.

$c \in P$ and therefore a transitive closure should not be computed for $H \,\|[\, C \,]\|\, P$ on a single synchronization. Section 5 considers multiple synchronization.

EXAMPLE 8 Given host policy $\mathsf{MLS} = \{\mathsf{unclass}\} \rightsquigarrow \{\mathsf{secret}\}$, Palm policy PALM and conduit $\mathsf{C0} \cup \mathsf{C1}$ (Example 7), Figure 4 illustrates an indirect flow from secret to unclassified generated as a result of the synchronization $(\mathsf{mls} \fatsemi \mathsf{MLS}) \,\|[\, \mathsf{C0} \cup \mathsf{C1} \,]\|(\mathsf{palm} \fatsemi \mathsf{PALM})$ $\triangle$

Recall that the policy refinement relation may be used to to compare confidence in rated policies, whereby $R \sqsubseteq S$ means that we are no less confident in S than in R.

EXAMPLE 9 If we disregard flows involving C0 in Figure 4, then we have

$$\mathsf{mls} \fatsemi \mathsf{MLS} \sqsubseteq (\mathsf{mls} \fatsemi \mathsf{MLS}) \,\|[\, \mathsf{C1} \,]\|(\mathsf{palm} \fatsemi \mathsf{PALM})$$

and we are (mls) confident that the policy on the host is upheld when we have two-way synchronization of secret abacus data. We also have

$$\mathsf{palm} \fatsemi \mathsf{PALM} \sqsubseteq (\mathsf{mls} \fatsemi \mathsf{MLS}) \,\|[\, \mathsf{C1} \,]\|(\mathsf{palm} \fatsemi \mathsf{PALM})$$

that is, that the policy on the Palm is also upheld.

However, because of the indirect synchronization flow depicted in Figure 4, our confidence drops to an overall Palm rating if there is two-way synchronization of secret abacus data and unclassified email:

$$\mathsf{mls} \fatsemi \mathsf{MLS} \not\sqsubseteq (\mathsf{mls} \fatsemi \mathsf{MLS}) \,\|[\, \mathsf{C0} \cup \mathsf{C1} \,]\|(\mathsf{palm} \fatsemi \mathsf{PALM})$$
$$\mathsf{palm} \fatsemi \mathsf{MLS} \sqsubseteq (\mathsf{mls} \fatsemi \mathsf{MLS}) \,\|[\, \mathsf{C0} \cup \mathsf{C1} \,]\|(\mathsf{palm} \fatsemi \mathsf{PALM})$$

To uphold confidence in the host policy, only one-way synchronization (down-load) of email data should be supported. We have

$$\mathsf{mls} \fatsemi \mathsf{PALM} \sqsubseteq (\mathsf{mls} \fatsemi \mathsf{MLS}) \,\|[\, \mathsf{C1} \cup \mathsf{C2} \,]\|(\mathsf{palm} \fatsemi \mathsf{PALM})$$

$\triangle$

During synchronization, a number of conduits may be invoked, each one checking the consistency of their respective application database(s). In flow policy terms, these conduits may be modeled as individual flow policies, or as one overall conduit policy.

LEMMA 2 Given rated policies H,P, C_0 and C_1 then

$$H \,\|[\, C_0 \cup C_1 \,]\|\, P = H \,\|[\, C_0 \,]\|\, P \cup H \,\|[\, C_1 \,]\|\, P$$

This follows since since relational composition distributes over union.

COROLLARY Given a rated policy S then it follows that

$$(S \sqsubseteq H \,\|[\, C_0 \,]\|\, P) \wedge (S \sqsubseteq H \,\|[\, C_1 \,]\|\, P)$$
$$\Leftrightarrow S \sqsubseteq H \,\|[\, C_0 \cup C_1 \,]\|\, P$$

This means that we can reason about conduits independently. □

Example 10 Example 9 models two conduits that two-way synchronizes secret with abacus data (C1) and one-way down-load synchronizes unclass with email (C2) in terms of one flow policy. Using Lemma 2 the same result may be achieved as

$$\begin{array}{lcl} \mathsf{mls} \fatsemi \mathsf{MLS} & \sqsubseteq & (\mathsf{mls} : \mathsf{MLS}) \,|[\, \mathsf{C1} \,]|\, (\mathsf{palm} \fatsemi \mathsf{PALM}) \\ \mathsf{mls} \fatsemi \mathsf{MLS} & \sqsubseteq & (\mathsf{mls} : \mathsf{MLS}) \,|[\, \mathsf{C3} \,]|\, (\mathsf{palm} \fatsemi \mathsf{PALM}) \end{array}$$

