Composing Cross-Domain Solutions

Ashish Gehani Gabriela F. Ciocarlie

SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025, USA
{ashish.gehani,gabriela.ciocarlie}@sri.com

ABSTRACT

High-assurance systems with multiple security levels use data filters to facilitate the safe flow of information from higher to lower classification levels. Since filters play a critical security role, they must be formally verified. However, a wide range of sanitization strategies has been developed to address the wide variety of data content and contexts that arise in practice. As the diversity of the content and context increases, the complexity of monolithic filters grows rapidly, making them decreasingly tractable for formal verification. The MILS philosophy argues for the decomposition of any functional unit that is too large to be formally verified. Inspired by MILS, we argue that (i) data sanitization should be decomposed, (ii) each filter should handle a specific type of content, and (iii) the sanitization should provide a streaming differential privacy guarantee. Together, these will allow formal assurances for the data sanitization in the system.

Categories and Subject Descriptors

K.4.1 [**COMPUTERS AND SOCIETY**]: General—*Privacy*; D.4.6 [**OPERATING SYSTEMS**]: Security and Protection—*Information Flow Controls*

General Terms

Design, Security

Keywords

composition, cross domain solution, downgrade, filter, sanitize, MILS, streaming differential privacy, certification

1. INTRODUCTION

Despite being an integral component of the U.S. Defense Department's global information grid (GIG), current Cross-Domain Solutions (CDS) for assured information sharing (AIS) are known to have significant limitations, particularly for net-centric operations [15, 5]. Furthermore, CDS deployment times must be reduced from current levels (of months to years) to satisfy stringent requirements, such as U.S. Navy training event setup times (of one month for U.S.-coalition events, one week for U.S.-only events, or one day for Department of Defense-only events) [27].

This agile CDS vision can be achieved by decomposing the problem into subproblems that are more tractable, and then integrating the component solutions. The building blocks for achieving this include formal specifications for generic downgrading engines, formal languages for data sanitization rules [12, 7], filters for specific data types, and attribute-based access control [18]. Using pre-certified commercial off-the-shelf (COTS) CDS facilitates rapid deployment in the field. However, the availability of COTS devices depends on their timely evaluation. To this end, we examine an approach for complex data sanitization operations in decomposed filters that aims to speed up CDS evaluation time.

Section 2 describes the settings in which the need for data downgrading has arisen. Section 3 explains why the increasing complexity of downgrading operations poses a challenge to certifying the available solutions, the motivation for a paradigm shift in how downgrading is performed, and the limitations of current data sanitization technology in the new framework. Section 4 describes an architecture for decomposing downgrading to facilitate certification, and composing the assurance provided. We conclude in Section 5 that despite the challenges, it should be possible to deploy data sanitization technologies for the increasingly complex data and contexts of high-assurance systems within the target timeframes.

2. DATA DOWNGRADING

We examine the use of data downgrading in two settings: high-assurance systems and privacy-preserving data publication. We distinguish between a *filter*, *guard*, and *downgrader*, with the first performing a reduction in data fidelity, the second verifying it, and the third effecting the combination.

High-assurance CDSs are instrumental when information sharing across security domains is necessary *and* the repercussions of security breaches are very significant – for example, when sharing seemingly benign information discloses a latent vulnerability that can compromise a system. If the impact is virtually irreversible, even infrequent occurrences are not acceptable. The most conservative approach is to block all channels of information flow between components with differing security classifications. However, this prevents

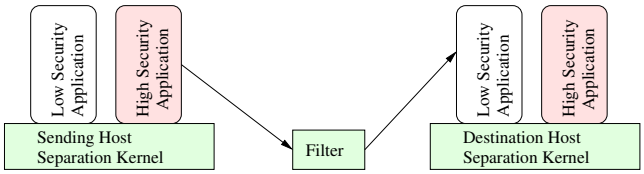

Figure 1: A *filter* intercedes on channels between hosts to sanitize data flowing from a higher security level to a lower level.

legitimate high-to-low information sharing and data transfers from lower-security sources to higher-security destinations. In the latter case, unidirectional communication links with data diodes can facilitate data flow, but this provides no protection against the introduction of malware.

