Information Protection via Environmental Data Tethers

Matt Beaumont-Gay
mattb@cs.ucla.edu

Kevin Eustice
kfe@cs.ucla.edu

Peter Reiher
reiher@cs.ucla.edu

Laboratory for Advanced Systems Research
UCLA Computer Science

ABSTRACT

Faced with an increasing number of incidents involving leaks of confidential data, it is clear that new data protection strategies are needed. We propose *Data Tethers*, a new paradigm which uses policies based on environmental factors to determine when sensitive data may be stored on a machine and when it must be encrypted or removed from the machine entirely. We discuss a number of example scenarios where existing data protection systems provide insufficient protection and Data Tethers would prevent data exposure. We also discuss a proposed implementation of Data Tethers, including a number of different environmental inputs.

Categories and Subject Descriptors: D.4.6 [Operating Systems]: Security and Protection—Access controls, cryptographic controls, information flow controls
General Terms: Design, Human Factors, Security
Keywords: Data security, authorization

1. INTRODUCTION

Several high-profile data exposure incidents in recent years have brought attention to the need for better protection of sensitive information. In 2006 alone, the list of such incidents included the theft of a laptop containing confidential data on 382,000 Boeing employees [28], the theft of a laptop holding personal information on 2400 residents of a Marine Corps base [2], the discovery during a drug raid of a flash drive containing sensitive data from Los Alamos National Laboratory [30], and the discovery of flash drives with U.S. military data for sale at a bazaar in Afghanistan [17]. These events occurred, in large part, because a legitimate user of the data gained access to the data, presumably in a secure environment such as a private office building, and then brought it into an environment where it was vulnerable to theft.

There are two key observations we take from these incidents. The first is that existing access controls on the data were insufficient to protect it. Any access controls that were in place did not provide sufficiently strong binding between the user and the user's access rights, so the attacker was able to gain access to the data. While full disk encryption may have offered adequate protection in some of these cases, it did not protect the data which was exposed via removable media.

The second key observation is that the data should never have left the secure environment in the first place—or at least not in plaintext. No office worker should be working on truly sensitive data in a public space. Perhaps it was truly necessary to transport the data on a laptop or flash drive from one secure location to another, but regardless, while it was in transit, the user did not need access to it.

These observations clearly demonstrate the need for a new means of data protection, which we propose in this paper: *Data Tethers*. Our thesis is that by restricting the environment in which data is accessible, data owners can be sure that the data will be protected by the more traditional security measures in place in that environment. One obvious question, then, is what exactly is included in our notion of an environment? A slightly less obvious, but very important question is what happens when the environment suddenly becomes insecure?

As a simple example (to be expanded upon below), consider sensitive data on a laptop. Policy specifies that the data may only be accessed in one particular physical location. What happens when the user picks up the laptop and walks out of the secure location? Worse, what happens if the laptop is put to sleep in the secure location and wakes up somewhere else? What if the sensitive data has been read into memory by some application and displayed on the screen? Engineering a system to deal with these issues will take some care.

2. RELATED WORK

Using environmental factors to control the accessibility of data is not a new idea. In fact, time of day as an access control input is used as an example in Saltzer and Schroeder's classic 1975 paper [29]. However, that work puts it in the category of arbitrary user-defined sharing controls, or "protected subsystems," an idea which never gained much traction.

Moyer and Ahamad revisit environments and access control in [23], which introduces Generalized Role-Based Access Control (GRBAC). They use an environmental role as part of their generalization of traditional access-control schemes. GRBAC is a step in the right direction; however, Moyer

NSPW'07, September 18–21, 2007, North Conway, NH, USA.

and Ahamad do not discuss access revocation, which is a significant concern in dealing with dynamic environments.

One of the most closely related works is by Covington et al. on access control in the Aware Home [9]. Building on GRBAC, Covington et al. give a number of example situations where the environmental context, filling an "environment role," is important to allowing or disallowing an action in the Aware Home. They discuss the problems of environment role revocation and secure and efficient capture of environmental data, which are directly relevant to Data Tethers.

