Privacy as an Operating System Service

Sotiris Ioannidis
si@cs.stevens.edu
Stevens Institute of Technology

Stelios Sidiroglou
stelios@cs.columbia.edu
Columbia University

Angelos D. Keromytis
angelos@cs.columbia.edu
Columbia University

1 Introduction

The issue of electronic privacy has of late attracted considerable attention. The proliferation of Internet services and, perhaps unavoidably, Internet crime, in conjunction with expanded government monitoring of communications has caused irreparable damage to the basic definition of privacy (the state or condition of being free from unwanted surveillance[1]).

Implementing privacy in personal computer systems has traditionally been the domain of the paranoid computer specialist. In order for basic privacy to become pervasive among the non-technical user base, we believe that it must imitate the usage of other successful security (and other) services. Services like filesystem encryption, email and web security are successful because they are invisible to the user. Other services (not related to security) such as backups, networking, file searching, *etc.*, also gain traction by being well integrated with the user's operating environment. In most cases, this means embedding such services in the OS.

In this work, we propose a new paradigm for implementing privacy, as an operating system service. We believe that privacy, similarly to other security services, is a service that has cross-application appeal and must therefore be centrally positioned.

Privacy as on operating system service has some clear advantages:

- Unification: Traditionally, functionality that needed to be shared across multiple applications was included in the OS, essentially forming the "ultimate" library.
- Transparency: One of the greatest advantages of having privacy as an OS service is the potential transparency to the end user since it removes the need of installing, configuring and managing third-party tools.
- Management: Privacy as an OS service allows for a single point of privacy scrubbing[2] and privacy-policy enforcement. Having a central location for implementing policy and privacy scrubbing greatly simplifies the administration of privacy. Furthermore, it can increase privacy robustness since scrubbing is not implemented by disjoint programs. Finally, a centralized location facilitates privacy maintenance as new privacy consideration become easier to integrate,

On the other hand, there are some potential problems arising from an OS-centered solution:

- Protocol Spanning: The operating system must have knowledge of the data and meta-data representation of applications. It needs to use this information to sanitize private information for each application in the system, or at least for those applications that the user has specified. For example, in order to scrub user name information in Microsoft Word and Open Office documents, the scrubbing module will have to be able to parse and according to policy remove user name references in both formats.
- Single Point-of-Failure: Adopting a centralized operating system approach introduces the risk of global failure. If the operating system has a fault in the way it sanitizes private information, all applications will be affected.
- Performance: It is possible that due to the centralized nature of an OS-center solution, that we might cause a performance bottleneck when executing privacy operations.

[1] Paraphrased from Apple's Dictionary application.

[2] The process of removing personally identifying information while maintaining the integrity of the data [16].

