Using Information Flow Control to Evaluate Access Protection of Location Information in Mobile Communication Networks

Heiko Mantel[1], Axel Schairer[1], Matthias Kabatnik[2],
Michael Kreutzer[3], and Alf Zugenmaier[3]

[1] German Research Center for Artificial Intelligence,
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
{mantel,schairer}@dfki.de
[2] Institute of Communication Networks and Computer Engineering, University of Stuttgart,
Pfaffenwaldring 47, 70569 Stuttgart, Germany
kabatnik@ind.uni-stuttgart.de
[3] Institute of Computer Science and Social Studies, Dept. of Telematics,
Albert-Ludwigs-University, Friedrichstrasse 50, 79098 Freiburg, Germany
{kreutzer,zugenmaier}@iig.uni-freiburg.de

Abstract. The increasing functionality provided by mobile devices entails that a considerable amount of sensitive data is stored on them. The possibility to reprogram these devices leads to new security threats like, e.g. Trojan horses or computer viruses, which make the problem of how to guarantee security of this data more important and also more difficult. Protecting privacy of location information in mobile phones is the task on which we focus in this article. It is well known that access control and communication filters provide adequate mechanisms to ensure privacy technically. However, for formalizing security objectives in our setting, i.e. protecting privacy of location information, we argue that information flow control is a more adequate approach. To this end, we use an example to illustrate how security properties which are motivated by standard access control techniques may fail to detect certain insecurities and demonstrate that this problem can be avoided when information flow control is applied.

1 Introduction

In mobile communication networks the location information of terminal devices is the basis for location-dependent routing for speech and data services. At first, the information referred solely to connection establishment but now more and more this information is used for new services that go beyond providing a communication channel.

Accurate location information can be used to enhance the network performance of cellular mobile networks: It does not only help to improve decisions when to hand over from one cell to another but also statistical location information can be used as an input to the planning of cellular networks (cf. [DMS98]). Accurate location information enables billing services like location sensitive billing (”home zone”). Beyond it, location services that deliver the location information to a requester make possible a whole range of services that network operators of mobile communication networks can offer. Examples are:

– Safety services: emergency services, roadside assistance etc.
– Tracking services: fleet management, asset tracking, pet tracking, people tracking, etc.
– Information services: traffic information, nearest drugstore, position specific advertising etc. (cf. Fig. 1).
– Law enforcement like pinpointing the position of a stolen cellular phone.

Fig. 1. Example for a Location Information Service

These examples show that location information needs to be given to other parties than only the subscriber. However, the increasing demand for and the increasing use of location information poses the question of privacy. Subscribers should have control over this information, so that it cannot be used against their intention directly or by collecting the complete history of location information, which would yield a profile of their movements. On the one hand side, it would inhibit desirable and reasonable uses of the mentioned services if one completely prevents others from accessing the location information. On the other hand, making the location information available to third parties may violate the protection goals of a subscriber. Therefore, mechanisms are required in order to allow subscribers to control access to their location information.

Examples for information that can be offered to other persons or to services include:

– The location information in adjustable granularity,
– the identity assigned to the location information, and
– the point of time the location information has been evaluated.

Moreover, it is possible to offer a history of this information.

Location information can be determined by the network (tracking) or by the end device (positioning). In principle, both parties could store it and make it available. In order to maximize the control that a subscriber has over access to this information we assume that the information is stored locally inside the mobile end device.

Some years ago, mobile systems were limited to simple telephone functionality, providing dialing capabilities only. Meanwhile, mobile phones are an extended communication and data processing environment. Next to the very successful Short Message Service (SMS), communication capabilities like Wireless Application Protocol (WAP) and General Packet Radio Service (GPRS) have entered the field. In addition to improved communication capabilities, the computational power of mobile phones has been increased. There are simple data base functions for storing and managing phone book information and received messages. To improve customer attraction many mobile phones host computer games. WAP provides scripting capabilities and the Wireless Telephony Application (WTA) functionality enables data services to program the mobile phone. In order to make even more complex services feasible manufacturers are about to integrate full run-time environments into the mobiles.

Motorola and Nokia have announced mobile phones that support Sun's Java 2 Platform, Micro Edition (J2ME). Sun has specified an architecture including Application Programming Interfaces (APIs) called Mobile Information Device Profile (MIDP) that enables open, third-party, application development. Recent developments (e. g. WebSIM in [GKP00]) have realized web server functionality within a Subscriber Identity Module (SIM) card that is addressable via the SMS communication link.

We observe a shift of possibilities to influence the mobiles' behavior towards the customer. Early versions of mobiles had to be fully programmed by the manufacturer. Later on, operator specific programming was possible. Now we see programmability for content providers (WAP applications). The next step, programmability by the customer is about to become reality. This is even more likely since merging of mobile telephones and Personal Digital Assistants (PDAs) seems to be an almost natural step. This enables simple integration of new services, like a fully customizable location service, into the mobile terminal.

2 Application Scenario

In the rest of this article we concentrate on the following concrete application scenario: employees in stand-by service using mobile phones provided by their employer. The employer's action plan concerning staff on call service in an emergency depends on which of the employees is nearest to the location of the emergency site. Therefore, the mobile phones are equipped with a positioning system, e.g. Global Positioning System (GPS), and an application that allows the employer to poll the current location of his employees. The application has been developed and deployed by the company before the mobile phones have been handed out. A poll results in a number of requests that are sent to the mobile phones, and the answers are collected in a report for the employer, e.g. on a page on the company's intranet.

