Privacy Monitoring and Assessment for Ubiquitous Systems

Architecture design and Java prototype implementation

Mitja Vardjan and Jan Porekar
Research department
SETCCE
Ljubljana, Slovenia
{mitja.vardjan, jan.porekar}@setcce.si

***Abstract*—Pervasive services and their respective front-ends can have access to large amounts of personally identifying and sensitive private information. Most of the platforms allow for user to control which particular private information APIs the services can access and allow the end-user to accept or reject a particular privacy policy. However, the actual privacy practices of services may differ from the ones promised in the privacy policy. This paper outlines the basic assessment mechanisms and measures that enable user with insight on how much private information is actually being accessed and forwarded by each service deployed to the platform. The paper presents architecture for monitoring and assessing privacy practices of pervasive services deployed into a generic pervasive service platform. Furthermore, the paper presents platform specific privacy assessment implementation design for Java OSGi based pervasive platforms and describes an initial implementation and preliminary privacy assessment results, based on correlating data access events with data transmission events.**

Keywords-privacy; assessment; monitoring; pervasive; ubiquitous

I. INTRODUCTION

The ubiquity of smart phones and Internet connected devices with integrated sensing capabilities is resulting in more and more collecting, processing, aggregating and trading of personal information. As third party applications deployed on smart phones and internet connected devices can gain access to sensitive personal information, the amount of privacy threats and attacks is increasing (see [1], [2], and [3]).

The main focus of digital privacy protection is on a-priory protection minimizing and preventing the actual data release out of the data subject's realm. A lot of work has been done also on a-posterior privacy protection addressing protecting the data subject's interests and threats resulting from situations after the data have been released to the data controller [5]. Threats, misuse cases and generic a-posterior solutions for privacy in ubiquitous systems have been investigated in [4] and [6]. Lately, Hildebrandt et al. [7] have argued that in the new data intensive world it is not enough to merely stick to the minimization of data release and data concealment. The suggested solution is to keep all privacy practices and additionally increase transparency on how the data is collected and stored (see [7] for more information). As a result of this there has been an increasing trend in research to specify and prototype the so-called Transparency Enhancing Tools (TETs). Some recent research that is related to this paper includes the privacy logging tools that have been investigated by Hedbom et al. [8]. Furthermore, the system-wide dynamic taint tracking and analysis system capable of simultaneously tracking multiple sources of sensitive data has been investigated by the TaintDroid [9]. The system provides real-time analysis by leveraging Android's virtualized execution environment.

In this paper, we present a Privacy Transparency Enhancing ARchitecture (PEAR) along with initial implementation of Transparency Enhancing Tool (TET). The tool enables monitoring of privacy practices performed by third party services deployed on SOCIEITES ubiquitous platform [12] that is based on Virgo OSGi container. First, we describe the privacy monitoring and assessment architecture and describe how it relates to management of pervasive third party services deployed on a ubiquitous service platform. Then, we present initial set of privacy assessment mechanisms and privacy assessment visualizations.

II. PRIVACY MONITORING AND ASSESSMENT ARCHITECTURE

A typical service installed on user's personal mobile or other pervasive device can have access to a local database (Figure 1) where personal data is stored, including user related activity, location, and medical information. Mobile device's built-in sensors may continuously insert new data into the database. Other sensitive data such as credential storage and user profile can also be made available to the service. Services can then use the ability to communicate with local network or the internet and send any available data to other nodes and users.

To assess privacy practices of services, the privacy assessment component monitors any data access or data transmission by other services, calculates privacy assessment metrics, and outputs the assessment result to the end user (Figure 1). The user can then make an informed decision about any further actions against a service. The user can deny the service access to local data, disallow data transmission, or uninstall the service (Figure 1).