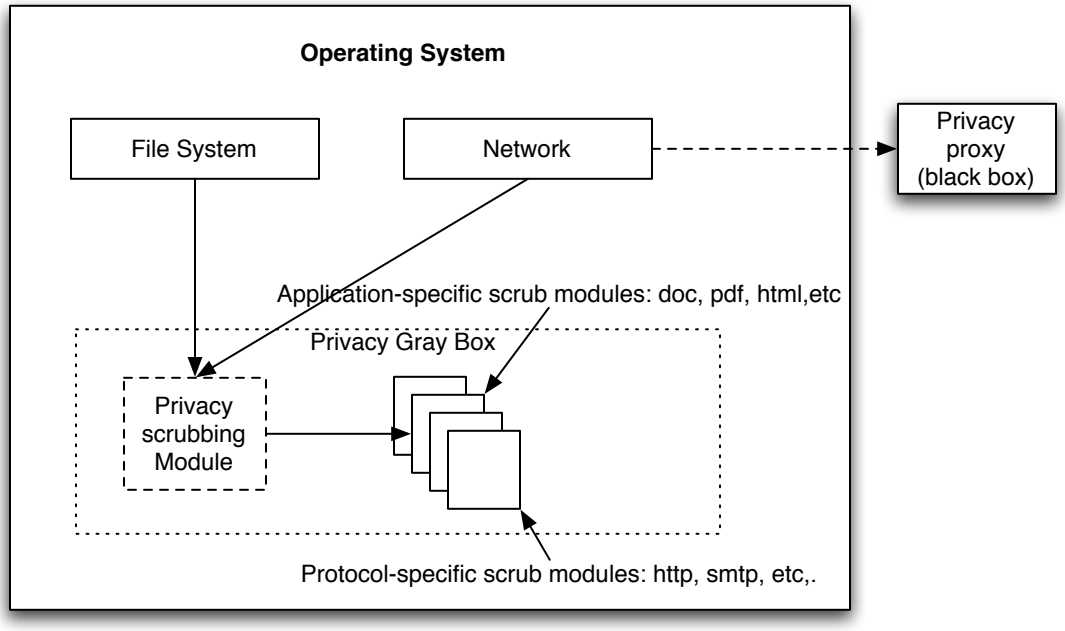

Figure 1: **Architectural overview: A privacy-servicing module lives inside the operating system. File and network requests as well as other relevant system calls are redirected to this module where scrubbing happens. The privacy module can be extended with application- or protocol-aware modules. For a purely network-oriented architecture, the privacy module can be a proxy "black box" that scrubs traffic on the fly.**

2 Approach

Having identified the benefits of providing privacy as an operating system service, the question that arises is what exactly will this service provide to the user and how will it be implemented.

We envision the following functionality for privacy enhanced operating systems:

- Transparent, privacy-providing storage and network: The file system and the network stack will include mechanisms that scrub data according to the privacy policy.
- Privacy-enhanced system calls: A privacy layer immediately below the system-call layer, where data can be scrubbed as they enter or exit the kernel.
- Privacy Proxies (Middleware): Network services without local support for privacy should be able to utilize proxies that provide such functionality.
- Privacy Libraries: Common code modules that allow application developers to integrate privacy-preserving services (a sort of "libwrap" [18] but for privacy).
- Privacy-policy Management: The OS must provide an interface that allows users and applications to specify various types of privacy requirements that must be provided.

2.1 Components

In the past, there have been a couple of attempts at implementing privacy at the system level, such as Domain Type Enforcement [6] and Mandatory Access Control [14]. Unfortunately, these systems were plagued by issues of complexity that rendered them unusable, especially for the non-technical user.

Services like Spotlight in Mac OS X [1] and Microsoft's upcoming Vista filesystem have facilitated the

process; they introduced application-aware filesystems through the use of meta-files or extensions. Given that there are mechanisms available to parse particular file structures, how can we use this ability to implement privacy?

There are a number of available tools that perform data scrubbing on a per file basis. The majority of these tools treat the file as a black box. For example, they will remove cookie, history, document history, log files, and preference files. These techniques work well when the user does not care about the posterity of the data, but they cannot address more complex information scrubbing where often the only course of action is manual removal.

Manually scrubbing files can be an extremely arduous process, as shown by a document published by the NSA, on how to safely publish reports that were created using Microsoft Word to PDF [3]. The highly manual process involves labor-intensive techniques such as creating a new Word file where the contents of the original report are pasted, sensitive areas replaced with garbage text and then exported to PDF. PDF offers the ability of selectively hiding text portions at the code level. However, when this is not coupled with encryption, the underlying code for hiding text can be easily bypassed [4].

2.2 Deployment

As illustrated in Figure 1, we envision a number of possible deployment architectures:

- White box: In this scenario, the privacy module resides inside the operating system. Network and filesystem requests can be intercepted or redirected to application- or protocol-specific scrubbing modules.

- Black box (proxies): Under this context, privacy modules are implemented completely outside of the operating system. One can envision subscription services where privacy filesystems can be mounted over network file storage, or network privacy proxies that scrub traffic on the fly.

- Gray box: Finally, we anticipate a scenario where virtual machines can be used to implement privacy services. This scenario is analogous to the use of middleware in other application domains.

2.3 What can we scrub?

Having argued about the benefits of privacy as an operating system service, the question becomes: what information can be scrubbed? Information scrubbing can be applied to three types of data: Meta-data, para-data[3] and "regular" data. Meta-data is a set of data that describes information about other data. For example, information that is visible to the operating system, such as file permission and access times. Para-data is a layer of data that the relevant tools need to understand application-specific parameters. For example, in order to remove superfluous information from Word documents, a scrubber utility would need to be able to parse the specific format. The problem is further exacerbated by having to support legacy and proprietary data formats.