Location information available on the mobile devices is rather finely grained. Revealing the exact location in response to the employer's poll would, therefore, constitute

a potentially undesirable violation of the privacy of the employees, even more so when by frequent polls an exact trace of the employee's movements could be constructed. On the other hand, the employer's interest in knowing about the current location of the employees can be satisfied by information that is more coarsely grained, without the employees' private life being monitored in an unacceptable manner.

We assume the location information is available on the mobile phone to the highest resolution up to which it can be provided by the positioning system. Requests to the mobile phone include an indication of the requester and a specification of which resolution of the location information is desired by the requester. The user of the phone is able to configure the phone in a way that determines for each possible requester the maximal resolution to which the user is willing to give location information. The exact information consists of several fields, e.g. bits. Starting with the most significant field, a higher resolution can be achieved by adding the information stored in the next less significant field. The user can then configure the maximal resolution for a particular requester by maintaining a table mapping possible requesters to the number of fields of the location information that he is willing to reveal to the requester. In our case, employees would want to grant the employer access to location information with a resolution of, say, 10 km. The intention is that this should be sufficient to determine the city the employee is in, but not the area within the city.

Since an employee has an interest to be protected from giving too much information to the employer in the first place, it cannot be ruled out *a priori* that the employer has tried to build trapdoors into the application loaded onto the mobile phone. Also, enforcement of the employee's policy should be achieved by means that do not rely on the correctness of the application program. Possible solutions that come to mind include a communication filter that the application cannot circumvent when communicating with the employer. It is with this kind of solution we are concerned in the rest of this article. We will, therefore, ignore other threats to the privacy of the current location of the employee, like, e.g. the possibility of the network provider to collect and reveal the cell in which the phone is booked into the net, or sending un-encrypted replies to the wrong recipient.

For ease of presentation, in the rest of the article we restrict the range of possible resolutions to two, e.g. a resolution of 10 km or more coarsely grained and finest resolution. For simplicity we call these *city* and *street* resolution, respectively. In the same vein, we only consider two possible values, "Freiburg" and "Stuttgart", for the city field of the location, and two possible values for the street information: We have called these "pub" and "office", in order to imply that the employer should be able to find out in which city employees are, but not where they spend their stand-by time.

3 Controlling Access to the Location Information

The privacy issues that arise from allowing third parties to access location information can be considered with the imperative: Only give the information to those who are permitted to receive it! Inspired by mandatory access control we set up the architecture to control access to the location information by an access filter as depicted in Fig. 2.

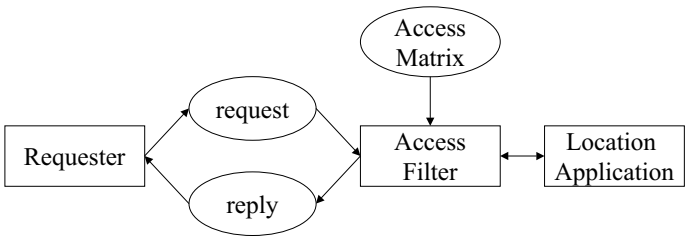


Fig. 2. Controlling Access to the Location Application

Access to location information is restricted by an access filter for incoming requests and outgoing replies. Each incoming request is buffered, the resolution that it asks for is looked up in the access matrix. If there is no permission for the requester to access the desired resolution, the request is discarded. Otherwise, the signature of the request is verified and the request is permitted. If this test passes, the request is forwarded to the location information service program. The result has to pass the information filter again before it is returned to the requester.

The access matrix for the example in Sect. 2 is shown in Fig. 3.

Resolution	Allowed Requester's ID (*rid*)
Street	self
City	self, employer

Fig. 3. The access matrix

The employer is permitted to request up to the resolution of city only, the user of the mobile may request the location information for both resolutions.

The usual inheritance relationships, i.e. that a requester with permission to request street information is always allowed to request city information as well, are maintained by the administration tool used to modify the access matrix. Only a person with physical access to the device may modify the access matrix. For the purpose of this article it can be assumed to be static.

It can be shown that only legitimate requests are answered.

4 Formal Specification

The location information scenario has been formalized in the VSE-system [AHL$^+$00] using a trace-based approach. Each *trace* is a sequence of states and events which starts with a state, the *initial state*, and continues with events and states. An *event* models an atomic action, like, e.g., the transmission of a message on some channel, the testing of a value in some memory location, or the assignment of a new value. Each trace models

a possible behaviour of the system. Consequently, a system can be specified by a set of traces which models all possible behaviours. For example, trace τ (depicted below) models a behaviour which starts in state s_0, the initial state of the system. In s_0, event e_1 occurs which results in state s_1. In s_1, event e_2 occurs resulting in state s_2, etc.

$$s_0 \xrightarrow{e_1} s_1 \xrightarrow{e_2} s_2 \longrightarrow \ldots$$

In our setting, a *state* is a mapping of state objects to values. The state objects for the location information scenario are depicted in Fig. 4. `owner id` stores an identifier of the owner of the mobile phone. The current location information is stored in the street and city field of `current position`. Updates of the location information which are received from the net are buffered in `update buffer`. The access control information is stored in `access matrix`. Incoming requests are stored in the `filter buffer` and in `internal buffer`. Outgoing replies are stored in `output buffer`. In the specification, a *request* consists of three elements: the id of the requester, the id of the owner, and the desired resolution of the location information. A *reply* has an additional element which contains the location information.