We intend to leverage the GRBAC model of environmental roles in Tethers, as Covington et al. did in the Aware Home. However, the engineering challenges of applying environmental constraints to data protection are substantially different than the work accomplished by Covington et al. In particular, environment role revocation in the Aware Home is viewed as a performance concern, with a tradeoff between ease of revocation and more frequent checks of environmental factors. In Tethers, taking action when (or before) the environment becomes insecure is a key aspect.

Durfee et al. present a system related to Tethers in [12]. Posture-Based Data Protection (PBDP) only makes data available on a mobile device when the device can prove to a server that its security posture is acceptable. PBDP is also similar to network admission control [7], but adds an element of continuous monitoring such that if the device's security posture changes for the worse, data is made unavailable. The proposed mechanism for PBDP, a cryptographic file system with keys stored on a secure server, is very similar to the mechanism we propose for Tethers. One major difference between PBDP and Tethers is that Durfee et al. explicitly ignore the problem of users copying data out of protected files, thus eliminating any protection offered by the system. We take it as an assumption that users will do so, and so we design Tethers to include guards against such copying. In addition, PBDP does not take measures to prevent information leakage via other system mechanisms, such as sensitive data still being displayed on the screen after it should be unavailable. In Tethers, we consider that threat and others beyond just whether the data is readable on disk.

Corner and Noble's Zero-Interaction Authentication (ZIA) [8] is also a very closely related work. In ZIA, a user carries a short-range radio-frequency token, which stores a cryptographic key which is used to encrypt the keys that protect the user's data. When the computer wants to access an encrypted file, it must request that the token decrypt the appropriate key, which can only happen if the token is within radio range. The computer also polls the token so that when it moves out of range, data in memory is re-encrypted. Corner and Noble share many of the same concerns that we have, such as the weakness of traditional disk encryption systems and the difficulty of getting users to fully cooperate with security measures. In addition, they address a number of efficiency issues which are relevant to Tethers.

We consider ZIA to be a special case of a data tether. In fact, we may implement ZIA with Tethers as a means of showing the generality of our approach. ZIA's token proximity is an example of an environmental input as used in Tethers, and data encryption, as provided by ZIA, is one of the possible actions which may be specified by a Tethers policy. We also note that ZIA only protects files on disk; once the data is in memory, they note that it should be protected by some other means. While ZIA is certainly an important precursor to Data Tethers, the generality of Tethers is a significant conceptual improvement over ZIA.

A major class of security systems against which Data Tethers competes is full-disk encryption (FDE). The core idea behind FDE, and related OS-integrated systems like BitLocker [21] and FileVault [1], is to encrypt all data before it is written to the disk, and transparently decrypt it after the user has presented appropriate authentication credentials. The fundamental problem with FDE systems is that once the user has authenticated, often simply by entering a password at boot time, the only protection offered is against the hard drive being removed from the machine. An attacker who gains access to the user's login session will have complete access to the data. In addition, vulnerabilities have been found in specific implementations [38]. However, we note that FDE could be a component in a Tethers system, given a sufficiently powerful interface. Tethers could provide a key to the FDE system while the computer is in a secure environment, and tell the FDE system to forget the key when the computer leaves the secure environment.

Several companies have begun offering data leakage prevention products, including McAfee [19], Vontu [35], Verdasys [34], Reconnex [27], and Websense [37]. The fact that there is a competitive market in this space is a strong indicator that there is demand for systems which address the problem of data leakage. To the best of our knowledge, though, there is no public, objective information on the technology underlying these products, so we cannot evaluate how they might meet the goals of Data Tethers.

3. THREAT MODEL AND CRITICAL ASSUMPTIONS

The threat addressed by Data Tethers is an attacker who has access to the computer while the computer is in an insecure environment, and who wants to gain access to the data stored on the computer. One of our primary threats, and in fact the original motivation for the Tethers concept, is is an attacker who has physical access to the computer, and moreover has access to the user's session; e.g. the user left the screen unlocked, or the attacker has learned the user's login credentials via social engineering [16]. Attacks like this are at least as much of a concern as the laptop thefts cited in Section 1. The thieves in those instances probably did not have the user's login credentials. We also consider an attacker who has remote access to the computer, possibly with the user's credentials, but again only when the computer is in an insecure environment. Finally, since data exposure via removable media is a real concern, we consider means of preventing such exposure.