△

In practice, it may be appropriate to run conduits separately on the host system. In Example 10, separate conduits C1 and C3 can run as untrusted single level processes (at secret and unclass, respectively). To have mls-rated confidence in the flows modeled by C4, synchronization would have to be regarded as trusted since it can, in principle, simultaneously read and write secret and unclass data. Existing research on secure transaction processing is applicable to the development a trusted/multilevel secure synchronization manager.

5 Cascading Conduits

Thus far we have considered flows resulting from a single synchronization. In practice, a Palm is repeatedly synchronized with one or more hosts. Additional flows may emerge as a result of a cascading effect brought about by the repeated synchronization.

Consider a Palm with rated policy P that synchronizes with a host (rated policy H) via conduit C. The resulting flow policy on the Palm can be defined as the projection

$$P' = (H \,|[\, C \,]|\, P)@\alpha P$$

that is, the resulting flows defined over the alphabet of P. A similar policy can be constructed for the host policy.

$$H' = (H \,|[\, C \,]|\, P)@\alpha H$$

A second synchronization may result in additional flows, that is, the resulting policy $H' \,|[\, C \,]|\, P'$ is not necessarily equal to the original policy $H \,|[\, C \,]|\, P$. This is illustrated in the following example.

Example 11 A Palm P synchronizes with host H via conduit C.

$$\begin{array}{lcl} P & == & \top\{k, l, m\}; \\ H & == & H_x \cup H_y; \\ C & == & C_x \cup C_y; \\ H_x & == & \{a\} \leadsto \{b\} \cup \top\{c\}; \\ C_x & == & \{k\} \leadsto \{a\} \cup \{b\} \leadsto \{l\} \cup \{m\} \leadsto \{c\} \\ H_y & == & \{y\} \leadsto \{z\} \cup \top\{x\}; \\ C_y & == & \{x\} \leadsto \{k\} \cup \{l\} \leadsto \{y\} \cup \{z\} \leadsto \{m\} \end{array}$$

The flows resulting from synchronization are depicted in Figure 5. The additional flows $k \mapsto l$, $l \mapsto m$ are indicated by dashed arcs labeled with a '1'. The dashed arcs labeled '2' are due to a cascading effect that the additional flows from the first synchronization generate during a second synchronization. The policy stabilizes after two synchronizations, when the overall policy is $(H \,|[\, C \,]|\, P)@\alpha P) \,|[\, C \,]|((H \,|[\, C \,]|\, P)@\alpha H)$. △

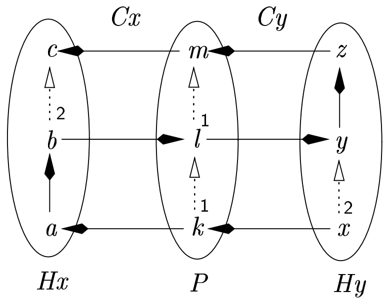

Figure 5: Multiple Synchronizations and Cascading Flows.

In general, the overall flow policy can be computed by repeated calculation of the synchronized policies, as defined in Figure 6. This algorithm terminates and may be viewed as a variation of computing a transitive closure using iterative squaring [3]. We are currently implementing rated policies using Binary Decision Diagrams [2].

$$
\begin{array}{l}
S_{old} = \top(\alpha H \cup \alpha P);\\
S_{new} = H \,|[\, C \,]|\, P;\\
\textbf{while}\,(S_{old} \neq S_{new})\{\\
\quad P = S_{new} @ \alpha P;\\
\quad H = S_{new} @ \alpha H;\\
\quad S_{old} = S_{new};\\
\quad S_{new} = H \,|[\, C \,]|\, P;\\
\}
\end{array}
$$

Figure 6: Computing Cascading Conduit Flows.

Cascading flows emerge when one or more Palms synchronize with one or more systems. Reconsider Example 11; Palm P alternatively synchronizes with two hosts (upholding policies) H_x and H_y via conduits C_x and C_y, respectively. The first synchronization with H_x reveals flow $k \mapsto l$; this is followed by synchronization with H_y which reveals $x \mapsto y, l \mapsto m$. This stabilizes after an additional synchronization with H_x, revealing flow $b \mapsto c$.