- Meta-data: Removing meta-data, depending on the underlying technology, can be a straightforward operation. For example, copying a file generally does not replicate most meta-data in the new copy. There are, of course, exceptions, as is the case with the data aware `cp` command on HFS+ on Mac OS X.

- Para-data: This type of data typically requires labor-intensive scrubbing, primarily due to the fact that the removal tools need to understand application-specific parameters. For example, in order to remove superfluous information from Word documents (such as the author's username), a scrubber utility needs to parse the specific file format. The problem if further exacerbated by the need to support legacy and proprietary formats.

- Data: Implementing privacy scrubbing on raw data is a tricky proposition. In some ways, it is the most effective mechanism in the privacy-scrubbing arsenal, since it can remove delicate information such as names and social security numbers (SSN) according to some policy. Unfortunately, this flexibility carries a hefty price tag. First, data scrubbing would need to deal directly with a variety of application formats. Second, great care must be placed in the interaction with other applications that depend on the integrity of the data. For example, one would need to be careful with internal file checksums, and applications such as Tripwire and virus scanners.

[3] John Ioannidis suggested the term "para-data."

2.4 Using Privacy Policies

Given the subjective nature of privacy, the use of policy becomes a requisite. The success of projects like Platform for Privacy Preferences (P3P) for the web [9] has set the standard for privacy policy. Can something similar be established for operating system policy? What are the parameters that need to be addressed for operating systems? Can privacy policies be used for trust negotiation?

2.5 Trust Negotiation

In automated trust negotiation [17], participants specify access-control policies for the disclosure of credentials. The negotiation phase consists of a sequence of exchanges that are controlled by the access-control policies defined for the credentials. At each round, parties gain higher levels of mutual trust, permitting access-control policies for more sensitive credentials to be satisfied, which in turn enable these credentials to be exchanged. A similar approach can be applied to a privacy service, that is, participants can follow a series of rounds of releasing private information.

3 Discussion

Users are becoming aware, albeit slowly, of the privacy implications associated with the generation of copious amounts of data in their systems. A study by Garfinkel [12] shows that some users actually attempted to sanitize their discarded disk drives by formatting them. Of course, the current set of utilities (fdisk, format, newfs, delete, *etc.*) do not remove the data effectively [7], but this is an auspicious sign nonetheless. Another good indicator that things are heading in the right direction, is the effort to make cryptographic filesystems and secure erasing the default option, as is the case for Mac OS X.

3.1 When Crypto is not Enough

Users primarily rely on two approaches to meet their data privacy requirements. The first one is to encrypt their data by using some application or an encrypting filesystem (or device driver) [8, 10]. While this protects their data from being divulged, it does not fully protect their privacy. Admittedly, encrypted data leak almost zero information. However, encrypted data are of little use to users; they require to be decrypted to actually be useful (barring specific operations that can be performed to encrypted data [5]). Furthermore, during exchanges of information, the consumer of the data will eventually get them in the clear. It is therefore of paramount importance that data is scrubbed properly, and according to policy.

An alternative mechanism towards privacy assurance is explicit manual scrubbing of sensitive information [3]. A simple web search looking for such scrubbing tools reveals a large number of applications primarily designed to delete specific files from users' computers. Files include temporary Internet files, cookies, web browsing history, free space sanitizing, *etc.* While such tools are useful, their scope is limited. The advent of intelligent file systems that hold extensive meta-data, and the presence of files that contain both data and para-data, render existing tools and mechanisms obsolete.

3.2 What About Privacy-Preserving Operations?

Lately, there has been extensive research interest in the areas of anonymity, privacy-preserving data-mining, private queries, privacy-preserving set operations, *etc.* [19, 11, 13, 15]. While the proposal of our work targets the "systems" aspects of privacy (namely, how to architect systems to provide privacy for the average user) there are a lot of lessons to be learned from this theoretical work.

The primary benefit of that work is the cryptographic tools that we can use in an OS-supplied privacy service. We may be able to combine these tools with scrubbing to offer stronger privacy guarantees in the cases where we cannot completely eliminate sensitive data.

3.3 Policy and Mechanism

To be effective, solutions like the ones proposed in Section 2 must be accompanied with a powerful way of specifying the appropriate privacy policies. In the previous section, we focused our attention on the mechanisms for achieving privacy as an OS service, and outlined the possibilities of such an approach.