This threat model brings with it the assumption that an environment deemed secure is actually secure. Beyond just physical security, we also assume that there are strong network admission control [14, 7] and anti-virus measures in place on the network in the secure environment. That is, an attacker should not be able to access sensitive data by introducing malware into the secure environment, but enforcing this constraint is beyond the scope of Data Tethers.

A further assumption that informs our design is that our users are cooperative, but not necessarily security-minded. Conventional wisdom holds that most users, if asked to make a computer security decision, will make the decision that

gets them back to doing their work the fastest. We assume that the users of our system will not actively try to subvert the system—if the user is the attacker, there is little that can be done. But even given that the user is friendly, we must further assume that he might thoughtlessly compromise security for the sake of convenience. For instance, if there are simple ways to work around the protections offered by Tethers, thus offering more convenient access to the data, he will use these workarounds. Likewise, if the system requires him to take explicit action for purely security reasons, for instance unplugging a USB key when he walks away from the computer, he is unlikely to actually do so. Clearly, there is a continuum between the most careful and security-conscious user and the most malicious and diabolical one. We are not designing for either extreme of this continuum, but rather for the user that we consider to be the common case, somewhere in the middle.

For several of the proposed mechanisms of the Tethers implementation, we assume that the computer is connected to the network while in the secure environment. There are of course policies which make this untrue, so we discuss ways of relaxing this assumption in Section 6.

3.1 Motivating Examples

We illustrate the operation of Data Tethers with a few examples. These examples show particularly important scenarios, but do not set the boundaries of the system's goals. Our primary motivating example for Data Tethers is laptop theft. Alice uses the laptop at work, copying sensitive data onto it. She puts the laptop to sleep and goes to the airport, where Bob steals it. Bob has somehow acquired Alice's password (concordant with the above assumptions), so even if the screen is locked, he now has complete access to the data.

None of the various flavors of drive encryption do anything to prevent this attack. Because Bob doesn't reboot the laptop, he has effectively bypassed boot-time authentication. Likewise, because Bob can access Alice's login session, an OS-based disk encryption system will happily continue to transparently decrypt the data after Bob steals the laptop.

With Tethers, the data will remain protected. When Alice puts the laptop to sleep, the system assumes that it will be woken up in an insecure environment, so it encrypts the data with a key not stored on the machine, or possibly even removes the data entirely. Bob can get access to the data in the former case only by brute-forcing the key, and in the latter case, the data will of course be removed in such a way as to prevent forensic recovery.

Consider another scenario: Alice is working with the sensitive data in her office, and leaves the application running with the sensitive data on screen when she puts the laptop to sleep. When she opens up the laptop in the airport, Bob passively observes the data by looking over Alice's shoulder. This, in particular, is an attack against which FDE is powerless. Our intended design for Tethers will note that the application has read the sensitive data into memory, and will thus suspend the application (encrypting its suspended state) and tell the window manager to hide its windows.

Yet another scenario is after-hours data access. Suppose the secure environment is only really secure while there are workers in the office, during business hours. After hours, it is much easier for an attacker to sneak in and gain access to sensitive data. If the time of day is incorporated into the environmental constraint, Data Tethers will make the data unavailable after the end of the business day, for instance by encrypting it with a key stored off-site. This provides the digital equivalent of a time-locked safe.

Finally, suppose Alice wants to copy some files to a USB flash drive so she can work with them at home. She plugs in the drive and copies over the entire directory containing the files she wants. Unfortunately, that directory also contains some sensitive files. If Alice later loses the flash drive, she has unknowingly compromised the sensitive data. A Data Tethers policy could specify that the sensitive data be inaccessible whenever removable media is present, thus keeping it from being copied to the flash drive in the first place.