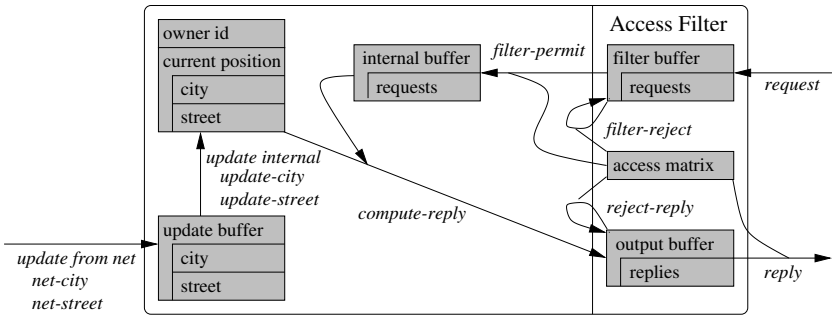


Fig. 4. Visualization of the Specification

Fig. 4 also illustrates the events which have been modeled in the formal specification. E.g. the event *filter-permit*(rid, cid, res) is enabled if and only if `filter buffer` contains a request $\langle rid, cid, res\rangle$ and `access matrix` contains an entry $\langle rid, cid, res\rangle$. An occurrence of *filter-permit*(rid, cid, res) adds the request $\langle rid, cid, res\rangle$ to `internal buffer` and removes it from `filter buffer`. In the figure, each event is depicted by an arrow which originates at the state objects on which the enabledness of the event depends and points to the state object which is affected by an occurrence of this event. The parameters of events are omitted in the figure.

Roughly, two lines of communication can be distinguished in the location information scenario. Firstly, the update of location information and, secondly, the request of location information. Updating the location information is initiated by one of the events *net-street*(cid, val) or *net-city*(cid, val) which respectively update the corresponding field of `update buffer` if *cid* is an identifier of the owner. If the street or city field of

`update buffer` contains a value *val* (*val* $\neq$ *undef*), then respectively *update-street*(*val*) or *update-city*(*val*) is enabled. These events reset the corresponding entries of `update buffer` and update `current position` accordingly.

The handling of requests is slightly more complicated. If a request from a requester with identifier *rid* for location information of *cid* (*cid* = `owner id`) with resolution *res* arrives at the access filter (occurrence of *request*(*rid*, *cid*, *res*)) then a request $\langle rid, cid, res \rangle$ is added to `filter buffer`. Depending on whether `access matrix` contains an entry $\langle rid, cid, res \rangle$ or not, respectively, one of *filter-permit*(*rid*, *cid*, *res*) or *filter-reject*(*rid*, *cid*, *res*) is enabled. Both events remove the entry from `filter buffer` but only *filter-permit* inserts this request into `internal buffer`, thereby ensuring that all requests which are propagated comply with the security policy defined by the owner of the mobile phone. *compute-reply*(*rid*, *cid*, *res*, *pos*) computes a reply $\langle rid, cid, res, pos \rangle$ for a request $\langle rid, cid, res \rangle$ from `internal buffer` where *pos* = *mkpos*(*city*, *street*), *city* is the city in `current position`, and, depending on *res*, *street* is either *undef* (if *res* = *'city'*) or the street from `current position` (if *res* = *'street'*). *compute-reply* deletes the request from `internal buffer` and inserts the computed reply into `output buffer`. An event *reply*(*rid*, *cid*, *res*, *pos*) models sending of a reply $\langle rid, cid, res, pos \rangle$ to the requester. It is enabled if the response is allowed by `access matrix`, i.e. if it contains an entry $\langle rid, cid, res \rangle$. An additional precondition ensures that *street* in *pos* = *mkpos*(*city*, *street*) is *undef* if *res* is *'city'*. Together these preconditions ensure that each reply that is sent complies with the security policy as defined in `access matrix`. If these preconditions are not met the entry in `output buffer` is discarded by *deny-reply*, i.e. no reply is sent. After the occurrence of either of these events, the reply is removed from `output buffer`. Additional preconditions ensure that no buffer entries in `filter buffer`, `internal buffer`, or `output buffer` can be overwritten by new ones before they have been processed.

4.1 Formalization of System Specification

Formally, state-event systems have been used as specification formalism. A *state-event system* $SES = (S, s_0, E, I, O, T)$ is a tuple where S is a set of states, s_0 is the initial state, E is the set of events with subsets I and O of respectively input and output events, and a transition relation $T \subseteq S \times E \times S$. The set Tr_{SES} of traces for a given state-event system is defined inductively: $s_0 \in Tr_{SES}$; if $(s, e, s') \in T$ and $\tau \in Tr_{SES}$ where $s \in S$ is the last state in τ then $\tau.e.s' \in Tr_{SES}$. The set of states, the set of events, and the distinction of input and output events have been specified according to our (somewhat informal) explanations above. In the initial state s_0, the values of all state objects are undefined with the exception of `owner id`, which contains already an identifier of the owner, and `access matrix`, which already contains the security policy. The transition relation T has been formalized using a pre-/postcondition notation which is not directly supported by the VSE-system. For example, the part of the transition relation which is concerned with event *filter-permit* is defined as follows ([] denotes an empty buffer):

filter-permit(*rid*, *cid*, *res*) affects `internal buffer`, `filter buffer`
Pre: `filter buffer` $= \langle rid, cid, res \rangle \wedge$ `internal buffer` $= []$
$\wedge$ *allowed*(*rid*, *cid*, *res*, `access matrix`)
Post: `internal buffer` $= \langle rid, cid, res \rangle \wedge$ `filter buffer` $= []$

This simple pre-/postcondition notation allowed us to formalize the complete state-event system on only $2\frac{1}{2}$ pages. Since this formalism is not directly supported by the VSE-system in the form we need it in Sect. 6, we had to translate the pre-/postcondition based specification into the specification language of VSE, VSE-SL, using abstract data types. The translation is done mechanically by a program we have implemented. The resulting VSE specification is structured in 17 specification modules and consists of 12 pages of specification.

