



Computer Science and Artificial Intelligence Laboratory
Technical Report

MIT-CSAIL-TR-2008-016

March 17, 2008

Cognitive Security for Personal Devices
Rachel Greenstadt and Jacob Beal



Cognitive Security for Personal Devices

Rachel Greenstadt
Harvard University

greenie@eecs.harvard.edu

Jacob Beal
MIT

jakebeal@mit.edu

Abstract

Humans should be able to think of computers as extensions of their body, as craftsmen do with their tools. Current security models, however, are too unlike those used in human minds—for example, computers authenticate users by challenging them to repeat a secret rather than by continually observing the many subtle cues offered by their appearance and behavior. We propose three lines of research that can be combined to produce *cognitive security* on computers and other personal devices: imprinting and continuously deployed multi-modal biometrics, self-protection through virtualization and trusted computing, and adjustably autonomous security.

1 Motivation and Overview

The best tools, in the hands of a skilled user, act as extensions of the human body. A craftsman makes a hammer an extension of her arm, a driver makes a car an extension of his body. We treat personal computing devices this way as well, using them to extend our thinking, communication, and acting capabilities even when we are not paying direct attention.

Trusting computers this way has its dangers. Unlike most tools or body parts, a single bad interaction (with a website or piece of code) can thoroughly compromise a machine, giving control over its data, resources, and operation to an adversary. The security community has largely responded to this state of affairs by erecting barriers between the user and the device—more passwords, icons, dialog boxes, and warnings to increase their vigilance. This vigilance comes at the expense of convenience and productivity and yet is still brittle.

We propose that computers should act more like humans in their security decisions and characteristics. Human minds have evolved *cognitive security*: rich and subtle mechanisms for handling trust and security in social interactions. For example, when presented with authentication documents, human verifiers are trained to examine the humans as well as the documents themselves. The personal interview, harnessing all our subtle cognitive security traits, is still the gold standard for determining malicious intentions in humans [15].

Currently, we suffer from cognitive dissonance from trying both to treat computers as an extension of our-

selves and to be constantly vigilant of the ways in which their security models are alien to those in our minds. If we can build machines that manage their security more like humans then we may be able to avoid some of the damage caused by this cognitive dissonance.

Our focus in this paper is on personal devices that serve one primary user, not servers or machines which are primarily shared resources. While such shared resources could also benefit from having better cognitive security properties, we have chosen to start with personal devices because of the lesser complexity of having one master and because edge devices are where the primary security threats currently originate.

We propose advancing three lines of research that can be combined to achieve cognitive security for personal devices. First, imprinting and continuously deployed multi-modal biometrics, allowing a device to reliably recognize its owner. Second, self-protection through virtualization and trusted computing, allowing a device to protect itself while running untrusted applications. Finally, adjustably autonomous security, allowing a device to make security decisions with user input for important judgement calls. Progress in these areas will help produce devices that users can safely treat as extensions of themselves.

2 Differentiating Between Users

To serve its owner well, a device must be able to reliably differentiate between its owner and other users. Conventional methods for doing this are brittle, however, so we argue for two fundamental changes in how a device determines whether its user is its owner. First, rather than using a challenge/response protocol, we argue that a device should recognize its user through ongoing measurements of many streams of readily available biometrics. Second, we argue that no user should ever be allowed to set the identity of the owner; instead, a device should imprint on the first user it has a sufficiently rich interaction with, much like a baby bird imprints on the first sufficiently mother-like object it encounters. These two changes should allow a device to establish a durable privileged relationship with its owner.

2.1 Recognizing the Owner

How can Abacus, a device, tell that it is interacting with Alice, its owner? Conventional security designs generally use a challenge/response strategy. Under this approach, when Alice begins a session with Abacus, it challenges her to prove her identity, often using secret information such as a password. Once she has passed this challenge, however, Abacus never challenges her again that session, except perhaps after long periods of idleness. Although the challenge may be arbitrarily complex and difficult to fake—a password, biometrics, hardware keys, etc—the pragmatics of human usage tend to erode this type of security:

- Challenges interrupt the natural use patterns of many personal devices, such as PDAs and phones.
- Humans are bad at memorizing secrets: they often choose passwords that can be cracked by brute force or data-mining, and frequently reuse the same password on many different devices or services.
- Remote device use (e.g. calling one’s answering machine to check for messages) precludes the use of “physical” proofs like fingerprint scanning or hardware keys.