4. SYSTEM DESCRIPTION

In this section we describe some specific functionality that we envision as the core of a Data Tethers implementation. Data Tethers is a new operating system service, so much of the implementation will happen on the host that is actually running Tethers. However there are also some auxiliary components that will be external to that system. We consider the following particular subsystems to be particularly important: a cryptographic file system, a mechanism to identify and protect processes with sensitive data in their memory spaces, a component which interacts with the computer's power-management system, a policy engine, a user interface, a network server which provides various security services, and a component which gathers environmental information.

In the following discussion, we consider this scenario. Suppose there is a file f containing sensitive data d. f will be encrypted on host H's disk under a key K, and associated with some policy p regarding what environment E is considered secure for d. K will be kept only in memory on H, and will be erased when H leaves E. However, K will be stored on some other host S, for instance, a server in the secure environment. S will provide K to H when H returns to the secure environment. (This is one of the primary reasons why, in Section 3, we assume a network connection.) This is not the only possible model of operation for Data Tethers, but we will use it as a starting point for discussing an implementation.

4.1 System Components

Cryptographic Filesystem.

To avoid modifying applications, the cryptographic operations necessary to read or write the cleartext of f must be implemented in the file system. This is not terribly difficult; cryptographic file systems have been used for many years [4], and modern cryptographic file systems do not impose an undue performance penalty [39]. The most substantial challenge is gracefully revoking access when K is removed. It may be the case that K is copied to various locations in memory, e.g., multiple file descriptors; in that case, Tethers must ensure that all copies of K are removed when necessary.

Process Management.

When H decides that d may no longer reside on the system, part of the cleanup process must involve taking action against running processes which have accessed d. Deter-

mining which processes have accessed d is a matter of information flow tracking [24, 13, 41]. That is, if a process has read data from f, and then communicates with another process, the latter process is considered to have had access to d. Once the system knows that set of processes, it must somehow make them inaccessible. The naive approach is simply to kill them off. A less extreme technique is to suspend each process and encrypt all of its state with a key that will only be accessible from the secure environment (for instance, with K). This approach will obviously involve hooking into the operating system's process control and memory management subsystems. Finally, in an ideal world, Tethers could send each process a message through some high-level IPC mechanism, asking it nicely to clean any sensitive data from its internal state; this of course would require changes to the applications. In addition, for this approach to work, we would need some assurance that the application actually did the requested cleaning, perhaps by only allowing signed, audited applications to access d.

Power Management.

Because laptops are one of our major use cases, we consider it important to interact with the operating system's power management features. Particularly, Tethers needs to know when the system has been instructed to go to sleep, because it must assume that the computer will wake up in a minimally secure environment, and thus must take proactive steps to protect whatever sensitive data is present. In addition, any sensitive information or cryptographic keys stored in memory must be appropriately handled if memory is written to disk as part of the sleep process.

Policy Engine.

We want to allow a wide range of policies for Data Tethers, both in the set of allowed environmental inputs and in the actions that the system can take upon moving into a new environment. As discussed in Section 4.2, the environmental input system will be pluggable; part of the plugin interface will be a definition of the values of the environmental input that can be used in policies. A plugin may provide some indication of how confident it is in its determination of environmental state; this confidence level can be used as part of a policy. Policies should specify what should happen when an environmental input is unavailable; for instance, the system could assume that the environment is insecure, or it could use a different input or set of inputs in place of the unavailable one. While allowing arbitrary actions may also prove useful, we will likely define a particular set of actions, such as encryption or removal. We may use XACML [18] or Ponder [10] as the policy language.

User Interface.

We want Data Tethers to be usable by people who are not computer or security experts. Thus, some consideration must be given to its user interface. The major problem we foresee is those times when data is made suddenly inaccessible by some environmental change not obvious to the user. There should be a clear notification that the data is inaccessible due to Tethers, the particular environmental condition that changed, and some description of the secure environment E, so the user can try to reenter E if she wants to regain access to the data. For the latter part of the notification, it may not be desirable to give a full description of E, so as not to give too much information away to the attacker. The detail in which the secure environment is described may be a component of the policy.

Security Server.

For a number of purposes, we find it convenient to assume an external server S to which H has a mutually authenticated, confidential channel while H is in E. It is not necessary that all of these purposes be served by a single physical machine (and in fact it may be undesirable for reasons of reliability), but we will assume that case for simplicity of discussion.