In order to simplify the specification and the resulting proof obligations, we assume that each buffer contains at most one entry, that there is only one possible owner (the employee) and one requester (the employer), that the security policy is not changed dynamically, and we have abstracted from timing information. The security policy allows requests with resolution *'city'* but forbids requests with resolution *'street'*. Note that these simplifications are not essential since VSE incorporates general theorem provers which can deal directly with infinite state spaces unlike model checkers. However, these simplifications reduce the verification effort while still allowing us to demonstrate the point we wanted to make.

4.2 Formalization and Proof of Security Property

The objective of the access filter entity of the mobile phone is to restrict requests for location information such that the security policy is respected. In the simplified employer/employee scenario which we have considered this holds if the employer cannot receive information about the street. This security objective had to be formalized in order to be verified in VSE. This has been achieved by the following four theorems:

1. The resolution of every reply complies with the security policy.
   $\forall s_1, s_2 \in S.\forall rid, cid, res, city, street.$
   $[(reachable(s_1) \wedge T(s_1, reply(rid, cid, res, mkpos(city, street)), s_2))$
   $\Rightarrow allowed(rid, cid, res, \texttt{access matrix})]$
2. Every reply with a defined street has resolution *'street'*.
   $\forall \tau \in E^*.\forall rid, cid, res, city, street.$
   $[(\tau.reply(rid, cid, res, mkpos(city, street)) \in Tr_{SES} \wedge street \neq undef) \Rightarrow res = \text{'street'}]$
3. For every reply there has been a corresponding permit.
   $\forall \tau \in E^*.\forall rid, cid, res, city, street.$
   $[\tau.reply(rid, cid, res, mkpos(city, street)) \in Tr_{SES}$
   $\Rightarrow \exists \beta, \alpha \in E^*.\tau = \beta.filter\text{-}permit(rid, cid, res).\alpha]$
4. For every permit there has been a corresponding request.
   $\forall \tau \in E^*.\forall rid, cid, res, city, street.$
   $[\tau.filter\text{-}permit(rid, cid, res) \in Tr_{SES} \Rightarrow \exists \beta, \alpha \in E^*.\tau = \beta.request(rid, cid, res).\alpha]$

The proof of the first theorem goes by induction over the length of the trace by which s_1 is reached. The other theorems are proved by induction over the length of τ. In the above formulas, the state-event system $SES = (S, s_0, E, I, O, T)$ denotes the specification of the location information scenario. E^* denotes the set of all event sequences and $\beta.e.\alpha$ denotes the event sequence resulting from concatenating an event sequence β, an event e, and an event sequence α. $reachable(s_1)$ expresses that the state s_1 can be reached from the initial state by some trace.

5 A Successful Attack

In our scenario we state the availability of location information within the mobile phone. We assume that there will be services that enable external requests of this information. Since we want to restrict the access to this information by deploying ordered levels of granularity for location information, the access control mechanism is indispensable.

In Sect. 4.2 we have presented the security properties our system is supposed to meet. Actually, our system design can meet these requirements as shown in the preceding section. But there still is a security problem. Since the security properties have been designed to guarantee a certain form of the responses, the system is vulnerable to an attack that changes the meaning of the form. Such an attack enables flow of information by alternating the semantic meaning of some data field when the communication channel is used by collaborating entities. In our example this can be achieved by installing a so-called Trojan horse within the system. Given the programmability of the mobile station, we assume that the employer provided the mobile station including some code (java applet) which realizes the paging service described above. This code usually behaves well. But if the applet detects that the employer requests the subscribers location it behaves maliciously. Whenever such a request enters the system the program enters an alternating mode. For the first request of the employer R a proper response is generated (R asks for the city information and the reply contains the city information as specified). Now the program is waiting for a second request of R. The second request will be answered in a different way. The program changes the meaning of the city field and maps the street information onto this field. “Office” is coded as “Stuttgart” while “Pub” is expressed as “Freiburg”. The external attacker must be synchronized to this shift of meaning and can decode the information accordingly. If the mobile system is monitored, the generated responses never violate the security properties. Even the filter device parameterized by the granularity of the incoming request, and applied to the responses does not improve the situation. The only check which is performed by the filter is on the syntactical correctness of the form of a reply.

6 Consequences

In this section, we revisit the argumentation that the access filter cannot prevent leakage of street information to the employer. Firstly, we illustrate how the Trojan horse has been formalized in VSE. Secondly, we show that the security property from Sect. 4 is inadequate because it also holds for the insecure system, i.e. the system with Trojan horse. Thirdly, we propose a different security property which is based on a notion of information flow control. Finally, we show that this security property is adequate. It holds for the system without Trojan horse but not for the system with Trojan horse.