Worse yet, challenge/response authentication is fundamentally vulnerable to theft. If you can duplicate the bits that Alice will be challenged to produce, then you can pretend to be her with impunity.

Humans do not normally recognize one another this way: in most everyday interactions, we recognize people by who they are and how they behave, rather than by the secrets that they know. The cues we use for recognition range from immediate and obvious, such as facial structure, voice, and gait, to subtle and slowly emerging, such as fidgeting behavior and preferred topics of conversation. Nearly all, however, are based on streams of public information that are made available naturally throughout an interaction. Moreover, most users already view these sorts of biometrics as both acceptable and trustworthy[7], particularly for a personal device where privacy is not at stake.

2.1.1 Continuously Deployed Biometrics

We advocate a continuously deployed multi-modal approach, in which many different low-fidelity streams of biometric information are combined to produce an ongoing positive recognition of a user. With an optimistic and ongoing recognition process, security can be stronger, because attackers must impersonate Alice well enough to satisfy many different cues throughout their interaction with Abacus, and also less intrusive, because Alice can gain privileges merely by interacting with Abacus.

There are potentially many cues available to help Abacus recognize Alice. To name a few:

- Camera: invariant facial structure, height, posture, body shape, hair shape and color, complexion, glasses and eye color, tics and motion patterns
- Microphone: voice tone, inflection, pattern of speech and pauses
- Button Array (e.g. keyboard, phone keypad): typing speed, pause patterns
- Pointing Device (e.g. mouse, PDA stylus): smoothness of arc, idle/wander patterns, speed of traverse, duration and frequency of clicks
- Accelerometer: walking speed, changes in posture
- Use Patterns: music choice, favorite web destinations, choice of words and phrasing, patterns of use and idleness, preferences for button vs. pointer input, preferred times of day
- Other Sensors: body temperature, fingerprints, conductivity

Alone, none of these cues is likely to even approach either the consistency or reliability of a strong password. As an aggregate, however, they may out-perform challenge/response approaches.

Although many devices will have only a subset of these sensors, there are enough cues available that any large subset ought to provide enough information for a good recognition signal. For example, with 20 cues, each with an independent 20% error rate, a 2/3 vote has only a 1 in 500,000 chance of producing an incorrect decision.

The machine-learning subfield of ensemble learning[9] is explicitly focused on the problem of building strong classifiers from many weak classifiers, and has produced algorithms such as AdaBoost[4] that should be easily applicable to the problem of combining cues to provide good security while never denying Alice access to her own device. Existing work in multi-modal biometrics (e.g. [11], [3]) has already shown that machine learning can be used to fuse data sources and boost the efficacy of person recognition. In a continuously deployed multi-modal approach, we simply greatly multiply the number of sources and the time-frame over which they are considered.

Because cues are based on readily available information, the goal of a cue recognition system cannot be perfect security. It will always be possible for a sufficiently determined attacker to study Alice thoroughly enough and invest enough time and effort in counterfeiting to fool Abacus. Rather, the goal of cue recognition is to make the cost of doing so high enough that an attacker will almost always prefer a different approach: even if the attacker knows how Abacus recognizes Alice, sustaining an appropriate ensemble of cues is likely to be difficult.

2.1.2 Rich and Poor Cue Availability

When Alice and Abacus are physically co-located, the majority of cues are likely to be readily available. In a situation of this sort, the evidence that Abacus is interacting with Alice is likely to be overwhelming. As such, these periods of rich cue availability are ideal sources of training data, which Abacus can use to gradually adjust its model to reflect how Alice herself changes over time.

If Alice is interacting with Abacus remotely, however, most of the cues will not be available. If Alice called Abacus from an ordinary phone to get her messages, for example, Abacus would have only her voice, tones produced by button presses, and her use patterns to judge her identity. When the interaction between Alice and Abacus is short and few cues are available, it may be advisable to also use a password or similar challenge/response method.

2.2 Imprinting Abacus on Alice

Currently, devices are designed so that a sufficiently privileged user can easily change anything on the machine, including how it recognizes Alice. The only way to avoid this is to have Abacus' recognition of Alice be immutable.

There is a metaphorical similarity here to the psychological notion of *imprinting*. Many animals go through critical periods where they rapidly learn to recognize a special stimulus. Imprinted relationships are extremely durable, often lasting a lifetime. Geese, for example, imprint on first suitable moving object they see shortly after hatching, and will ever after treat it as their parent[8].