 ISBN: 978-1-61208-270-7

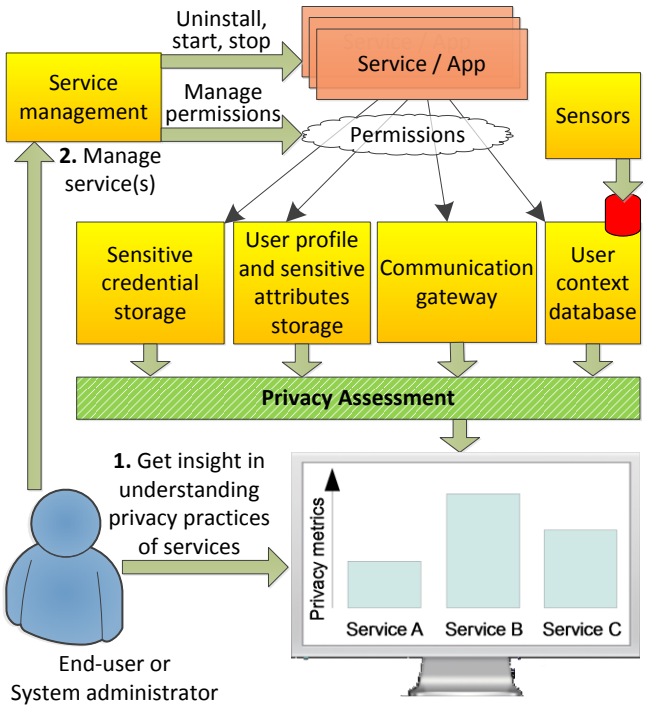

Figure 1. Interactions with the Environment

It should be noted that such a service is not necessarily a malicious service, or a service that violates the service level agreement (SLA). It is just a service that is suspected of accessing and/or transmitting user data more than it is necessary for its normal operation, whether the service behavior is in accordance with the SLA or not.

The approach shown in Figure 1 assumes a limited number of possibilities for a service to access or send data. These privacy monitoring points are integrated with privacy assessment component and whenever a service successfully uses these privacy monitoring points, the privacy assessment component is notified. From data properties, it can deduct and assess privacy practices of services.

Internally, privacy assessment component consists of four main building blocks shown in Figure 2. Arrows indicate data flow. The event logger is integrated with privacy monitoring points from Figure 1, collects events from the monitoring points and stores them into privacy practices log. The log itself is not available to other components and services because it contains sensitive information – the details about all recorded data access and transmission events.

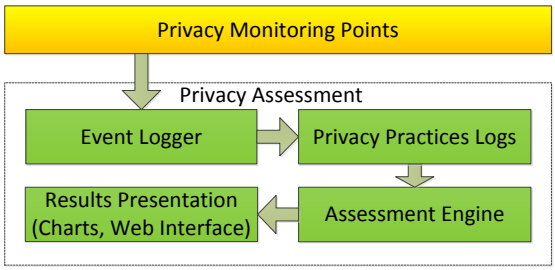

Figure 2. Internal Privacy Assessment Architecture

The assessment engine periodically parses the log, calculates privacy metrics for all software units or other entities in the log, and stores the results. The last block further aggregates results and exposes them to authorized external components, or displays them to the user directly.

A. *Data Access Event*

Any successful attempt to access user's data should be logged. Such data access events are not too hard to detect and log, assuming there are only a few possible ways to retrieve the data, and all of them are well-known. Examples include database access points and application-specific storage. Logging of data access events can be implemented by interception, such as with Aspect Oriented Programming (AOP) [13], or by integration of data access points and privacy assessment. The event details or features to store are at least those which may be available also for subsequent data transmission events and may help to correlate the events later. These details are typically time, data type, data size, and requestor identity (process ID, system user name, application-specific identity, etc.)

Collection of all that information alone can be problematic from the privacy point of view and the user needs to trust the privacy assessment software will not abuse the collected data.

B. *Data Transmission Events*

Capturing data transmission events is not as straightforward as capturing data access events. While there are only a limited number of data access points where particular data can be accessed, there are plenty possibilities (in terms of software) to transmit data over network.

However, there are cases where a single and well-known communication gateway is expected to be used. For example, in some pervasive environments the data receiver is not resolvable to an IP, email or other common address, and the communication gateway (or a limited number of communication gateways) offers a convenient way to transmit messages to specified receivers (Figure 7). While this is not a requirement for the proposed architecture, it enables feasible and efficient gathering of additional data, associated with a particular transmission event. Examples of such additional data include the environment-specific receiver endpoint, software module which transmitted the data, and transmitter identity if there are many possible identities with different authorization levels.