6.1 Formalization of Trojan Horse

The Trojan horse works by alternating between two different modes of computing a reply to a request by the employer. In the first mode, it passes back the current city and in the second mode, it passes back the current street encoded as a reply pretending

to contain information only about the current city. The Trojan horse switches between these modes after every reply. In the second mode, it uses the encoding depicted below.

street	encoding
Pub	*Freiburg*
Office	*Stuttgart*

Note that only the second way of computing a reply violates the confidentiality of the information about the current street. We make use of this fact by simplifying the presentation (and specification) by focusing on the second type of reply. Thus, we specify the Trojan horse by events *trojan-horse-reply* which replace the corresponding events *compute-reply*. The transition relation for events *trojan-horse-reply* is specified as follows.

$$
\begin{array}{l}
\mathit{trojan\text{-}horse\text{-}reply}(\underline{\mathit{rid}},\underline{\mathit{cid}},\underline{\mathit{res}},\mathit{mkpos}(\underline{\mathit{city}},\underline{\mathit{street}})) \\
\quad \text{affects } \texttt{output buffer}, \texttt{internal buffer} \\
\textit{Pre:}\ \ \texttt{internal buffer} = \langle \underline{\mathit{rid}},\underline{\mathit{cid}},\underline{\mathit{res}} \rangle \wedge \texttt{output buffer} = [] \\
\qquad \wedge (\underline{\mathit{res}} = \text{`}\mathit{street}\text{'} \Rightarrow \langle \underline{\mathit{city}},\underline{\mathit{street}} \rangle = \texttt{current position}) \\
\qquad \wedge (\underline{\mathit{res}} = \text{`}\mathit{city}\text{'} \Rightarrow ((\mathit{street}(\texttt{current position}) = \mathit{Pub} \Rightarrow \underline{\mathit{city}} = \mathit{Freiburg}) \wedge \\
\qquad\qquad\qquad (\mathit{street}(\texttt{current position}) = \mathit{Office} \Rightarrow \underline{\mathit{city}} = \mathit{Stuttgart}))) \\
\textit{Post:}\ \ \texttt{output buffer} = \langle \underline{\mathit{rid}},\underline{\mathit{cid}},\underline{\mathit{res}},\mathit{mkpos}(\underline{\mathit{city}},\underline{\mathit{street}}) \rangle \wedge \\
\qquad \texttt{internal buffer} = []
\end{array}
$$

6.2 Security Property holds for Insecure System

Although the system with Trojan horse leaks information about the current street, i.e. the current street is returned encoded as a city, all security properties from Sect. 4.2 still hold. After all, we have only changed the way in which the city field of a reply is computed. It is easy to see that in the presence of the Trojan horse we still have that

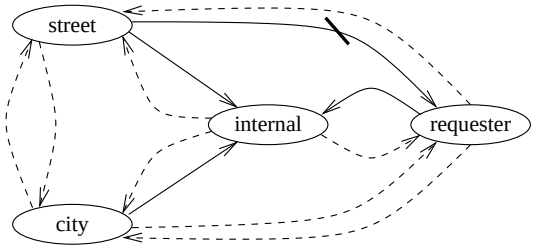
1. the resolution of every reply complies with the security policy,
2. every reply with a defined street has resolution *'street'*,
3. for every reply there has been a corresponding permit, and
4. for every permit there has been a corresponding request.

The proofs of these properties are similar to the ones in Sect. 4.2.

6.3 An Improved Specification of the Security Property

The system with Trojan horse leaks information because the Trojan horse can encode street information into the city field of a reply. The underlying problem is that the security properties 1–4 (cf. Sect. 4.2) only ensure that a reply with a non-empty street field may leave the device only if the requester has been granted access to street information. However, these properties are not concerned with the information communicated by the content of the fields in a reply. Since the system with Trojan horse is apparently insecure, we conclude that these security properties do not provide an adequate specification of the intended notion of security.

As a solution, we now propose an improved specification of the intuitive security objectives, which is based on *information flow control*. This approach has been introduced by Goguen and Meseguer under the name *non-interference* [GM82]. Rather than controlling the access to certain objects, information flow control allows one to prevent information flow which would violate the security objectives. In information flow control, different security domains have to be identified, which can be thought of as, e.g., groups of users, processes, memory sections, data, or other parts of the system that are relevant for security. Security properties can be formalized as restrictions on the information flow between domains. Using this approach, confidentiality as well as integrity properties can be expressed. However, in our example, we investigate aspects of confidentiality only. We distinguish four domains: street, city, internal, and requester. Domain street, e.g., is associated with information about the current street and domain requester with information which can directly be accessed by the employer (the only requester according to our simplification). Consequently, our security objective can be formalized by restricting information flow from domain street to domain requester.



e	$dom(e)$
net-city	city
net-street	street
update-city, *update-street* *filter-permit*, *filter-reject* *compute-reply*	internal
request, *trojan-horse-reply*	requester