Connections to S should be restricted such that when H attempts to authenticate to S, it can be assumed that H is within some security perimeter. Likewise, H may be able to use the reachability of S as an environmental input. Ideally, the data that H uses to authenticate to S should be tightly bound to H, as by a trusted platform module (TPM) [33]. A TPM could also provide attestations to S: for instance, an attestation that H is running unmodified Tethers code, or that H executed a virus-scanning program provided by S.

One major purpose for S is to act as a key escrow service for the keys which H cannot hold outside of E. As discussed in Section 6, one possible mechanism we consider is wiping sensitive data from H entirely; S can serve as the backing store for the data such that it can be restored to H when appropriate. Clearly, if S is performing either of these duties, it must be protected commensurate with the value of the data.

Other Components.

There are other low-level components that we consider desirable for a complete data protection system. These mostly fall under the category of information leakage protection, and thus may be provided by existing work in that area [25]. If possible, Tethers should have some control over the swap system, to prevent sensitive data from being swapped out unless encrypted [26]. Another memory-related concern is direct memory addressing (DMA); we want to protect against a process writing to removable media via DMA. Likewise, IEEE 1394's OHCI standard allows DMA from devices into host memory, which obviously could lead to an information leak or other attack [11]. It is also desirable that sensitive data not be sent over arbitrary network connections, so some controls on the network stack or the socket API may be necessary.

4.2 Environmental Inputs

A crucially important component of Data Tethers is the system that gathers environmental information. This system will have a pluggable design, to make it simple to add arbitrary environmental inputs. It will be tightly integrated with the policy engine, taking from it a set of environmental inputs which are relevant to active policies and reporting back the state of those inputs.

We propose a number of environmental inputs which we foresee as useful for expressing common Tethers policies. We consider particular implementations with an eye towards making it difficult for an attacker to wrongly convince the system either that it is in a secure environment (allowing access to data), or that it is in an insecure environment (causing a denial of service).

One crucial input is physical location. Securely and accurately determining a mobile device's location is a difficult problem; we hope to use techniques from the literature to provide location information to Tethers. While many researchers have tackled the problem of localization, fewer have looked at the problem with an adversary in mind. Capkun and Hubaux have proposed distance-bounding methods [5, 6], but these require significant infrastructure and special hardware. Another set of approaches uses location-limited sideband channels such as a temporary physical connection [31], infrared communication [3], or audio or visual channels [15, 20]. While some of these approaches may be vulnerable to repeater attacks, those attacks are generally mitigated by our assumption of physical security in the secure environment. Depending on the technique used, Tethers policies that involve physical location may need to map a region in an absolute coordinate system to a semantically meaningful location, such as an office.

Another useful environmental input is the current time. Version 4 of the Network Time Protocol can use the Autokey protocol [22] to provide cryptographic authentication for time synchronization messages. In addition, for mobile devices, location information should be used as a sanity check for the time zone setting; without secure time zone information, time of day cannot be a secure environmental input.

As mentioned in Section 2, Zero-Interaction Authentication is a special case of a data tether. The presence of the user, as signified by the presence of a specific hardware token with short communication range, is the environmental input in this case. The notion can be generalized somewhat: for instance, it may be useful to allow some data to be accessible only when a supervisor is in the office, or only when two users are present (the "four-eyes principle"). An external system, such as a physical access-control system or a personnel location system such as Active Badge [36], may provide this information to Tethers.

The set of currently running applications is another possible environmental input. For instance, peer-to-peer file-sharing applications are a concern for some enterprises, as a user may install such an application and unwittingly share the entire contents of his hard drive. A Data Tethers policy could include a whitelist of acceptable applications, and running an application not on this list would cause sensitive data to become inaccessible.

We might also be interested in using the presence of a specific peripheral device or type of peripheral to be used as an input. A secure "dongle" or specific USB key might serve as an enabling token, letting data be accessed or written. Conversely, we may wish to make sensitive data unavailable while any removable storage device is present, thus preventing the data from being written to the device. An active 802.11 or Bluetooth interface may be considered a security risk, and so the presence of such could be used as an environmental input.