When it is required to detect and log all transmitted packets, it is necessary to use a traffic logger or parser on a lower level, which cannot be omitted when transmitting data. In such more general case where the logging point for data transmission is, e.g., on the TCP level (using a network traffic parser in Figure 7), less details about data transmission event can be gathered. This paper focuses on the former case, where a common gateway is assumed.

III. PRIVACY ASSESSMENT METRICS

The assessment results or privacy assessment metrics should give the user an estimated measure of how privacy

 ISBN: 978-1-61208-270-7

invasive each unit of software is. The software unit can be anything that can be mapped to a particular service. The given assessment values shall be directly comparable between multiple software units and also between multiple time intervals. Therefore, the metrics value should increase with number of data access and data transmission events. For a fixed number of data access and data transmission events, it should increase with estimated probability that the accessed data have in fact been transmitted. In addition to metrics, derived from correlations between the events, number of data access events and number of data transmission events themselves are important metrics. All these metrics are presented in details below.

Correlations are calculated from the available features from data access and data transmission events. First, correlations for individual event pairs are calculated for every available feature and labeled $c_{ij,k}$, where i is index of data access event, j is index of data transmission event, and k is feature index. An individual correlation c_{ij} is product of all feature correlations, e.g., time correlation and data size correlation for given events pair.

$$c_{ij} = \prod_{k=1}^{n} c_{ij,k} \quad (1)$$

In Figure 3, all non-zero correlations between individual data access and transmission events c_{ij} are symbolized with dotted lines. Event pairs where data transmission precedes data access are obviously uncorrelated, and correlations c_{11}, c_{21}, c_{22}, c_{31}, c_{32}, c_{41}, c_{42}, and c_{43} are zero and not shown in Figure 3.

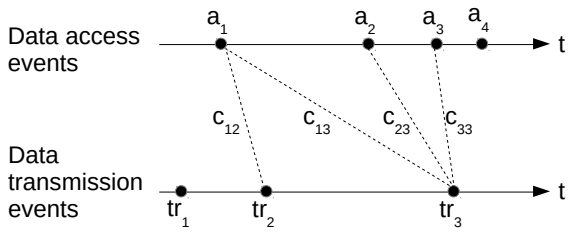

Figure 3. Events through time

Data size correlation function for individual event pairs is given by (2). The argument is difference in sizes of accessed data s_a and transmitted data s_t. If it was assumed that whenever a piece of data was accessed and then transmitted, its transmitted size was exactly the same as the size of original local data (ignoring even the addition of transmission protocol headers), then Kronecker delta function would be most appropriate. To amend for possible modification, aggregation, splitting, compression and encryption of the data before its transmission, the function is smooth, based on Gaussian curve, with asymptote greater than zero. Constants k_1 and k_2 present x-axis scaling and value at infinity, respectively. They can be adjusted to fine tune the algorithm.

$$c_{ij,1} = e^{-\left(\frac{s_t - s_a}{k_1}\right)^2} \cdot (1 - k_2) + k_2) \quad (2)$$

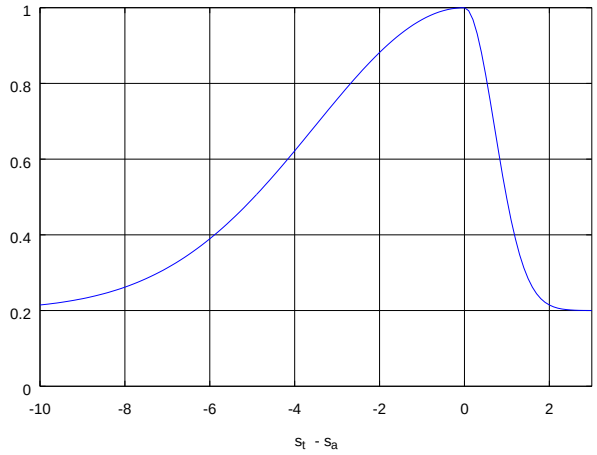

Figure 4. Correlation in data size for a single pair of events

Figure 4 shows an example of (2) where value of k_1 depends on sign of s_t - s_a. The example in Figure 4 is suitable for environments where data aggregation is less likely than data compression and splitting. For environments where services may often encrypt the data before transmission, it is even more important to adjust k_1 in this manner, because encryption usually includes compression.