Fig. 5. Flow Policy and Domain Assignment

Somewhat more formally, each domain corresponds to a set of events, whereas, the set of all domains must form a disjoint partition of the set E of events. The domains in our example are defined in the table on the right hand side of Fig. 5 by a domain assignment *dom* which associates a domain with each event. E.g. all *request* and *reply* events are associated with domain requester. The diagram on the left hand side of Fig. 5 depicts a *flow policy*, which defines the restrictions on the information flow. A crossed arrow from a domain D_1 to a domain D_2 expresses that there must not be any information flow from D_1 to D_2. This not only requires that occurrences of events in D_1 cannot be observed by D_2 but also that such occurrences cannot be deduced from other information. A solid arrow from D_1 to D_2 expresses that occurrences of events in D_1 can be observed by D_2. A dashed arrow from D_1 to D_2 expresses that occurrences of events in D_1 *cannot* be observed by D_2 but that we do not care if they can be deduced, however such deduction must not reveal any information about domains which are confidential. The main security requirement in our example is formalized by the crossed arrow from domain street to domain requester. This is the only crossed arrow in the flow policy. Note that there need not be a crossed arrow from, e.g. domain internal to domain requester although, e.g., the occurrence of an *update-street* event should not be

deducible by the requester. The argument for this is as follows: assume the requester could deduce the occurrence of an *update-street* event then he could also deduce that a corresponding *net-street* event has previously occurred. However, this cannot be the case because of the crossed arrow from street to requester, and, therefore our initial assumption, i.e. that the requester can deduce information about *update-street* events must have been false.

In order to prove that a system specification satisfies a flow policy (for a given domain assignment), we have to define formally, what information flow means. Starting with Goguen's and Meseguer's non-interference [GM82] several other definitions of information flow have been proposed. Since the original proposal from [GM82] is restricted to deterministic systems, we use a generalization [Man00a] which is compatible with non-determinism.[4] In order to define formally what information flow means, one first has to deal with direct information flow which results from observations of the running system. We modeled the behaviour of our system by a set of traces, i.e. sequences of events. Given a trace $\tau \in E^*$, the requester can directly observe only occurrences of request- and reply events. From these observations and his knowledge about the program contained in the system, he may be able to deduce information about the occurrence of other events, which he cannot directly observe. For example, if the requester receives a reply (occurrence of event $reply(rid, cid, res, pos)$) then he can deduce that an event $compute\text{-}reply(rid, cid, res, \ldots)$ has previously occurred. Of course, many other deductions are possible from this observation. However, the key property we have to ensure is that the requester cannot deduce that any *net-street* event has occurred. Formally, this is achieved by *backwards strict deletion* (abbreviated by *BSD* in the sequel). Informally, *BSD* demands that, if a behaviour τ is possible for the system all sequences which result from τ by deleting certain *net-street* events (from right to left in τ) are also possible traces. Given any observation of the requester, this ensures that the requester cannot deduce that any *net-street* events have occurred. For an introduction to properties of this kind and a formal definition of *BSD* we refer to [Man00a]. Note that, properties like *BSD* are closure properties on the set of traces of a system, which demand that if some sequence is a possible trace then certain other sequences must be possible traces as well. For proving these kind of properties local verification conditions, so called *unwinding conditions*, are used. Unwinding conditions and corresponding unwinding theorems for *BSD* have been presented in [Man00b].

6.4 Improved Security Property holds only for Secure System

In order to specify our security objectives adequately, we proposed a specification of security properties based on information flow control in Sect. 6.3. The resulting proof obligation requires that the set of traces Tr_{SES} is closed under *BSD* for the view of every domain, i.e. BSD_D must be fulfilled for every domain *D*. *We have proved that this statement holds for the system without Trojan horse formally in the VSE-system*. For discharging the proof obligation, we have applied the unwinding results from [Man00b]. *Unwinding conditions* are local conditions which are formulated in terms of pre- and

[4] The first proposal for a generalization of non-interference to non-deterministic systems, due to Sutherland [Sut86], has been followed by several other proposals.

postconditions of single events (rather than being a global closure condition on the set of traces, like *BSD*). For *BSD*, there are two unwinding conditions *lrf* and *osc* (cf. [Man00b] for a formal definition). The corresponding *unwinding theorem* ensures that *BSD* holds if one can construct an *unwinding relation*, a preorder, i.e. a reflexive and transitive relation, such that *lrf* and *osc* are satisfied for every domain.

Some experiences from our proof of the information flow property shall be summarized but the details of our proof in VSE are outside the scope of this paper. Firstly, the unwinding relation has been constructed using an *invent-and-verify* approach, i.e. by first guessing the relation and then proving that it is appropriate. Our first guess of the unwinding relation has turned out to be flawed, which we found out when we were unable to discharge certain proof obligations (as usual with invent-and-verify). It is our impression that the development of tool support for a mechanical construction of unwinding relations should be possible. However, we have not followed the latter approach in this case study. Secondly, four proof obligations have required major effort during proof construction: the proofs of the unwinding conditions for the view of the requester-domain (proofs of *lrf* and *osc*) and the proofs that the unwinding relation is a preorder (reflexivity and transitivity). Among these proofs, the one for *osc* was most tedious. During the proof we experienced that, for the choice of the unwinding relation, there appears to be a trade-off between simplifying the proof of *osc* and of transitivity.

As we expected, our attempts to prove the information flow property for the system with Trojan horse failed. In fact, any such attempt is bound to fail because the closure property does not hold for the system with Trojan horse. The removal of *net-street*-events in certain traces of the system with Trojan horse yields event sequences that are no traces of the system, thus violating *BSD*. One counterexample is the following trace (omitting non-vital parameters):

net-street(...,Office).*update-street*(...,Office).*request*(...,*'city'*).
filter-permit(...,*'city'*).*compute-reply*(...,*mkpos*(*Stuttgart*)).
reply(...,*mkpos*(*Stuttgart*)).*net-street*(...,Pub).*update-street*(...,Pub).
request(...,*'city'*).*filter-permit*(...,*'city'*).*compute-reply*(...,*mkpos*(*Freiburg*)).
reply(...,*mkpos*(*Freiburg*))

Deleting the last *net-street*-event from this trace (*net-street*(...,Pub)) results in an event sequence which cannot be adapted to a trace of the system by adding or removing events in domain internal and domain city which occur after the deleted occurrence of *net-street*(...,Pub). Consequently, *BSD* does not hold for the system with Trojan horse.