Likewise, when Abacus has its first long and cue-rich encounter with Alice, we argue that it should imprint on her. During that first encounter, it should learn to recognize Alice in as many different ways as possible, fixing her characteristics in a protected memory where not even Alice herself can modify it. Alice can thus trust that even a successful attack on Abacus will not compromise its relationship with her.

2.2.1 Loans and Sales

Imprinting need not make it any more difficult for Alice to loan out Abacus to her friends. If Abacus implements a reasonably fine-grained access control system, then it can adjust the privileges in real-time based on its ability to recognize its current user.

Transferring ownership, on the other hand, is a major shift, since only Abacus has access to its imprinting data. The new owner, Bob will not want an Abacus whose primary loyalty is to Alice rather than to him, so Abacus must re-imprint. Alice, meanwhile, will not want an Abacus that can re-imprint while carrying her private information. The only way out of this dilemma is for a

transfer of ownership to involve a “death and rebirth”—Abacus must reinitialize itself completely, destroying any data that Alice has on it, then imprint freshly on Bob. The difficulty of causing device death will likely depend on the type of device.

3 An Architecture for Machine Integrity

We envision Abacus as obedient to Alice's interests in an autonomic sense. Just as Alice's heart keeps beating even when she's sleeping, or not paying attention, or when she's so emotionally distraught that she thinks it is broken, so too should Abacus require minimal action from Alice to work. Currently it is just too easy for machines to be compromised for them to be trusted in this way.

Human security has some strong advantages over that of machines. Our human minds are basically inviolable; while it is easy to guess the gist of what someone is thinking or manipulate another person into doing what you want, it is impossible to actually read another person's mind or control their will. In contrast, computers are completely at the mercy of their users and application programming, making them vulnerable to attack and compromise by any rogue user, application, or remote exploit.

In humans, this security is effected by three features: (1) physical security in the form of the skull, the blood-brain barrier and limited access and inputs, (2) isolation of different regions of the brain and the ability of certain regions to examine and reflect upon the activity and thoughts of other regions and (3) a complex cognitive security system that acts to protect our secrets and self-control and manage relationships with other humans. We cannot replicate this security entirely in Abacus, but we can seek to gain some of its properties strengthening integrity, isolation, and secure decision-making skills in machines.

The problem is actually harder for Abacus than for humans because Abacus must be capable of protecting its integrity from Alice while still obeying her wishes. Alice will want to run the dubious JavaScript application and Abacus has to let her, or else she will greatly resent the intrusion. The key is to recognize that it is a risky or untrusted program and find a way to run it safely while still isolating the Abacus core and protecting the integrity of it and other critical data and applications.

Building the kind of trustworthy agent devices we envision requires some architectural support. Developments in trusted computing, virtualization, and instrumentation provide a strong basis for the development of more sophisticated security models. While there may be other ways to provide the needed security, this combination is already an active area of research (sHype, Xense, etc) [2].

Such an architecture is illustrated in Figure 1. The trusted computing base—the software which must be vulnerability-free to protect the system—is shown in yellow. The core of the system is protected by a trusted platform module (TPM) that protects key material and ensures the integrity of the virtual machine monitor (VMM) above it. The security manager, verified by the VMM below it, runs in a virtual machine that manages the rest of the VMs. Abacus can spawn virtual machines to run untrusted code in isolation and uses instrumentation to detect malicious actions on the part of this code. Without virtualization, trusted computing can attest to specific software configuration, but such systems are not easily changed and brittle; there is no way to run untrustworthy code. Virtualization alone is not enough either. Without trusted hardware, there is no root of trust to build the system on: the VMM could be resting on top of a stealthy root kit like Blue Pill[12].

While such architectures have been limited to PCs, we also envision security managers running on other personal devices, like phones and PDAs. Increasing numbers of phones are running full-scale operating systems, and virtualization is becoming more and more efficient and lightweight. The continuing march of Moore’s law will also make more layered approaches to security more feasible long term.

3.1 Trusted Computing

Trusted computing refers to certain hardware additions to the platform, including a secure co-processor that allows for (1) secure storage and memory and (2) attestation: the ability of running code to prove that it is running in some specific environment [16]. One of the key benefits of using trusted computing is the isolation from attacks it provides. Key material can be stored securely and maintain security even in the event of software compromise. Attestation by the secure core can be used to maintain the integrity of software executing on the machine (for example a VMM) and bootstrap more complex security layers in software from there.