If a system's security-critical components are decomposed into modules that can each be completely verified and between which no unauthorized interaction can occur, then the integrated system's security can be assured. The *separation kernel* introduced by Rushby [23] provides such functionality by isolating partitions running with multiple independent levels of security on a single host and controlling the information flow between partitions, as illustrated in Figure 1. Moreover, dowgraders need to cope with multiple types of data [22], requiring transformation and sanitization mechanisms to allow the information flow.

3. CERTIFICATION CHALLENGE

Since downgraders operate at the boundary between data of different sensitivity levels, they are security-critical components of a computing infrastructure. Assurance of the security of a system is therefore dependent on being able to verify the correctness of downgrader operations.

Early downgrading focused on specific data types and predefined contexts. For example, the U.S. Department of Defense's Global Positioning System (GPS) used to downgrade the location information available to civilians (and therefore also adversaries) by adding a pseudorandom error to part of the signal [16]. While this approach simplified the process of verifying that the downgrading operation conformed to its specification, the need for more complex downgrading policies became apparent with the development of *differential GPS*. (The errors added during downgrading could be calculated in real time by a receiver at a known location that then retransmitted the error stream to consumers who could use it to determine their own location with greater accuracy.)

As the range of data content sensitivities, environments from which it originates, communities with which it needs to be shared, and kinds of operations being performed on it continue to increase, so has the complexity of the rules used to downgrade data flowing between different security classifications. The statistical guarantees provided by privacy-preserving data publication algorithms depend on the soundness of the data sanitization infrastructure.

Certification of a downgrader requires detailed requirements and specifications, proofs of correctness, a structured design process, detailed documentation, and the development of test cases with sufficient coverage. Thus, as the downgrader becomes increasingly complicated, the cost of formal assurance and the time that elapses before it can be deployed in the field also grow.

4. DECOMPOSING SANITIZATION

A few approaches for handling data in multi-level secure systems have relied on separate instances of an application running at each security level [13, 26]. In contrast, we address the problem of downgrading data that has components with multiple classification levels, as occurs during complex joint training missions [4]. We advocate an approach that leverages the nature of the data being downgraded and the available trusted computing infrastructure to decompose the downgrading functionality to the point that each module can economically be formally specified and have its operational behavior verified. This is illustrated in Figure 2.

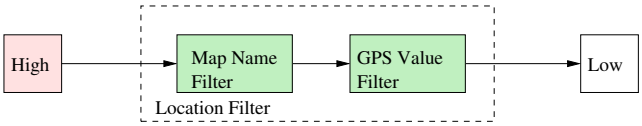

Figure 2: Data sanitization functionality can be decomposed until each sanitization primitive can be certified. For example, sanitization of location information can be split into one operation that redacts blacklisted names from a map while a second perturbs the numeric values in GPS values.

Architectural Context

Since we use an architecture-based solution to make complex data sanitization practical, we first describe the MILS [4] context in which it is framed.

The MILS philosophy advocates a top-down approach to securing a system, mirroring the architectural pattern employed by the *multiple independent levels of security* initiative [2]. The essence of the MILS architectural pattern is to first describe the required security policy in terms of a *policy architecture* (typically, with a diagram in which boxes encapsulate processing functions and arrows depict information flow). The difficulty of making the assurance case is then examined, assuming a direct mapping between the architecture and a physical realization of it. If there is difficulty in proving the assurance case, then the policy architecture is further decomposed until a point is reached where the assurance case can be proved.

The MILS approach separates the desired security properties from the resource-sharing problem. Commodity implementation of individual modules is facilitated through the development of *protection profiles* that articulate the properties that the resources must possess. Downgraders form one class of such functionality. A significant reason that current downgraders are limited to simple operations, such as redacting patterns contained in a database of blacklisted terms or perturbing numeric values using fixed rules [24], is to ensure that they are consistent with the MILS philosophy of individual modules implementing well-defined security properties.