The above trace yields the following observation for the requester:

request(...,*'city'*).*reply*(...,*mkpos*(*Stuttgart*)).*request*(...,*'city'*).
reply(...,*mkpos*(*Freiburg*))

This observation does not allow the requester to deduce that precisely the above trace has occurred because there are a couple of similar traces which yield the same observation. However, the requester can deduce that first an event *net-street*(...,Office) and later on an event *net-street*(...,Pub) must have occurred. Recall that this does not comply with the flow policy in Fig.5 and that it would also violate the intuitive security objective (the employee would not like if the employer is able to deduce that he has gone into the Pub!).

7 Towards a Solution

In the previous sections it has been shown that the security properties as stated in Sect. 4.2 do not prohibit an information flow on the street resolution towards an attacker. On the other hand, it is not clear how to implement the more adequate formulation of the security properties in Sect. 6. A mechanism is needed to enforce the security properties.

The problem, that the system is subverted by a Trojan horse is well known from computer systems running foreign code. Web browsers are a typical example.

A first and straight forward solution to prevent any program code from transmitting location information with fine granularity is shifting the access control barrier from the entry point of the requests towards the storage. Thus, access of the application is inhibited, if a requestor has limited access rights.

Another concept is the limitation of access rights for an application. This is the principle which is applied in the Java sandbox concept. Depending on the source of the application, the access rights regarding certain resources are limited. For more flexibility the rights could be assigned depending on the request, causing the instantiation of a Java virtual machine (JVM). A certain policy could be activated depending on the clearance of the source of the application. The policy information parameterizes an access control mechanism towards the location storage.

A drawback of this approach is that a program restricted in such a manner could not provide any service to the mobile user (e.g. presenting location depended information on the mobile phone's display). Thus, a clearance class depending on the initiator of an application instance reduces flexibility.

We propose a different approach to overcome the Trojan horse problem and the inflexibility of the solution above. The underlying problem is not caused by the entity that sends the request. Instead the problem is the entity that finally receives the answer (although they are usually the same). Protection mechanisms that act on the entrance of a domain are useful for integrity protection of the domain. In terms of confidentiality, the protection must be placed at the exit of a domain.

Therefore, we suggest the application of mandatory control mechanism [BL75] in combination with the principle of high water marking [Wei69]. The idea behind this concept is the assumption, that an application running within the mobile is harmless (concerning confidentiality of location information) as long it does not communicate with a domain which has a lower classification.

Every communication channel (different target addresses are different channels) that might be addressed by the application after accessing location information, is assigned a clearance level for a certain resolution (city or street in the example).

An application is assigned a certain resolution level, according to the storage units (e.g. bytes) it has read already. Whenever the program reads some location information, this resolution level is updated according to the high water mark principle. The mark (resolution level) is shifted towards higher resolution only. In our example an application has the initial level "0" (no access to location information occurred). After reading the city information this level is shifted to "1". Reading the street memory leads to a classification of "2". Reading some lower resolution information at a later point of time does not reduce the resolution level.

Whenever the application tries to write information on a channel (e.g. display, SMS, etc.) the channel's/addressee's clearance is compared against the resolution level of the application. If the acquired resolution level is higher than the clearance of the channel the system discards the write operation. The resolution level simply states the possibility of information flow from location memory towards a certain channel and leads to a pessimistic decision in order to protect the confidentiality protection goal.

Concerning a mobile system augmented with Java capabilities, the Java security concepts do not solve the problem. Although the granularity of security mechanisms has been refined [GMPS97], the principle of stack inspection [WF98] is limited to a history related view. Thus, rights of processes can be restricted when a certain function is called (within the subroutine context), but rights will be reverted after returning from the subroutine. Any operation performed during a procedure call will have no influence on the upper levels of calls in the stack of the application.

In the case of mobile systems where location information might be the primary privacy concern, e.g., the JVM should be extended by an additional label that stores the resolution level and is used according to the scheme given above. The system calls for sending any information into communication channels have to be checked by the JVM with an ACL containing the clearance for each medium or address against the label attached to the application running in the JVM. Due to the limited scope of the location information the computational requirements for the necessary run time checks are acceptable.

In case where more differentiated protection is needed, other solutions must be used. Myers and Liskov, e.g., propose in [ML00] a solution based on labels expressing access policies attached to variables that enable information flow checks performed on the byte code of applets at compile time.

In order to prevent covered channels within the local environment, channels between different applications, and towards any shared resources must either be treated in the same way as communication channels or access must be disabled if the resolution level is greater than zero. An approach concerning mechanisms to avoid timing channels can be found in [Aga00,FGM00].

8 Conclusion

Protecting the privacy of location information in mobile devices has been the main concern in the case study presented. How to formalize the intuitive security objectives in an adequate way has been a key question in this process.