Example 12 The problem of cascading flows during multiple synchronizations can be viewed as a generalization of the network cascade problem [4, 10, 12]. Assurance levels can be represented as confidence ratings, and conduits correspond to connections between systems. Flow cascades may be determined by computing the transitive closure of all system policies. For example, given ratings $\mathsf{B1} \leq \mathsf{A1}$, host policies H_a and H_b connected directly, where

$$
\begin{array}{l}
H_a = \mathsf{A1} \between (\{\mathsf{unclass}\} \leadsto \{\mathsf{secret}\})\\
H_b = \mathsf{B1} \between (\{\mathsf{secret}\} \leadsto \{\mathsf{topsecret}\})
\end{array}
$$

Since multilevel policies are transitive, then the overall rated policy is computed as the transitive closure $\mathsf{NET} = (H_a \cup H_b)^*$. This network can be evaluated as $\mathsf{B1}$, but not $\mathsf{A1}$ since we can show that $\mathsf{A1} \between \mathsf{MLS} \not\sqsubseteq \mathsf{NET}$. Our approach is more general than the solution to the network cascade problem since we can reason about networks of components supporting different and possibly non-transitive flow policies. $\triangle$

6 Discussion and Conclusion

In this paper we considered security policy issues that arise when synchronizing handhelds with host systems. A framework was developed that allows us state our confidence in the security of the individual components and test whether that confidence can be maintained when the components synchronize. While the examples were straightforward and were limited to multilevel-style policies, we have shown elsewhere [5] that reflexive flow policies can be used to express a wide variety of security policies. Thus, for example, it is possible to analyze the security vulnerabilities that arise when synchronizing a Palm with a system that enforces Clark-Wilson style policies.

We believe that the framework is applicable to the more general problem of security in networks of heterogenous components. These components represent systems, or alternatively, COTS components whose potential accesses are articulated as a flow policy. It is not necessary for these components to have an *explicit* access control mechanism; the flow policy represents the access limitations that we believe the software effectively upholds. Thus, in the sense of [1], every component in the system can be regarded as contributing to the overall Trusted Computing Base. In our framework we can distinguish the merit of each component's contribution. This gives rise to a paradigm for analyzing security of secure/non-secure components:

1. Identify suitable confidence ordering.

2. Develop rated flow policies for components. Ensuring that every relevant entity is modeled, including users, files, databases, devices, and so forth.

3. If a component incorporates an access control mechanism then the security policy upheld corresponds to the flow policy. In the case of discretionary access, the policy will be based on our confidence of whether access is likely to be granted.

4. If a component has no access control mechanism then the policy represents the access limitations

that we believe the component implicitly provides.

5. Analyze synchronizations.

We use an ordering relation to provide a meaning for confidence. This allows us to compare our confidence in different policies. Alternative confidence metrics may be possible. For example, the probability of a particular access constraint being upheld, or costs related to the insurance value of of the individual systems. The probabilistic approach taken in [11] examines how insecurity may propagate through a protection schemes. Probabilistic and other measures of confidence or trust have also been studied in the context of authentication metrics [13] and it would be worth investigating their applicability to security policies in general.

If a particular composition does not achieve our desired level of confidence there are two alternatives. One is to determine what is the highest level of confidence that can be achieved by the composition; this is a straightforward search over the relation. The other alternative is to limit the accesses possible by the conduits. We expect that an attempt to do this in an optimal way would lead to hard complexity results similar to those for the cascade problem [8, 9] and access-control in heterogenous networks [7]. Devising practical approaches to addressing this in the context of our framework is a topic for future study.

Acknowledgments

Thanks to the anonymous referees and the Workshop audience for useful comments on this research.

References

[1] B Blakley and D.M. Kienzle. Some weaknesses of the TCB model. In *IEEE Symposium on Security and Privacy*. IEEE CS Press, May 1997.

[2] R.E. Bryant. Symbolic boolean manipulation with ordered binary decision diagrams. *ACM Computing Surveys*, 1992.

[3] J.R. Burch et al. Symbolic model checking: 10^{23} states and beyond. *Information and Computation*, 98:142–170, 1992.

[4] J.A. Fitch and L.J Hoffman. A shortest path network security model. *Computers and Security*, 12:169–189, 1993.