Trusted computing, as expressed in the initiatives of the Trusted Computing Group (TCG) [1] formerly known as TCPA, and the Microsoft Next Generation Secure Computing Base (NGSCB, formerly known as Palladium) has been widely criticized for its potential to be used for unpopular DRM and ability to create software lock-in. However, most of these criticisms can be addressed by allowing owner override of the trusted platform’s functions [14].

Trusted computing enables secure boot by allowing each level of the machine (from bios and firmware up through the operating system and applications) to be measured and then stored as a hash on the TPM. These measurements are checked before control is transferred

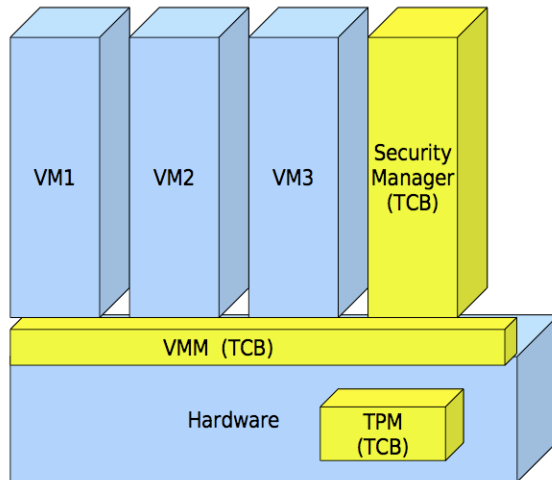


Figure 1: An illustration of the architecture for Abacus. The trusted computing base (TPM, VMM, and security manager) is shown in yellow. The rest of the hardware is untrusted, as are the other virtual machines for running applications.

to the next level and can be used to ensure that security-critical code is not maliciously altered [1]. Bootstrapping trust using a secure processor alone is quite brittle; any time the software or hardware configuration changes, the trust chain must be built anew. Virtualization provides a way to add additional flexibility and isolation using software. This is what NGSCB was trying to do with the nexus “secure right side” and the “insecure left side” on Windows.

3.2 Virtualization

Virtualization is the process of representing hardware resources abstractly in software. In its most extreme incarnation, emulation, all hardware is abstracted. There are also lightweight versions that only virtualize key system calls or drivers. While virtualization is most well-known for allowing multiple operating systems to run on the same machine, its security benefits are primarily gained by isolating pieces of software from each other. When the hardware (and some software) is abstracted, software running at different levels of trust can each be run on their own virtual machine. Different virtual machines are managed by a virtual machine monitor (VMM). If the VMM is secure (a large assumption but more plausible than operating system security) then each of these sets of software will execute in isolation, and a compromise in one VM will not affect the others or the VMM.

The basic architecture is illustrated in Figure 1. The hardware contains a trusted platform module, that can be accessed by a virtual machine monitor that manages a number of virtual machines. The security management

software will be run on a virtual machine on top of the VMM, so that the VMM can remain simpler and more likely to be bug-free. Nonetheless, the security manager will still be a part of the trusted computing base and need to be designed carefully, to avoid failures such as leaking biometric information about Alice or providing TPM owner override to an imposter. The security manager will have the ability to spawn virtual machines for the purpose of running applications on them and also for the purpose of testing the integrity of other applications and virtual machines. For instance, it is often possible to detect the presence of malicious code by running it in an instrumented virtual machine and taking note of suspicious actions [10]. In addition, root kits can often be detected by reconstructing the semantic view of the operating system from the virtual state in the VM in another VM and then comparing the outputs [6].

With good virtualization technology, it should be possible for much everyday activity to be handled safely by confining untrustworthy code to isolated virtual machines. It is important, however, that all these virtual machines have their I/O fused together to give Alice a seamless experience, rather than forcing her to navigate between them.

4 Adjustably Autonomous Security

If Alice understood what Abacus was being asked to do by the programs that she is running on it, she would usually be able to make good judgments about what behaviors are suspicious. For example, if Alice runs a program to send greeting cards to her friends, she would be happy to see it send those cards, upset if it tried to modify her browser, upset if it sent a list of her contacts or ads for Viagra to some random address, and might allow it to send its creator email with statistics on how many cards had been created.