We have applied two approaches: firstly, using techniques which are well-known from mandatory access control and, secondly, applying a property from a general framework for information flow control [Man00a]. Certain insecurities have not been detected when we followed the approach inspired by access control. Note that the underlying problem is not specific to our specification (including the use of a communication filter rather than a proper reference monitor) but a more general theoretical problem, e.g. covert channels cannot be detected by mandatory access control [BL75]. However, using information flow properties instead has revealed these insecurities. This suggests

that information flow control is more appropriate than access control for formalizing the intuitive notion of security in this setting.

Nevertheless, access control has been beneficial to implement the security objectives which have been formalized by an information flow property. This experience is in line with experiences made in a previous case study in the setting of smart card operating systems (for a different information flow property and a different access control model) [SRS+00]. In the current case study, we have used a combination of a communication filter inspired by the Bell/LaPadula model [BL75] with a high water mark model [Wei69]. We argued that the high water mark model can be implemented using an application level labelling mechanism (cf. Sect. 7).

The experiences from this case study suggest a couple of useful directions for future work. In particular, we plan to generalize and to improve the tool support which we have prototypically implemented for the purposes of this case study. Moreover, it would be desirable to prove formally that our solution based on the combination of the communication filter and a high-watermark model (cf. Sect. 7) indeed satisfies the information flow property from Sect. 6. An implementation of the high-watermark model appears to be possible at the high-level programming language, machine language, operating system, or at the hardware level. The question which of these levels is most appropriate for protecting the privacy of location information in mobile devices in practice has been outside the scope of this article. However, this is an important question. Another general direction of future work is how to develop systems by refinement from abstract specification in the context of information flow properties in practice. Theoretical foundations for this have been setup in [Man01].

Acknowledgments. We would like to thank Moritz Strasser for the illustration in Fig. 1. Moreover, we thank Stefan Denne, Serge Autexier, and Alexandra Heidger for their helpful comments on the presentation. This research has been partly supported by the Deutsche Forschungsgemeinschaft DFG (German Research Society).

References

[Aga00] Johan Agat. Transforming out Timing Leaks. In *27th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, Boston, Massachusetts, January 19-21 2000. ACM.

[AHL+00] Serge Autexier, Dieter Hutter, Bruno Langenstein, Heiko Mantel, Georg Rock, Axel Schairer, Werner Stephan, Roland Vogt, and Andreas Wolpers. VSE: Formal Methods Meet Industrial Needs. *Special Issue on Mechanized Theorem Proving for Technology Transfer of the STTT-Springer International Journal on Software Tools for Technology Transfer*, 3(1):66–77, 2000.

[BL75] D. E. Bell and L. J. LaPadula. Secure computer system: Unified Exposition and Multics Interpretation. Technical Report ESD-TR-75-306, MITRE Corp. MTR-2997, Bedford, MA, 1975.

[DMS98] Christopher Drane, Malcolm Macnaughtan, and Craig Scott. Positioning GSM Telephones. *IEEE Communications Magazine*, 36(4):46–54,59, 1998.

[FGM00] R. Focardi, R. Gorrieri, and F. Martinelli. Information flow analysis in a discrete-time process algebra. In *Proceedings of the IEEE Computer Security Foundations Workshop*, pages 170–184, Cambridge, UK, July 3–5 2000, IEEE.

[GKP00] Scott Guthery, Roger Kehr, and Joachim Posegga. How to Turn a GSM SIM into a Web Server. In *Proceedings of CARDIS'2000*, Bristol, UK, September 20-22 2000. HP Labs.

[GM82] J. A. Goguen and J. Meseguer. Security Policies and Security Models. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 11–20, Oakland, CA, April 26–28 1982.

[GMPS97] Li Gong, Marianne Mueller, Hemma Prafullchandra, and Roland Schemers. Going Beyond the Sandbox: An Overview of the New Security Architecture in the Java Development Kit 1.2. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, Monterey, California, December 1997.

[Man00a] Heiko Mantel. Possibilistic Definitions of Security – An Assembly Kit –. In *Proceedings of the IEEE Computer Security Foundations Workshop*, pages 185–199, Cambridge, UK, July 3–5 2000. IEEE.

[Man00b] Heiko Mantel. Unwinding Possibilistic Security Properties. In *Proceedings of European Symposium on Research in Computer Security (ESORICS)*, LNCS 1895, pages 238–254, Toulouse, France, October 4-6 2000. Springer.

[Man01] Heiko Mantel. Preserving Information Flow Properties under Refinement. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 78–91, Oakland, CA, May 14–16, 2001. IEEE.

[ML00] Andrew C. Myers and Barbara Liskov. Protecting privacy using the decentralized label model. *ACM Transactions on Software Engineering and Methodology*, 9(4):410–442, October 2000.

[SRS+00] Gerhard Schellhorn, Wolfgang Reif, Axel Schairer, Paul Karger, Vernon Austel, and David Toll. Verification of a Formal Security Model for Multiapplicative Smart Cards. In *Proceedings of European Symposium on Research in Computer Security (ESORICS)*, LNCS 1895, pages 17–36, Toulouse, France, October 4-6 2000. Springer.

[Sut86] D. Sutherland. A Model of Information. In *9th National Computer Security Conference*, September 1986.

[Wei69] C. Weissman. Security Controls in the ADEPT-50 Time Sharing System. In *AFIPS Conference Proceedings*, volume 35, New Jersey, 1969.

[WF98] D. S. Wallach and E. W. Felten. Understanding Java Stack Inspection. In *Proceedings of the 1998 IEEE Symposium on Security and Privacy*, pages 52–63, Oakland, California, May 1998. IEEE.