Filter Configuration

Downgraders operate on structured data that has a defined data model [3]. The information is transferred in fixed format messages with multiple data fields and the allowable range for each specified in the associated metadata. The information is intended to support interoperability between machines. Consequently, any piece of data that is a candidate for sanitization is either of a type for which a filter exists or has an associated data model that can be used to decompose the data into constituent objects. The same analysis can be applied recursively to the constituent objects, allowing a suitable set of filters to be selected. Together, the set can be used to sanitize the target data object.

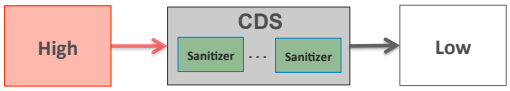

Figure 3: *Intra-CDS* composition deploys multiple filters within a single downgrader.

Assuming the set of filters needed has been determined, they can be deployed in multiple possible configurations. The sanitization guarantees must compose in all cases. In the first configuration, a single downgrader may utilize multiple filters internally to handle different data types, as illustrated in Figure 3. Traditional monolithic downgraders utilize this approach, parsing the incoming data and applying one or more sanitization algorithms to subsets of the information.

Figure 4: Data flows sequentially through consecutive filters when *serial CDS* composition is employed.

In future environments, where the CDS is decomposed into modular pieces that can be verified, two new configurations arise. In one configuration that we consider, multiple downgraders are employed in serial order, with computational operations performed between, as illustrated in Figure 4. One downgrader is used when the input of the intermediate computation must be sanitized, while a second downgrader is used to sanitize the output. The final configuration manifests when data flows from a higher classification level to a lower classification through multiple paths that do not intersect. In this case, a separate downgrader is needed along each path, as illustrated in Figure 5.

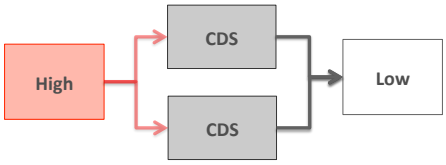

Figure 5: Data can also flow simultaneously through multiple filters when *parallel CDS* composition is utilized.

Sanitization Algorithms

Data sanitization algorithms have been extensively studied in the context of publishing privacy-sensitive data. In this setting, individuals contribute their records to a trusted publisher that processes the collected information. The results are stored in a database that can be queried [14]. To preserve the privacy of individual record owners, a downgrader sanitizes information derived from such databases.

Early work proposed perturbing the query inputs and outputs, and restricting the number of queries [1]. Since then, a range of strategies has been utilized [6], including *generalization* that coarsens, abstracts, or collects multiple data in equivalence classes, *suppression* that removes records from the sanitized output, *swapping* that interchanges attributes from different records, *randomization* that adds noise to perturb the data, and *multi-views* that provide sanitization through diverse perspectives.

The approaches may be domain agnostic and focus on *quasi-identifiers*, data fields that are potentially sensitive. Examples of this include *k-anonymity* [25], which aims to ensure that the output contains at least k records with the same quasi-identifiers, *l-diversity* [19], which ensures that auxiliary fields contain at least l different values, and *ϵ-privacy* [20]. Alternatively, the approaches may be customized to specific data types, such as network addresses [28], data formats, such as audit logs [17], or specific application domains, as is the case for each scheme that guarantees *differential privacy* [8].

Historically, the database that is being sanitized is assumed to be static. In many contexts this is a reasonable assumption since all the relevant information is collected first, before sanitization and querying is performed. In the case of a CDS, the data may never have been observed previously. Traditional offline algorithms must be replaced by online ones that perform data sanitization by operating on a stream of information as its arrives. Recent research on *streaming differential privacy* [9] provides a framework for designing sanitization algorithms appropriate for a CDS. Of particular utility is the fact that the guarantees in this framework compose. This provides a sound basis for designing modular filters that can be used in MILS architectures while ensuring that their combination provides formal sanitization guarantees.