Notice that these scenarios place us on the horns of a dilemma. On the one hand, these scenarios cannot be distinguished by what resources are involved, only by the semantic content of their usage, and we cannot expect Abacus to differentiate between them without help from Alice. On the other hand, computers do so many things so rapidly that asking Alice for many judgment calls will quickly overwhelm her and annoy her into disabling security.

What we need is a way for Abacus to filter and summarize its behavior so that Alice is only asked to make a few relevant judgment calls. For Alice to be able to rely on Abacus' judgment about what decisions need her input, Abacus needs to be able to judge how valuable and private a piece of information is, know what Alice is expecting a program to do, know what types of program behavior are worthy of suspicion, and simulate Alice's judgment so that it does not pester her when the

answer should be obvious. Although this is an unsolved problem, closely related problems are being studied in other domains, such as collaborative planning[13] and adjustable autonomy[5].

Any successful system will need to begin with a built-in knowledge base (containing facts like "money is valuable" and "sending spam is bad"), which is then modified as it interacts with Alice. Abacus will need a good model of how humans think about security so that it can infer Alice's likely judgments about new situations based on her reactions to previous situations. The same behavioral information used to recognize Alice might enrich this learning process as well—for example, guessing that information that causes stress in Alice is likely to be important to her.

Knowing what behaviors are suspicious is likely to be a harder problem and involve getting descriptions of program behaviors from some third party. Likely, Abacus will need some trust anchors that can provide machine-readable security advisories.

Finally, there needs to be careful user interface design that allows Abacus to give Alice information about the judgments that it is making, such that she can notice and react to them appropriately without interfering with her normal usage.

5 Conclusion

Cognitive security leads to better overall security because of the better match between a personal device and its user. This can be achieved by combining three lines of research: trusted computing and virtualization to protect the integrity of the device, imprinting with continuously deployed multi-modal biometrics to allow the device to recognize its owner, and human-like security models to enable responsible decision-making by the device. The greatest challenge is the last: human-like threat perception and response is a problem of potentially limitless complexity, and adversaries will exploit any systematic flaw in the design. Even a partial solution, however, may be effective enough to fundamentally change the trust relationship between users and their devices.

6 Acknowledgments

This work has been partially funded by NSF Grant #6898853 and the Harvard Center for Research on Computation and Society. Thank you also to Stephen Murdoch for information about the current state of trusted computing research.

References

- [1] The trusted computing group. <http://www.trustedcomputinggroup.org>.

- [2] S. Berger, R. Caceres, K. Goldman, R. Perez, R. Sailer, and L. van Doorn. vtpm: Virtualizing the trusted platform module. In *15th USENIX Security Symposium*, July 2006.
- [3] J. Bigun, J. Fierrez-Aguilar, J. Ortega-Garcia, and J. Gonzalez-Rodriguez. Combining biometric evidence for person authentication. In *Advanced Studies in Biometrics*. Springer, 2005.
- [4] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
- [5] Eric Horvitz. Principles of mixed-initiative user interfaces. In *Conference on Human Factors in Computing Systems (CHI)*, 1999.
- [6] X. Jiang, X. Wang, and D. Xu. Stealthy malware detection through vmm-based "out-of-the-box" semantic view reconstruction. In *ACM CCS*, 2007.
- [7] L. Jones, A. Anton, and J. Earp. Towards understanding user perceptions of digital identity technologies. In *ACM Workshop on Privacy in the Electronic Society*, 2007.
- [8] Konrad Lorenz. *Studies in animal and human behavior*. Harvard University Press, 1970.
- [9] Robi Polikar. Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):21–45, 2006.
- [10] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu. The ghost in the browser: Analysis of web-based malware. In *USENIX Workshop on Hot Topics in Understanding Botnets*, April 2007.
- [11] Arun Ross and Anil Jain. Information fusion in biometrics. *Pattern Recognition Letters*, 24:2115–2125, 2003.
- [12] Joanna Rutkowska. Blue pill project. <http://bluepillproject.org/>.
- [13] David Sarne and Barbara Grosz. Estimating information value in collaborative multi-agent planning systems. In *AAMAS 2007*, 2007.
- [14] Seth Schoen. Eof - give tcpa an owner override. *Linux Journal*, December 2003.
- [15] Andrew Simkin. Interrupting terrorist travel: Strengthening the security of international travel documents. <http://www.state.gov/r/pa/ei/othertstmy/84339.htm>, May 2007.
- [16] Sean W. Smith. *Trusted Computing Platforms: Design and Applications*. Springer, 2005.