Correlation in time is a completely different function, based on sigmoidal function of argument Δt or t_t - t_a, where t_t and t_a are times of data transmission and data access, respectively. Equation (3) presents the function for positive Δt values.

$$c_{ij,2} = \left(1 - \frac{1}{1+e^{-\frac{\Delta t}{k_3} - k_5}}\right) \cdot \frac{(1-k_4)}{\left(1 - \frac{1}{1+e^{k_5}}\right)} + k_4 \quad (3)$$

For negative argument values the correlation in time is zero (see Figure 5, for example) because a transmission event should not be correlated to any later data access event.

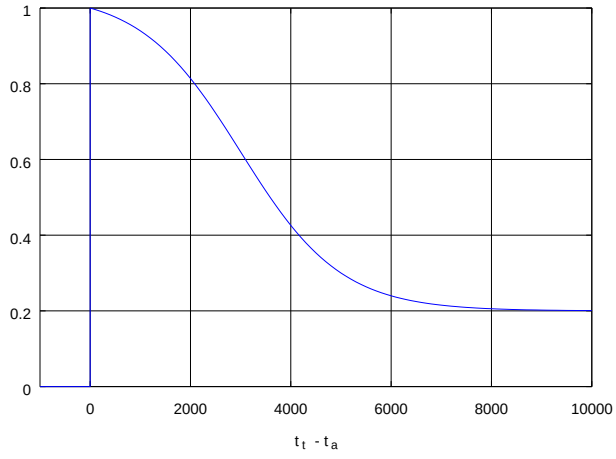

Figure 5. Correlation in time for a single pair of events

The combined correlation for a single pair of events is calculated by (1). Additional features can be added where applicable. If only time and size features are available, then *n* from (1) equals 2. Correlations given by (1) are still intermediate results. Next step is to calculate correlation for a single data transmission event using (4). This is accomplished in two basic manners, which produce two or more results:

- Correlation with all data access events, i.e., for every *i* in (4).
- Correlation with only those data access events where the requestor matches the sender from the transmission event, i.e., for a limited set of *i* values. Depending on implementation, the requestor and sender can have one or more formats and meanings, leading to one or more distinct correlation results.

$$c_j = \sum_i c_{ij} \quad (4)$$

The rationale for multiple correlations with data access events is availability of alternative routes from data retrieval to transmission (Figure 6). Correlation of a transmission event with only those data access events where the requestor matches data sender is the more reliable correlation in terms of least false positive errors. If it is known that component C_B transmitted data that has similar features to data D that had been retrieved previously by component C_A, then it is possible that component C_B transmitted data D. This possibility is more likely if it is known that C_A equals C_B, hence the greater reliability of such correlation. This correlation is most appropriate in cases where software components are assumed to retrieve the data and then transmit the data themselves. For events where this assumption is not satisfied (alternative route 2 in Figure 6), this correlation is more prone to false negative errors than correlation with all data access events.

The correlation with all data access events does not make that assumption, but is more likely to falsely correlate a data transmission event with data access events, i.e., it is more prone to false positive errors. This is especially problematic in a pervasive and sensor rich environment where multiple data are accessed and processed by third party services and may be independently transmitted to other parties. Such cases highlight the convenience, or even necessity of using a single communication gateway at high level where more features about the requestor and transmitter can be retrieved.