Threat Model

The framework in which streaming differential privacy can be defined for a CDS assumes that each filter operates on a stream of data objects. Each incoming data object is processed in a single step, during which the object is examined, the internal state of the filter is emitted, and an output may be emitted. This is illustrated in Figure 6. The filter can also produce output at the end of the stream.

We conjecture that the *continual observation* [11] threat model used in streaming differential privacy can be adapted to allow an adversary to monitor all the output produced by a filter, as illustrated in Figure 7. In the MILS setting, the output of the filter flows to a lower security classification level where an adversary may have increased access.

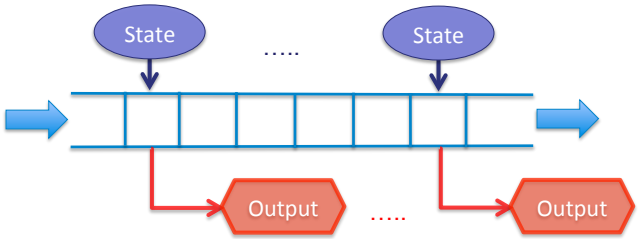

Figure 6: A CDS filter processes each data object in a separate step.

Pan privacy [11] guarantees that sensitive information that has been streamed through the filter will not be leaked even if the internal state is compromised, as illustrated in Figure 8. Pan privacy can be considered in distinct settings. In the first, the leakage of the internal state may be *announced* (when responding to a subpoena, for example). Alternatively, the compromise may be *unannounced*, in which case an adversary can access the internal state without any forewarning.

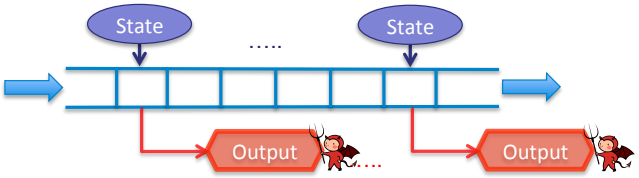

Figure 7: The output is assumed to be under *continual observation* by an adversary.

It is worth noting that increasing the power of the adversary decreases the possible guarantees that can be provided. In the MILS context, pan privacy is not necessary since the platform on which the filter operates would provide the necessary assurance that the internal state is not available to the adversary. However, since a filter that does not leak sensitive information after an intrusion may be of utility in a broader context, we consider the pan privacy case as well.

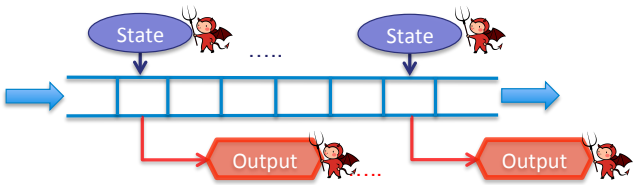

Figure 8: *Pan privacy* assumes that the adversary can gain access to the internal state of the filter.

Characterizing Leakage

To sanitize data that is being released to a lower security classification level, it is necessary to characterize the leakage of information. This is particularly challenging when the leakage itself may depend on auxiliary information that is available externally, but never observed by the downgrader. Intuitively, the amount of information that is being leaked corresponds to the quantity of data that must be removed from an unsanitized stream of objects S to yield a sanitized output stream S'.

S = axbxcxdxxxex

S' = abcdxe

Figure 9: *User-level X-adjacent* streams are identical if the elements in X are eliminated.

The notion of leakage can be formalized analogously to the way it is defined for streaming differential privacy. Instead of formulating leakage over query streams, we focus on streams of data objects. We consider streams S and S' to be *user-level X-adjacent* if the only difference between them is the presence of data objects $x \in X$. Figure 9 depicts two streams S and S' that are identical (as shown in red) after all occurrences of x (as shown in black) are eliminated. In this case, S and S' are X-adjacent. If only a single instance of each object $x \in X$ is replaced, then the two streams are considered to be *event-level X-adjacent*. Figure 10 depicts event-level X-adjacent streams that are identical (as shown in red) when a single element (shown in black) is eliminated.