There are many other types of environmental characteristics that could be used as input to Data Tethers. For example, the identity of the hosting infrastructure network is an interesting input. Many environments may wish to restrict data access only to devices connecting via a specific network. More generally, the presence or absence of a specific network entity, for instance a service or a specific node, could be used as input to Tethers. This could allow data only to be accessed when used in conjunction with a specific network service or other computer.

4.3 Feasibility of Implementation

We intend to build a working Data Tethers system in order to show that the engineering issues are tractable and to examine the usability issues. Our list of possible implementation platforms can be divided into two categories: commodity operating systems and experimental operating systems. Specifically, in the former category we place Linux, Windows, and the various flavors of BSD. The free operating systems have the advantage of being open-source and thus easily modifiable, while Windows is a more realistic target for getting non-computer experts to use the system. In the latter category, we particularly consider Asbestos [13] and HiStar [41]. We consider these experimental operating systems because they are designed around information flow tracking, which we consider one of the largest engineering challenges of Data Tethers. However, these operating systems are not suitable for day-to-day use, especially by non-expert users.

Given the above considerations, we conclude that augmenting a commodity operating system with information flow tracking, while nontrivial, will suit our purposes best. One of the major contributions of Security-Enhanced Linux [25] is a set of mechanisms for defining and enforcing static constraints on information flow, which Tethers could certainly leverage. However, static information flow tells us where sensitive data could have ended up; for our purposes, we want *dynamic* information flow tracking to determine where the data actually did end up. We may be able to use the techniques in [32] or [40] to accomplish this.

A major challenge to making a truly secure Tethers system is obtaining accurate and secure environmental data. As discussed in Section 4.2, there are methods for securing all of the various environmental inputs which we consider. However, some of these methods require special hardware or infrastructure support. With only commodity hardware, certain environmental inputs may be less accurate or impossible to implement securely.

5. TESTING

There are several important properties of a Data Tethers implementation that must be tested as the system is built. In particular, there are several areas of performance that are important for the usability of the system. One is the overhead associated with maintaining accurate knowledge of the current environment. This overhead must be measured, and ways found to reduce it if it turns out to be excessive. Another performance measure is the overhead due to cryptography during file accesses. This overhead should be commensurate with that found in existing cryptographic file systems. The last major performance concern is the amount of time required to put the system in a secure state when it moves out of a secure environment. If this amount of time is too large, a laptop might take significantly longer to go to sleep, and worse, there will be a larger window for an attacker to gain access to the data we are trying to protect. In general, since Tethers is a security system which introduces a new set of security-related behaviors, we must test its security properties—for instance, by assigning a "Red Team" to develop attacks against it.

6. OPEN ISSUES

We foresee encountering a number of issues in the process of turning Data Tethers into a working system. Consider a laptop which leaves a secure environment, thus causing some data to be wiped from the disk. When the laptop returns to the secure environment, should the data be restored as soon as possible, or should the system wait until the user actually tries to access the data? There are clearly cases where one approach will perform better than the other; it may be the case that the best control over this behavior is a heuristic or a per-file policy.

Consider a case where an application is displaying some sensitive data on the screen when the computer leaves the secure environment. Even if the application is (for instance) suspended and encrypted, the sensitive data will still be resident in the video buffer. Tethers may need to communicate with the display manager in order to truly clean up the data. This observation can also be applied to other output devices, though in practical terms, the video buffer is likely to be the main problem.

We now revisit the assumption that the computer is connected to the network when it is in a secure environment. While many of the use cases we envision have as their secure environment a network-friendly setting such as an office, we would like the system to be flexible enough to accommodate partially or completely disconnected operation. This presents a challenge. Without a way to send cryptographic keys to another computer, encrypting sensitive data risks either data destruction, if the keys are stored only in memory, or data compromise, if the keys are stored on disk.