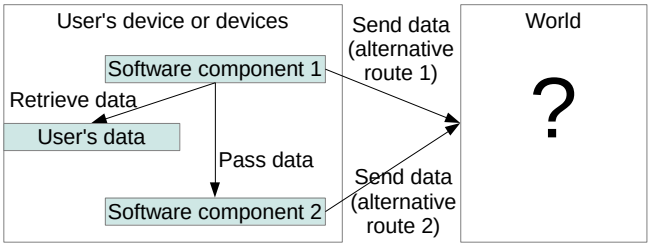

Figure 6. Alternative routes of data transmission

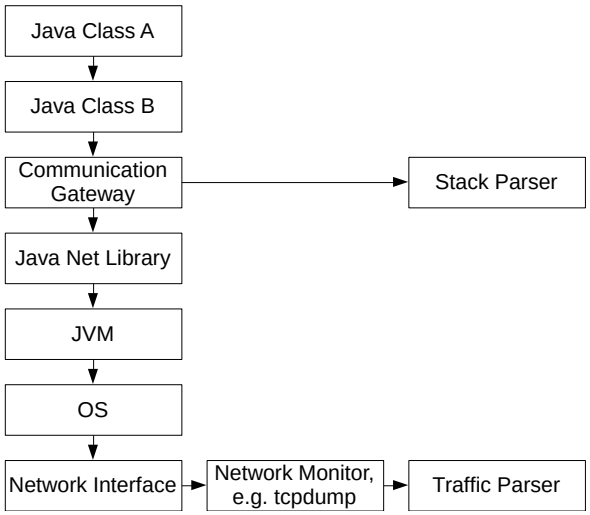

Figure 7. Data Transmission Detection

Regardless of the correlation type used in (4), final metrics for data sender *s* are calculated by summing those c_j where the sender associated with c_j matches *s* (5). This produces at least two different C_s values for each sender: at least one $C_{s,all}$, where (4) had been calculated for every *i*, and at least one $C_{s,matching}$, where (4) had been calculated for only those events where the requestor and sender match.

$$C_s = \sum_s c_j \quad (5)$$

IV. PRIVACY ASSESSMENT IMPLEMENTATION FOR VIRGO

The Privacy Transparency Enhancing Tool has been developed for ubiquitous platform SOCIEITES [12], based on Virgo application server. In current implementation, the data access monitoring points and communication or sharing monitoring points have been realized.

The platform supports the concept of identities and each local service or a remote peer can be associated with an identity. Details such as requestor and sender identity and Java class name are captured for all data access and transmission events by parsing Java stack trace (Figure 7). On this platform it is very likely that services will use the common communication gateway for communication with other peers because it offers a convenient way to resolve remote identities and communicate with them. If this was not the case, a traffic parser on a lower level should have been used instead (Figure 7).

Java class name and identity values are stored as requestor in data access events and sender in data transmission events. This results in three distinct correlation types for a single data transmission event produced by (4):

- correlation with all data access events,
- correlation with only those data access events where the requestor class matches the sender class from the transmission event, and
- correlation with only those data access events where the requestor identity matches the sender identity from the transmission event.

 ISBN: 978-1-61208-270-7

These result in two final results for each sender class and two final results for each sender identity, which make four C_s result types from (5). The first two metrics are calculated for each class, and the last two for each identity:

- assessment metric for data sender class where all data access events were correlated,
- assessment metric for data sender class where only data access events with matching requestor class were correlated,
- assessment metric for data sender identity where all data access events were correlated, and
- assessment metric for data sender identity where only data access events with matching requestor identity were correlated.

For transmission events where data is passed to and transmitted by some other class (alternative route 2 in Figure 6), the value of correlation c_j from (4) with only data access events with matching class is zero. However, the correlation with events with matching identities amends for the missing information about passing the data between components. Correlation with those data access events where the requestor identity matches data sender identity implies a reasonable assumption – both components, services or classes that retrieve and transmit the data, do that under the same identity. In the unlikely situation where this is not the case, the metric derived from correlation with all data events still provides non-zero results.

V. ASSESSMENT RESULTS

A. *Presenting Results to User*

Only high-level results calculated with (5) are shown to the user. A web based graphical user interface (GUI) shows results in form of bar charts, generated by the prototype (Figures 8-12). Figures 8, 9 and 12 show overall numbers of data access and data transmission events by identity and class name. Figure 9 shows the remote identities of data receivers. These basic results complement the metrics calculated by the assessment and help the user understand service behavior.

