Idea: Trusted Emergency Management

Timothy E. Levin[1], Cynthia E. Irvine[1], Terry V. Benzel[2], Thuy D. Nguyen[1], Paul C. Clark[1], and Ganesha Bhaskara[2]

[1] Naval Postgraduate School, Monterey, CA 93943, USA,
{levin,irvine,tdnguyen,pcclark}@nps.edu
[2] USC Information Sciences Institute, Marina Del Rey, CA 90292
{tbenzel,bhaskara}@isi.edu

Abstract. Through first-responder access to sensitive information for which they have not been pre-vetted, lives and property can be saved. We describe enhancements to a trusted emergency information management (EIM) system that securely allows for extraordinary access to sensitive information during a crisis. A major component of the architecture is the end-user device, the security of which is enhanced with processor-level encryption of memory. This paper introduces an approach to more efficiently use the processor-encryption feature for secure data storage, as well as ISA instructions for the management of emergency state.

1 Introduction

During crises, first-responders can more effectively save lives and property if given access to certain restricted information relating to physical security (e.g., blueprints); individual privacy (e.g., medical records); and classified information (e.g., *continuity of government plans*), etc. However, the large number of potential first responders makes it infeasible to pre-screen them all, e.g. via national security clearances. Yet, if sensitive information is made available but not protected adequately, extensive damage could result. We describe a policy and operational model for emergency information management (EIM), and an architectural foundation for the realization of EIM in a modern IT environment where transient trust is possible [1]. We introduce a technique for leveraging processor-level encryption of data in memory to protect data in persistent storage; and we describe a set of new processor features for secure distributed state management, and describe how they are used to in the EIM context [2]. The target platform for this research is a dual-use hand-held computer, the *E-device*.

2 Model for Emergency Information Management (EIM)

In the emergency-response milieu we consider the following roles, each of which has a direct stake in effectiveness of the E-device. *First responders*, are, e.g., members of medical, police, fire, transportation, communication, construction, maintenance and other organizations, who may be called on at the scene of a disaster. *The Authority* is an organization that coordinates emergency response in a

given context, such as the Department of Homeland Security, non-governmental organization, or a selected enterprise department. The *Third Party* is one or more data providers that supply emergency information. As a simplification for our initial work, we consider Third Parties to be mutually trusting and are represented simply as a single Third Party. *Emergency information* is information designated to be available to emergency first responders, which they may not have been vetted or cleared to see.

Emergency information is *owned* by third parties, who may not wish it to be generally distributed or shared, or may be constrained from doing so (currently) due to regulatory hurdles.The E-device is initialized with certain emergency information, which can be updated in the field. The collection of E-devices, and the Authority's and the Third Parties' trusted systems, comprise the Emergency Network. For simplicity, we characterize the emergency state of the Emergency Network as either *on* or *off*. The Authority manages the emergency state, and communicates state changes to E-devices and Third Parties.

3 Transient Trust Policy and Stakeholder Trust Model

We assume a strict policy regarding the authorized accesses of users to data objects that is consistent across the emergency network. The Authority defines an emergency policy that allows additional, *extraordinary accesses* by end users to emergency information, which may occur only (transiently) during an emergency. While such accesses do not violate the security policy, per se, they are beyond the pale of usual MAC and DAC controls [3]. Together the strict policy and the emergency policy describe the complete emergency network security policy. The temporal constraint on extraordinary accesses is a key element of this policy, as it reduces the window of opportunity for inappropriate use of information resulting from adverse security events outside of the control of the trusted computing base, e.g., inadvertent password disclosure, or malicious insider behavior. The natural variability of mobile device connectivity means that the emergency network stakeholders must agree on a revocation policy should an E-device lose connectivity during an emergency, e.g., an upper bound on delayed revocation [4].

Operational agreements define the information sharing policies and levels of protection to be afforded to shared information, including the level of assurance provided by the E-device. The agreements may provide for Third Party *trusted* applications to be hosted in the E-device's Trusted Partition (see below). The Third Parties rely on the Authority to declare the start and end of the emergencies, and to correctly configure the E-devices, including communication keys.