There are multiple opportunities and choices in this domain. From a user perspective, this should be the only point of interaction with the privacy services; the actual operation of the privacy service must be transparent. Towards this goal, there are several steps we can take. First, the operating system privacy module should be preconfigured as "privacy maintaining" by default. For exam-

ple, it should generate warnings, or automatically scrub, email messages when the user attaches files that contain private information. Second, it must have default policies on what constitutes private data, like SSNs, phone numbers, birth dates, *etc.* Third, it should clean up automatically generated information, like meta-data inserted in HTML pages by web editing tools.

Ideally, the operating system will export a privacy management interface similar to the one found in web browsers [2]. Through this interface, users would be presented with basic privacy options and the ability to use third-party plugins for specific applications.

4 Conclusions and Future Work

We have made the case for privacy as an operating system service. While more traditional approaches, such as scrubbing utilities and cryptographic folders, offer some privacy guarantees, they do not address the problem in a comprehensive manner. We believe that the operating system (kernel and system libraries) presents the ideal place for the deployment of privacy mechanisms that can be then utilized by a plethora of applications.

We have presented an initial description of how such privacy services would work and interface with user applications, as well as a discussion on defining privacy security policies that apply to user data. This work only scratches the surface of the *privacy-as-service* problem, leaving a lot of possible directions for investigation. For example, we still need to answer questions on efficiency, user interfaces, application interfaces, which we hope to investigate in the future.

References

[1] Mac OS X. http://www.apple.com/macosx.

[2] The Mozilla Project. http://www.mozilla.org/.

[3] Redacting with Confidence: How to Safely Publish Sanitized Reports Converted From Word to PDF. http://www.fas.org/sgp/othergov/dod/nsa-redact.pdf, December 2005.

[4] At&t leaks sensitive info in nsa suit. http://news.com.com/2100-1028_3-6077353.html?part=rss\&tag=6077353\&subj=news, May 2006.

[5] N. Ahituv, Y. Lapid, and S. Neumann. Processing Encrypted Data. *Communications of the ACM*, 30(9):777–780, 1987.

[6] L. Badger, D. F. Sterne, D. L. Sherman, and K. M. Walker. A domain and type enforcement UNIX prototype. *Computing Systems*, 9(1):47–83, 1996.

[7] S. Bauer and N. B. Priyantha. Secure Data Deletion for Linux File Systems. In *Proceedings of the 2001 USENIX Security Annual Technical Conference*, 2001.

[8] M. Blaze. A Cryptographic File System for Unix. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, November 1993.

[9] L. F. Cranor. *Web Privacy with P3P*. O'Reilly Media, Inc., September 2002.

[10] R. C. Dowdeswell and J. Ioannidis. The CryptoGraphic Disk Driver. In *Proceedings of USENIX Annual Technical Conference, FREENIX Track*, pages 179–186. USENIX, June 2003.

[11] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology - EUROCRYPT 2004*, 2004.

[12] S. Garfinkel and A. Shelat. Remembrance of Data Passed: A Study of Disk Sanitization Practices. pages 17–27, January 2003.

[13] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay — A Secure Two-party Computation System. In *Proceedings of the Usenix Security Symposium*, August 2004.

[14] M. Nyanchama and S. L. Osborn. Modeling mandatory access control in role-based security systems. In *IFIP Workshop on Database Security*, pages 129–144, 1995.

[15] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-Preserving Data Mining. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 439–450. ACM Press, May 2000.

[16] L. Sweeney. Replacing Personally-Identifying Information in Medical Records, the Scrub System. In *Journal of the American Medical Informatics Assoc.*, pages 333–337. Cimino JJ, 1996.

[17] W. H. Winsborough, K. E. Seamons, and V. E. Jones. Automated Trust Negotiation. In *Proceedings of DARPA Information Survuvability Conference and Exposition, Volume I, IEEE Press*, pages 88–102, January 2000.

[18] W.Venema. TCP wrapper: Network monitoring, access control, and booby traps. In *Proceedings of the 3rd Usenix Security Symposium*, September 1992.

[19] A. Yao. Protocols for secure computation. In *Proceedings of the IEEE (FOGS)'82*, 1982.