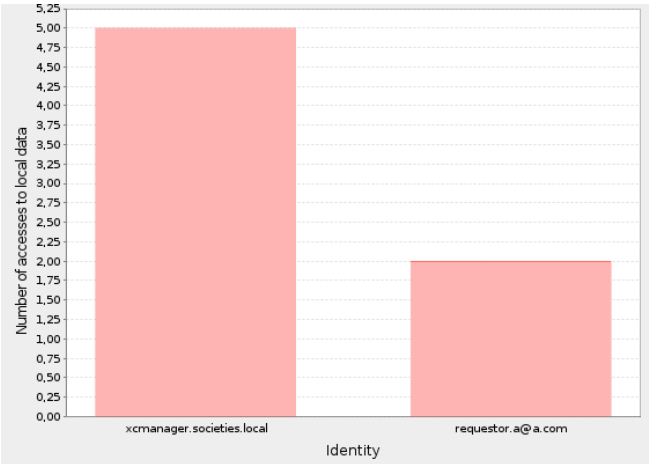

Figure 8. Local data access by identities

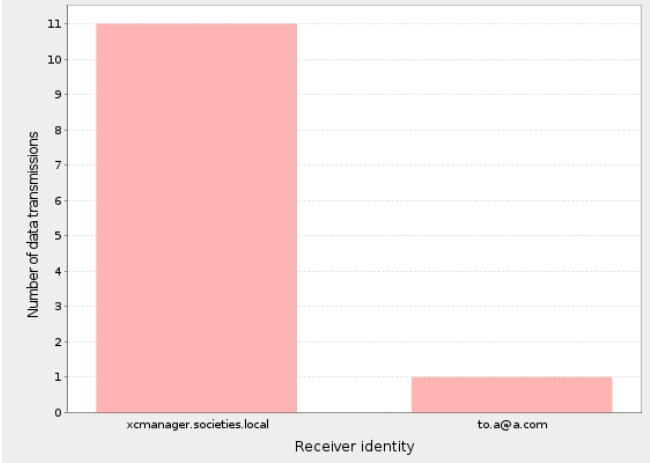

Figure 9. Data transmissions by receiver identity

Privacy assessment metrics for a particular class are shown in Figure 10, and Figure 11 shows the metrics for a particular identity. There are two classes that have transmitted data (the blue bars in Figure 10), but they have not accessed the data (value of both light-red bars is zero). However, this does not necessarily mean they have not transmitted local data because the non-zero blue bars indicate some data access events that had occurred before these classes transmitted data and the classes in Figure 10 might had received the data from other data requestor classes (route 2 in Figure 6).

Similar picture is shown in Figure 11, where the non-zero light-red bar shows the identity under which local data has been accessed at least once, and then at least once something has been transmitted under same identity. The higher blue bar for that identity shows there had been data access events by other identities, too. These data access events increased number of correlations by all data access events, i.e., the number of summands in (4).