Storing the keys on removable media would seem to be a straightforward solution, but our assumptions about user behavior suggest that the user will just leave the storage device attached to the computer, making the stored keys accessible to an attacker. The approach used by ZIA, storing keys on a device worn by the user, would also be viable for Tethers. We mention earlier that a possible protection mechanism is to remove the data from the disk entirely. For this technique to be useful, the data must be stored elsewhere—for instance, on the secure server—and restored to the host when the computer is back in secure environment E and has a network connection. The advantage of this approach is that the computer doesn't need a network connection to make data secure when it leaves E, though it does need to use the network to make the data available again when it reenters E. This approach can also be used for external data encryption key storage. Furthermore, it can be used for a technique using public-key cryptography: generate a keypair, giving the Tethers host the public key and the secure server the private key; when the computer leaves E, encrypt the data encryption keys under the public key, and when the computer returns to E, send the encrypted keys to the secure server to be decrypted with the private key.

7. CONCLUSION

In this work we present an overview of Data Tethers, a system designed to protect data against disclosure. We observe that existing data security protection systems, such as traditional access-control systems and full-disk encryption, are insufficient in many real-world cases. We argue that by only allowing sensitive data to be accessible in a secure environment (defined by policy), we can guard against many data exposure risks. By allowing arbitrary environmental inputs, Tethers can be used to deploy a number of interesting security policies. Implementing Tethers will not be trivial. Potential difficulties include securely gathering environmental information and carefully managing keys and decrypted data.

Acknowledgements

We thank all of the NSPW participants for their invaluable comments and discussion. We also thank Matt Schnaider for early discussions surrounding this idea and for helpful feedback.

8. REFERENCES

[1] Apple. FileVault: Increased security for your computer. `http://www.apple.com/macosx/features/filevault/`. Visited 2007/5/4.

[2] Marine base seeks missing laptop. *Associated Press*, Oct 2006.

[3] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *Proc. NDSS*, Feb 2002.

[4] M. Blaze. A cryptographic file system for UNIX. In *Proc. CCS*, pages 9–16, Nov 1993.

[5] S. Capkun and J.-P. Hubaux. Secure positioning of wireless devices with application to sensor networks. In *Proc. INFOCOM*, pages 1917–1928, Mar 2005.

[6] S. Capkun and J.-P. Hubaux. Secure positioning in wireless networks. *JSAC*, 24(2):221–232, Feb 2006.

[7] Cisco Systems. Cisco Network Admission Control (NAC) solution executive overview. `http://www.cisco.com/application/pdf/en/us/guest/netsol/ns466/c654/cdccont_0900aecd80557152.pdf`, 2006. Visited 2007/5/4.

[8] M. D. Corner and B. D. Noble. Zero-Interaction Authentication. In *Proc. MOBICOM*, pages 1–11, Sep 2002.

[9] M. J. Covington, W. Long, S. Srinivasan, A. K. Dey, M. Ahamad, and G. D. Abowd. Securing context-aware applications using environment roles. In *Proc. Symposium on Access Control Models and Technologies*, pages 10–20, May 2001.

[10] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The Ponder policy specification language. In *Proc. Policies for Distributed Systems and Networks*, LNCS, pages 18–39, Jan 2001.

[11] M. Dornseif, M. Becker, and C. N. Klein. FireWire: All of your memory are belong to us. `http://md.hudora.de/presentations/firewire/2005-firewire-cansecwest.pdf`, May 2005. Visited 2007/5/4.

[12] G. Durfee, D. K. Smetters, and D. Balfanz. Posture-Based Data Protection. Technical report, PARC, 2007.

[13] P. Efstathopoulos, M. Krohn, S. VanDeBogart, C. Frey, D. Ziegler, E. Kohler, D. Mazières, F. Kaashoek, and R. Morris. Labels and event processes in the Asbestos operating system. In *Proc. SOSP*, pages 17–30, Oct 2005.

[14] K. Eustice, V. Ramakrishna, S. Markstrum, P. Reiher, and G. Popek. WiFi nomads and their unprotected devices: The case for Quarantine, Examination, and Decontamination. In *Proc. NSPW*, pages 123–128, Aug 2003.

[15] M. T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, and E. Uzun. Loud and Clear: Human-verifiable authentication based on audio. In *Proc. International Conference on Distributed Computing Systems*, pages 10–17, 2006.