S = abcdexfg

S' = abcdeyfg

Figure 10: *Event-level X-adjacent* streams are identical if each element in X is eliminated just once.

Composable Sanitization

A sanitization algorithm A is said to be ϵ-*differentially private* against continual observation if for all pairs of X-adjacent streams S and S'

$$e^{-\epsilon} \leq \frac{Pr[A(S) = \sigma_1\sigma_2 \ldots \sigma_t]}{Pr[A(S') = \sigma_1\sigma_2 \ldots \sigma_t]} \leq e^{\epsilon}$$

where $\sigma_1\sigma_2 \ldots \sigma_t$ is the output generated by running algorithm A on the streams S and S' [10].

Consider an algorithm A that operates in the streaming differential privacy framework. A maps elements of the stream to $I \times \sigma$, where I is the set of internal states of the algorithm A, and σ is the set of possible output sequences. A is said to be ϵ-*differentially pan-private* [9] (against a single unannounced intrusion) if for all event- or user-level X-adjacent streams S and S', inputs $I' \subseteq I$, and outputs $\sigma' \subseteq \sigma$

$$e^{-\epsilon} \leq \frac{Pr[A(S) \in (I', \sigma')]}{Pr[A(S') \in (I', \sigma')]} \leq e^{\epsilon}.$$

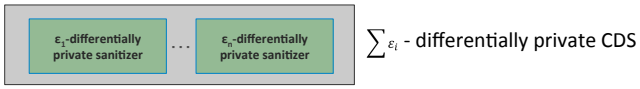

Figure 11: Differential privacy guarantees are additive over the collection of intra-CDS sanitizers utilized.

Algorithms that provide differential privacy guarantees can be composed while maintaining the assurance [10, 21]. This means that if an algorithm A_1 that provides an ϵ_1 differential privacy guarantee is combined with an algorithm A_2

Figure 12: Serial deployment of differentially private sanitizers results in an additive guarantee.

that provides an ϵ_2 differential privacy guarantee, the composition will provide an $\epsilon_1+\epsilon_2$ differential privacy guarantee.

If CDS downgraders implemented algorithms that provided analogous guarantees for streams of data (instead of streams of queries), it will be possible to deploy them in the intra-CDS configuration illustrated in Figure 11, the serial CDS configuration shown in Figure 4, or the parallel configuration depicted in Figure 5, while ensuring that the combination provides a formal sanitization guarantee.

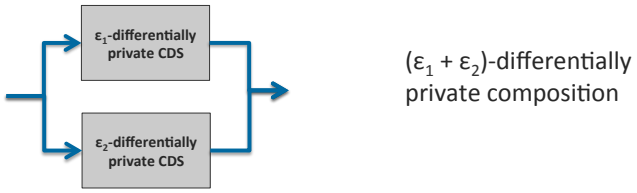

Figure 13: Parallel deployment of differentially private sanitizers results in an additive guarantee.

5. CONCLUSION

As the content and context of the data being transmitted in high-assurance systems continues to increase in complexity, the cost and time to certify cross-domain solutions is growing rapidly. We argue that downgrading functionality should be decomposed to the point where each filter provides a streaming differential privacy guarantee and its certification is economically viable. The resulting filters can be combined to provide equivalent functionality to that provided by monolithic downgraders. Of particular note is the fact that the streaming differential privacy guarantees of the constituent filters can compose.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant IIS-1116414. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

6. REFERENCES

[1] Nabil Adam and John Worthmann, Security-control methods for statistical databases: A comparative study, ACM Computing Surveys, Vol. 21(4), 1989.

[2] Jim Alves-Foss, W. Scott Harrison, Paul Oman, and Carol Taylor, The MILS architecture for high assurance embedded systems, International Journal of Embedded Systems, Vol. 2(3/4), 2006.

[3] Paul Beynon-Davies, Database systems, Macmillan, 2000.

[4] Carolyn Boettcher, Rance DeLong, John Rushby, and Wilmar Sifre, The MILS component integration approach to secure information sharing, 27th IEEE/AIAA Digital Avionics Systems Conference, 2008.