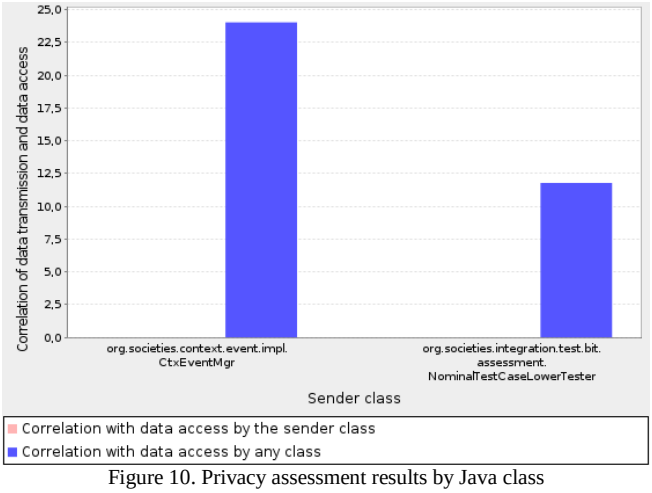

Figure 10. Privacy assessment results by Java class

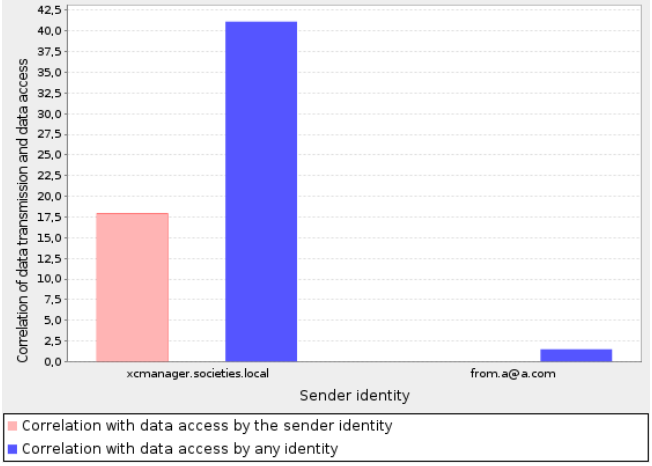

Figure 11. Privacy assessment results by identity

B. *Reacting to Results at Enforcement Points*

Seeing the privacy assessment results, the user can choose to react and impose limits on a particular service. To do that, enforcement points should be implemented at communication gateway, system firewall, and/or data access points (Figure 1). The more rigorous measure is to uninstall the services suspected of unnecessary user data transmission. Of course, there may always be services that access local data, receive and transmit data over network as part of their normal operation. A service with a low non-zero correlation may be even more suspicious than a service with higher correlation, e.g., if the former is not supposed to use network excessively or not at all.

VI. CONCLUSION AND FURTHER WORK

Privacy monitoring and assessment architecture was presented. The initial assessment analysis mechanisms based on correlating data access and transmission events have been described and the implementation of assessment visualization mechanisms has been presented. In the future we aim to extend the assessment method with more data type semantics and explore how current assessment mechanisms can support privacy risk and threat metrics combining our system with results from [10]. Additionally, we aim to port and adapt the prototype implementation for Android OS to support monitoring and assessing services deployed on smart phones.

ACKNOWLEDGMENT

Authors would like to thank the SOCIETIES project [11] consortium and EC for sponsoring the project.

REFERENCES

[1] L. Sheea, J.Alford, and R. Coffin, "Future of Privacy Forum Mobile Apps Study," http://www.futureofprivacy.org/wp-content/uploads/Mobile-Apps-Study-June-2012.pdf, June, 2012 [retrieved: March, 2013].

[2] M. Becher et al. "Mobile security catching up? Revealing the nuts and bolts of the security of mobile devices," Security and Privacy (SP), 2011 IEEE Symposium on. IEEE, 2011, pp. 96-111.

[3] A. Shabtai et al., "Google Android: A Comprehensive Security Assessment," IEEE Security & Privacy, March/April, 2010, pp. 35-44.

[4] M. Deng, K. Wuyts, R. Scandariato, B. Preneel, and W. Joosen, "A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements," Requirements Engineering, vol. 16, no. 1, 2011, pp. 3-32.

[5] C. Bryce et al., "Ubiquitous Privacy Protection: position paper," Proceedings of the 5th Workshop on Ubicomp Privacy in conjunction with the 9th International Conference on Ubiquitous Computing (UbiComp'2007), September, 2007, pp 397-402.

[6] K. Dolinar, J. Porekar, and A. Jerman Blazic, "Design Patterns for a Systemic Privacy Protection," The International Journal On Advances In Security, IARIA, 2009, vol2-3, pp. 267 – 287.

[7] M. Hildebrandt et al., "D 7.12: Behavioural Biometric Profiling and Transparency Enhancing Tools," FIDIS WP7 deliverable, http://www.fidis.net/fileadmin/fidis/deliverables/fidis-wp7-del7.12_behavioural-biometric_profiling_and_transparency_enhancing_tools.pdf [retrieved: March, 2013].

[8] H. Hedbom, T. Pulls, and M. Hansen, "Transparency Tools," Privacy and Identity Management for Life, doi: 10.1007/978-3-642-20317-6, Springer Berlin Heidelberg, 2011, pp. 135-143.

[9] W. Enck et al., "TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones," Proceedings of OSDI, 2010, pp. 393-407.