[16] Infosecurity Europe. Two thirds of workers reveal passwords for chocolate and a pretty smile. `http://www.infosec.co.uk/page.cfm/Action=Press/PressID=640`, Apr 2007. Visited 2007/5/4.

[17] J. Leyden. Afghan market sells US military flash drives. *The Register*, Apr 2006.

[18] M. Lorch, S. Proctor, R. Lepro, D. Kafura, and S. Shah. First experiences using XACML for access control in distributed systems. In *Proc. Workshop on XML Security*, pages 25–37, Oct 2003.

[19] McAfee. Data Loss Prevention. `http://www.mcafee.com/us/enterprise/products/data_loss_prevention/`. Visited 2007/10/29.

[20] J. M. McCune, A. Perrig, and M. K. Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. In *Proc. Symposium on Security and Privacy*, pages 110–124, May 2005.

[21] Microsoft. Windows BitLocker Drive Encryption. `http://www.microsoft.com/windows/products/windowsvista/features/details/bitlocker.mspx`. Visited 2007/5/4.

[22] D. L. Mills. The Autokey security architecture, protocol and algorithms. Technical Report 06-1-1, University of Delaware Electrical and Computer Engineering, January 2006.

[23] M. J. Moyer and M. Ahamad. Generalized Role-Based Access Control. In *Proc. International Conference on Distributed Computing Systems*, pages 391–398, Apr 2001.

[24] A. C. Myers. JFlow: Practical mostly-static information flow control. In *Proc. POPL*, pages 228–241, January 1999.

[25] National Security Agency. Security-Enhanced Linux. `http://www.nsa.gov/selinux/`. Visited 2007/5/4.

[26] N. Provos. Encrypting virtual memory. In *Proc. USENIX Security*, Aug 2000.

[27] Reconnex. `http://www.reconnnex.net/`. Visited 2007/10/29.

[28] L. Rosencrance. Boeing laptop with data on 382,000 employees stolen. *Computerworld*, Dec 2006.

[29] J. H. Saltzer and M. D. Schroeder. The protection of information in computer systems. *Proc. of the IEEE*, 63(9):1278–1308, Sep 1975.

[30] J. Seper. FBI eyes contractor in Los Alamos leak. *The Washington Times*, Oct 2006.

[31] F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Proc. International Workshop on Security Protocols*, LNCS, pages 172–182, 1999.

[32] G. E. Suh, J. W. Lee, D. Zhang, and S. Devadas. Secure program execution via dynamic information flow tracking. In *Proc. Architectural Support for Programming Languages and Operating Systems*, pages 85–96, Oct 2004.

[33] Trusted Computing Group. `https://www.trustedcomputinggroup.org/`. Visited 2007/5/4.

[34] Verdasys. Digital Guardian. `http://www.verdasys.com/digital_guardian.php`. Visited 2007/10/29.

[35] Vontu. `http://www.vontu.com/`. Visited 2007/10/29.

[36] R. Want, A. Hopper, V. Falcão, and J. Gibbons. The Active Badge location system. *Transactions on Information Systems*, 10(1):91–102, Jan 1992.

[37] Websense. Content Protection Suite. `http://www.websense.com/global/en/ProductsServices/CPS/`. Visited 2007/10/29.

[38] R.-P. Weinman and J. Appelbaum. Unlocking FileVault: An analysis of Apple's disk encryption system. `http://events.ccc.de/congress/2006/Fahrplan/attachments/1244-23C3VileFault.pdf`, Dec 2006. Visited 2007/5/4.

[39] C. P. Wright, J. Dave, and E. Zadok. Cryptographic file systems performance: What you don't know can hurt you. In *Proc. Security in Storage Workshop*, pages 47+, Oct 2003.

[40] W. Xu, S. Bhatkar, and R. Sekar. Taint-enhanced policy enforcement: A practical approach to defeat a wide range of attacks. In *Proc. USENIX Security*, pages 121–136, August 2006.

[41] N. Zeldovich, S. Boyd-Wickizer, E. Kohler, and D. Mazières. Making information flow explicit in HiStar. In *Proc. OSDI*, Nov 2006.