[5] Arthur Cebrowski and John Gartska, Net-centric warfare: Its origin and future, U.S. Naval Institute, 1998.

[6] Bee-Chung Chen, Daniel Kifer, Kristen LeFevre, and Ashwin Machanavajjhala, Privacy-preserving data publishing, Foundations and Trends in Databases, Vol. 2(1-2), 2009.

[7] Josiah Dodds, A development environment and static analyses for GUARDOL - a language for the specification of high assurance guards, Master's Thesis, Kansas State University, 2010.

[8] Cynthia Dwork, Differential privacy: A survey of results, Theory and Applications of Models of Computation, Lecture Notes in Computer Science, Vol. 4978, Springer-Verlag, 2008.

[9] Cynthia Dwork, Differential privacy in new settings, 21st ACM-SIAM Symposium on Discrete Algorithms, 2010.

[10] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy Rothblum, Differential privacy under continual observation, 42nd ACM Symposium on Theory of Computing, 2010.

[11] Cynthia Dwork, Moni Naor, Toniann Pitassi, Guy Rothblum, and Sergey Yekhanin, Pan-private streaming algorithms, 1st Symposium on Innovations in Computer Science, 2010.

[12] Boyd Fletcher, XML Data flow configuration file format specification, `http://iase.disa.mil/cds/helpful_tools/dfcf-specification_1_2_11.pdf`, 2008.

[13] Judith Froscher and Catherine Meadows, Achieving a trusted database management system using parallelism, Database Security II: Status and Prospects, North-Holland, 1989.

[14] Benjamin Fung, Ke Wang, Rui Chen, and Philip Yu, Privacy-preserving data publishing: A survey on recent developments, ACM Computing Surveys, 2010.

[15] GIG IA Architecture v1.1, `https://www.us.army.mil/suite/kc/13000401`

[16] Sameer Kumar and Kevin Moore, The evolution of Global Positioning System technology, Journal of Science Education and Technology, Vol. 11(1), 2002.

[17] Adam Lee, Parisa Tabriz, and Nikita Borisov, A privacy-preserving interdomain audit framework, 5th ACM Workshop on Privacy in Electronic Society, 2006.

[18] Ninghui Li, John Mitchell, and William Winsborough, Design of a role-based trust-management framework, IEEE Symposium on Security and Privacy, 2002.

[19] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam, l-diversity: Privacy beyond k-anonymity, ACM Transactions on Knowledge Discovery from Data, Vol. 1(1), 2007.

[20] Ashwin Machanavajjhala, Johannes Gehrke, and Michaela Goetz, Data publishing against realistic adversaries, Very Large Databases, Vol. 2(1), 2009.

[21] Darakshan Mir, S. Muthukrishnan, Aleksandar Nikolov, and Rebecca Wright, Pan-private algorithms: When memory does not help, 2010.

[22] Nancy Reed, Dave Bryson, James Garriss, Steve Gosnell, Brook Heaton, Gary Huber, David Jacobs, Mary Pulvermacher, Salim Semy, Chad Smith, and John Standard, Security guards for the future Web, MITRE Technical Report MTR 04W0000092, 2004.

[23] John Rushby, Design and verification of secure systems, ACM Symposium on Operating System Principles, Vol. 15, 1981.

[24] Richard Smith, Multilevel security, Handbook of Information Security, Wiley, 2006.

[25] Latanya Sweeney, k-anonymity: A model for protecting privacy, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, Vol. 10(5), 2002.

[26] Trusted Services Engine, Galois, `http://www.galois.com/files/TSE_Datasheet.pdf`

[27] U.S. Fleet Forces Command Briefing, Training - Cross Domain Information Sharing (T-CDIS) Summit, U.S. Joint Forces Command, Suffolk, VA, 28 October 2009.

[28] Jun Xu, Jinliang Fan, Mostafa Ammar, and Sue Moon, On the design and performance of prefix-preserving IP traffic trace anonymization, 1st ACM SIGCOM Workshop on Internet Measurement, 2001.