[10] R. Savola, "Towards a risk driven methodology for privacy Metrics development," Symposium on Privacy and Security Applications (PSA'10), August 2010, pp. 20-22.

[11] Self Orchestrating CommunIty ambiEnT IntelligEnce Spaces (SOCIETIES), EU FP7 project, Information and Communication Technologies, Grant Agreement Number 257493.

[12] M. Bordin, J. Floch, S. Rego, and A. Walsh, "D3.1: Service Model Architecture", SOCIETIES project WP3 deliverable.

[13] R. Johnson et al., "The Spring Framework – Reference Documentation," chapter 6, "Aspect Oriented Programming with Spring," http://static.springsource.org/spring/docs/2.0.x/reference/aop.html [retrieved: April, 2013].

 ISBN: 978-1-61208-270-7

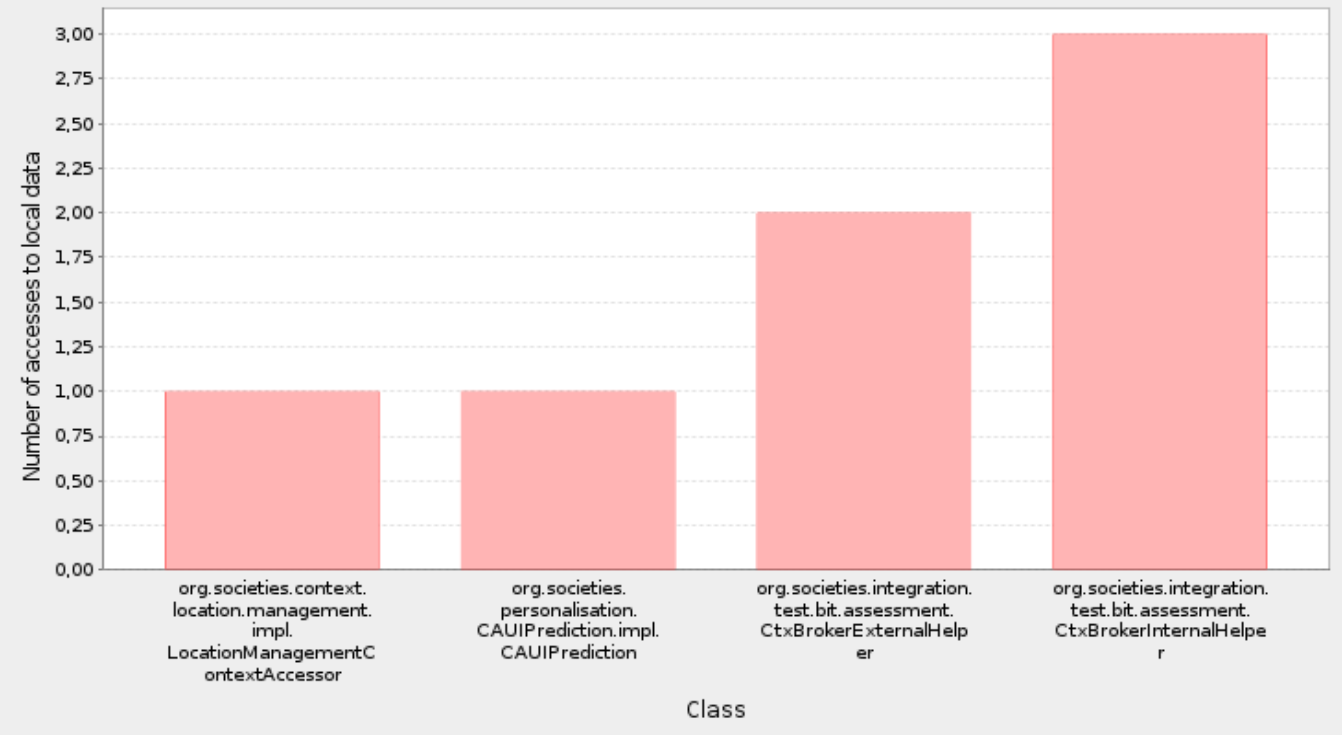

Figure 12. Local data access by Java classes

 ISBN: 978-1-61208-270-7