4 Security Architecture for EIM

The foundation of our solution is the SecureCore security architecture [5]. Its Trusted Management Layer (TML) comprises the Least Privilege Separation Kernel [6][7] and the Trusted Services Layer (TSL) layer [8]. The LPSK partitions

the platform's physical resources and controls interactions between the partitions in the form of a lattice-flow policy; controlling access to data while it is in transit; at rest (on disk); and when processed (in both on-chip and on-board memory). The TSL layer virtualizes certain LPSK resources for the use of applications (e.g., commercial OSs), and associates MLS security labels with the kernel's exported resources.

Users have interactive sessions with one partition at a time. The Trusted Partition provides high integrity services, such as logon-on and partition selection provided by the Trusted Path Application (TPA), and other services that may be provided by Third Parties, such as for the secure sealing of documents. Normal Partitions host a commercial OS and typical office applications, providing familiar and functionally rich user interfaces. Emergency information is stored and accessed in the Emergency Partition. The TML ensures that the only way a user can ever access emergency information is through an Emergency Partition during an emergency. When an emergency ends, the TML renders the Emergency Partition inaccessible to the E-device user; and it can transmit updated emergency data to the Third Party. Now to our hardware protection scheme.

We utilize several hardware cryptographic primitives like those postulated for the Secret Protected (SP) processor [9] [3]. The processor provides a special "crypto mode", in which access to both the cryptographic primitives and several non-volatile processor registers (DRK, SRK and CEM) are available to software. Three crypto-transform functions are provided: `sp_derive` hashes two words with the DRK; `secure_store` marks a cache line for transformation (i.e., hash and encryption with the DRK) upon eviction from the processor cache; and `secure_load` decrypts memory as it is loaded into the processor cache and validates its hash. To this, we introduce the *code integrity check* (CIC) processor mode, which supports privileged *supervisor* program protection: at compile time, the TML code is hashed with the DRK, and when this code is executed the inline hash values are validated, and execution halts if the validation fails. As a result of using CIC, TML code can only be executed on the intended device, and any unintended changes to TML code are immediately detected, thus adding to the TML's high assurance self-protection mechanisms.

5 Secure Storage Solution

The `sp_secure_store` instruction is intended to be used for transient memory encryption. Yet, we also need to efficiently encrypt data as it is transferred between memory and the disk, and the built-in hardware encryption function appears ideal for this purpose. However, the `secure_load` instruction decrypts data as it is loaded into the processor. The solution presented here addresses this by marking data for encryption with `secure_load`, pushing it out of cache (e.g., with *clflush* with the x86 ISA), and then using device DMA to move it, encrypted, onto the disk. Alternatively, if it is desired to use programmed I/O

[3] While we characterize these features as part of the hardware instruction set, some may be suitably instantiated through an off-chip device [10].

to write to disk, data previously marked with `secure_store` can be moved from memory into the processor with a normal *load* instruction, which will cause it to arrive in the processor register encrypted. On re-accessing the encrypted data from disk, it is decrypted using `secure_load`, and its integrity can be validated relative to memory-segment and disk-volume seals generated upon storage.

6 Emergency State Management Solution

The security of the Emergency Network depends on how securely the emergency state is managed. Although TML functions to receive, store, and respond to emergency status updates could be implemented in software, to make emergency management more robust, we extend the processor ISA with key *state management* primitives: two new instructions, `hw_update_state` and `hw_get_state`; a local state counter: `e_counter`; and a state bit: `e_state`, described next [5].

The Authority associates a sequential number with each emergency state change. To announce a change to the emergency state, the Authority generates a point-to-point message for each E-device and sends them over a trusted channel. The message contains the new emergency state and the corresponding state-change number, as well as the hash of this payload. Each message is also encrypted with the target E-device's DRK (secure broadcast is left for future work). When the TML receives an emergency message it submits it to the processor using `hw_update_state`. The processor validates the message hash against the payload and decrypts it with the DRK; it checks that the new counter value is greater than the previous value, to prevent message replay; and finally writes the payload state to the hardware e-state register. If the `hw_update_state` operation is successful, the TML sends an acknowledgement to the Authority and uses `hw_get_state` to retrieve the new e-state value. The TML then either announces a new emergency to the user and makes the Emergency Partition available, or terminates the existing emergency.