[5] S.N. Foley. The specification and implementation of commercial security requirements including dynamic segregation of duties. In *4th ACM Conference on Computer and Communications Security*. ACM Press, 1997.

[6] S.N. Foley and G. Hayes. PalmTE: Limited type enforcemented on the Palm handheld. *In preparation*, 2000.

[7] L. Gong and X. Qian. The complexity and composability of secure interoperation. In *Proceedings of the Symposium on Security and Privacy*, pages 190–200, Oakland, CA, May 1994. IEEE Computer Society Press.

[8] S. Gritalis and D. Spinellis. The cascade vulnerability problem: The detection problem and a simulated annealing approach to its correction. *Microprocessors and Microsystems*, 21(10):621–628, 1998.

[9] R.J. Horton et al. The cascade vulnerability problem. *Journal of Computer Security*, 2(4):279–290, 1993.

[10] J.K Millen and M.W. Schwartz. The cascading problem for interconnected networks. In *4th Aerospace Computer Security Applications Conference*, pages 269–273. IEEE CS Press, December 1988.

[11] I.S. Moskowitz and M.H. Kang. An insecurity flow model. In *New Security Paradigms Workshop*. ACM Press, 1997.

[12] National Computer Security Center, USA. *Trusted Network Interpretation*, 1987.

[13] M.K. Reiter and S.G. Stubblebine. Toward acceptable metrics of authentication. In *IEEE*

Symposium on Security and Privacy, pages 10–20, May 1997.

[14] J. M. Spivey. *The Z Notation: A Reference Manual.* Series in Computer Science. Prentice Hall International, second edition, 1992.

A The Z Notation

A set may be defined in Z using set specification in comprehension. This is of the form $\{ D \mid P \bullet E \}$, where D represents declarations, P is a predicate and E an expression. The components of $\{ D \mid P \bullet E \}$ are the values taken by expression E when the variables introduced by D take all possible values that make the predicate P true. For example, the set of squares of all even natural numbers is defined as $\{ n : \mathbb{N} \mid (n \bmod 2) = 0 \bullet n^2 \}$. When there is only one variable in the declaration and the expression consists of just that variable, then the expression may be dropped if desired. For example, the set of all even numbers may be written as $\{ n : \mathbb{N} \mid (n \bmod 2) = 0 \}$. Sets may also be defined in display form such as $\{1, 2\}$.

In Z, relations and functions are represented as sets of pairs. A (binary) relation R, declared as having type $A \leftrightarrow B$, is a component of $\mathbb{P}(A \times B)$. For $a \in A$ and $b \in B$, then the pair (a, b) is written as $a \mapsto b$, and $a \mapsto b \in R$ means that a is related to b under relation R. Functions are treated as special forms of relations. We use the generic schema notion to define functions giving the function signature followed by its definition.

$\mathbb{P}\, A$	The power set of A
$A \leftrightarrow B$	Relations between A and B
$A \rightarrow B$	Total functions from A to B
dom R, ran R	Domain and Range of relation R
id A	Identity relation over values from A
$R \mathbin{;} S$	Relational composition

B Justification for NSPW2000

The primary objective of this paper is to propose an approach for analyzing the access-control vulnerabilities that can arise from synchronizing PDAs with application systems. We are unaware of any existing results that considers the security of these devices. Meaningful security analysis can be done on application systems even when the PDA provides little or no access control. Achieving this security analysis requires a paradigm-shift on what an access-control policy represents.

Conventional access-control policies specify the access constraints that are to be enforced by a protection mechanism such as a security kernel or security-wrapper based architecture. We depart from this view by assuming that an access-control policy defines the access-limitations that we believe to be reflected by a particular component; whether upheld explicitly by a protection mechanism or implicitly as a result of our belief in the way a component with no protection mechanism behaves. Thus, while a PDA such as a Palm handheld does not have an access-control mechanism, we can still specify, albeit with low confidence, the access limitations that we believe the installed software implicitly provides.

Our approach leads to a new paradigm for modeling and analyzing the access-control vulnerabilities of systems that are comprised of components of varying security. For each component we specify our degree of confidence in the component's ability to uphold its security policy; every system component may be regarded as contributing in some way to the trusted computing base. The overall security policy can then be viewed as a composition of statements, at different degrees of confidence, about access-control. Security analysis determines how the interaction between these statements influences our confidence in security being upheld.