7 Related Work

Our work utilizes currently-proposed concepts for CPU-based cryptographic support. Although mechanisms for cryptographic support using coprocessors are available, e.g. the IBM 4758 co-processor [11] and the TCG Trusted Platform Module [10], these off-chip schemes are more vulnerable to internal attack by elements within the platform architecture

Anderson proposed a model for patient medical information protection, which accommodates *extraordinary* access to information under certain conditions [12]. In this model, access control lists restrict access to patient records. Usually, changes to the lists require user consent; if an emergency results in an over-ride of the access list mechanism, the user is "notified." In contrast, our approach provides the ability to control when emergency overrides may occur, control the extent of emergency override, and revoke permissions to information in real time.

8 Conclusions

We have described a system for secure dissemination and control of sensitive information during crises, and two significant enhancements to emergency information management: transient-memory encryption for secure data storage, and new hardware instructions to support distributed emergency state management.

Acknowledgments. This material is based upon work supported by the National Science Foundation under Grant No. CNS-0430566 and CNS-0430598 with support from DARPA ATO. This paper does not necessarily reflect the views of the National Science Foundation or of DARPA ATO.

References

1. Irvine, C.E., Levin, T.E., Clark, P.C., Nguyen, T.D.: A security architecture for transient trust. In: Proc. of Computer Security Architecture Workshop, Fairfax, Virginia, USA, ACM (2008)
2. Levin, T.E., Irvine, C.E., Benzel, T.V., Nguyen, T.D., Clark, P.C., Bhaskara, G.: Trusted emergency management. Technical Report NPS-CS-09-001, Naval Postgraduate School, Monterey, CA (Naval Postgraduate School)
3. McCollum, C.J., Messing, J.R., Notargiacomo, L.: Beyond the pale of MAC and DAC: defining new forms of access control. In: Proc. of Symposium on Security and Privacy, Oakland, CA, IEEE Computer Society (1990) 190 – 200
4. Grossman, G.: Immediacy in distributed trusted systems. In: Proc. of Annual Computer Security Applications Conference, New Orleans, Louisiana, IEEE Computer Society (1995)
5. Levin, T., Bhaskara, G., Nguyen, T.D., Clark, P.C., Benzel, T.V., Irvine, C.E.: Securecore security architecture: Authority mode and emergency management. Technical Report NPS-CS-07-012 and ISI-TR-647, Naval Postgraduate School and USC Information Science Institute, Monterey, CA (2007)
6. NSA: U.S. Government Protection Profile for Separation Kernels in Environments Requiring High Robustness, Version 1.03. National Security Agency (2007)
7. Levin, T.E., Irvine, C.E., Weissman, C., Nguyen, T.D.: Analysis of three multilevel security architectures. In: Proc, of Computer Security Architecture Workshop, Fairfax, Virginia, USA, ACM (2007) 37–46
8. Clark, P.C., Irvine, C.E., Levin, T.E., Nguyen, T.D., Vidas, T.M.: Securecore software architecture: Trusted path application (TPA) requirements. Technical Report NPS-CS-07-001, Naval Postgraduate School, Monterey, CA (2007)
9. Dwoskin, J.S., Lee, R.B.: Hardware-rooted trust for secure key management and transient trust. In: Proc. of 14th ACM conference on Computer and communications security, Alexandria, Virginia, USA, ACM (2007) 389–400
10. TCG: TCG specification architecture overview. Technical Report 1.2, Trusted Computing Group (2004)
11. Smith, S., Weingart, S.: Building a high-performance, programmable secure coprocessor. Computer Networks **31** (1999) 831–860
12. Anderson, R.: A security policy model for clinical information systems. In: IEEE Symposium on Security and Privacy, Oakland, CA (1996) 30–43