One user, many hats; and, sometimes, no hat—towards a secure yet usable PDA

Frank Stajano

University of Cambridge

Abstract. How can we design a PDA that is at the same time secure and usable? In current implementations the two properties are mutually exclusive. Because normal users find password entry inconvenient, the balance usually shifts away from security, leaving the PDA vulnerable if lost or stolen.

We begin by envisaging what an ideal PDA authentication mechanism might look like and by carefully examining alternatives to passwords such as tokens and biometrics.

We then expose another aspect of the security vs. usability problem. In many cases, when we turn on our PDA, we only access functionality (dictionary, calculator, web browser...) that requires no access to private data stored in the machine; why, then, should we pay the usability penalty of authentication in such cases? Moreover, we may want to grant another person temporary access to such "harmless" functionality, but without being forced to grant them unrestricted access to the whole machine.

To solve this problem we describe a system in which we may assign more than one "hat" to the owner of this single-user device, with each hat having specific privileges. The machine supports concurrent graphical logins for several hats and a convenient mechanism to switch between them. There is also provision for a userid associated with "no hat", to which one can switch without the need for authentication, and which can access all the harmless functionality. This scheme turns out to be applicable and useful well beyond the limited realm of PDAs.

1 Introduction

Nowadays, a usable and secure PDA[1] is an oxymoron. The two properties are more or less opposites: the usable PDA, like a paper diary, is accessible as soon as you open the cover; but anyone who finds it can read (and alter) all the data it contains. The secure PDA, instead, requests a password every time it wakes up, which makes it quite inconvenient for the legitimate owner to look up an appointment or address.

[1] Personal Digital Assistant, a pocket-sized computer with the primary function of an organizer. It usually works as a diary, address book and notepad. More advanced models may also act as an email client and web browser. The best modern PDAs have an open architecture and allow the user to rewrite the ROM with the operating system; some even run Linux natively.

B. Christianson et al. (Eds.): Security Protocols 2004, LNCS, to appear, 2005.

The amount of sensitive information entrusted to our digital butlers keeps growing [1]. The desire for a PDA that would stay secure even if lost or stolen is therefore justified. In practice, though, it is only hardened crypto-geeks who heroically withstand the password-induced inconvenience with quasi-religious fervour; standard human beings, instead, of which few use PDAs in the first place, immediately disable the password facility. Whenever security and usability fight, usability wins.

The first problem to be addressed is therefore that of finding alternatives to passwords that may offer greater usability for a comparable level of security. This issue will be examined in section 2.

Even if we devised a perfect authentication mechanism, though, we would still face another security problem. Consider the usage pattern of granting someone else temporary access to the non-private features of our machine, for example to allow them to play Tetris. Normally we have no way of doing so without also giving up control of the whole PDA. This scenario will be examined in section 3. Solving this problem will also, as a side effect, improve usability for the case in which the authentication method is still a cause of user frustration.

Finally, on the significance of the PDA platform: notwithstanding our own personal enthusiasm for it, we must reluctantly concede that the PDA is still a niche gadget of little commercial significance. Despite that, the quest for a secure and usable PDA is still a commercially relevant research topic in so far as most of the functionality of the PDA is now resurfacing in the computing product that, more than any other, can truly be described as ubiquitous—the mobile phone. Besides, much of what we shall say in section 3 also applies to laptops and even desktop computers.

2 Authentication

Imagine that, using telepathy, we had perfect authentication: imagine that the PDA could always tell, with absolute reliability and without the user having to do anything, who is currently holding it[2]. Then the usable and secure PDA could be implemented as follows: while my PDA is held by me, unlock it. As soon as it isn't, lock it. As you would expect, the correct way to establish this association in the first place is through the Resurrecting Duckling security policy model [2]: when I buy the PDA, I imprint it to me. Then nobody other than me could use my PDA (making it secure), whereas the PDA would always automatically unlock itself for me when in my hands (making it usable). In theory we would still have a problem with coercion[3] but the telepathic method may be assumed

[2] To be more precise, this hypothetical "perfect authentication mechanism" only needs to be able to tell whether the PDA is being held by its designated owner or not. It does not have to be able to recognize the identity of any *other* potential users, so long as it can reliably distinguish them from the owner.

[3] Cartoon image: the beaten-up banker, tied to a chair by the mafia thugs, has his PDA placed in his hand so that it unlocks and they can read his secret note about the combination of the safe.

to detect the duress condition and refuse access. The only problem is with implementation, because we don't know how to authenticate the owner to the PDA using telepathy.

Having drawn this asymptote of ideal behaviour, let us state our protection goals more explicitly. We want the PDA to be secure and usable for its owner. **Secure** here means that, if the device is lost or stolen, the effect is the same as if the device disintegrates: nobody can use the hardware any more, the owner loses any data not yet backed up, and nobody can retrieve any of the data that was on the machine, even if they have physical control of it (or of its remains). **Usable**, conversely, means that the owner finds the device unlocked every time she attempts to use it. She doesn't have to type a password into the device at every access. Ideally, just as for a paper diary, she doesn't have to type a password *ever*.

It is of course easy to provide each of these two properties on its own if the other can be ignored.

According to a traditional taxonomy of authentication methods, alternatives to passwords (*something you know*) include tokens (*something you have*) and biometrics (*something you are*). Either of these would be a step forward in the usability direction. They will be examined in sections 2.1 and 2.2 respectively.

We note from the requirements discussed above that the PDA is a machine with one very privileged user. In fact, even more than the so-called "personal" computer, the PDA is the archetypal *single-user* machine—so much so that the traditional login sequence for a PDA only asks for a password, without even *mentioning* a userid.

We also note that, to comply with the above security requirements, access control at the operating system level is by itself insufficient, since the attacker might open up the machine and access the secondary storage (flash memory or miniature hard disk) directly [3]. A secure PDA will have to encrypt its file system, although no current model does so in its manufacturer's configuration.

When should the authentication check be performed? In the ideal "perfect authentication" case it is a continuous ongoing process: as soon as the PDA is no longer held by the user, it notices and locks itself. In most practical implementations, instead, the check is only performed when certain (relatively rare) events occur: usually when the machine is initially turned on and when it wakes up from its sleep state. There is therefore a window of vulnerability between the moment in which the owner stops using the device and the moment in which the device goes to sleep. To reduce the size of this window one may force sleep soon after detecting inactivity; but, since the machine can't distinguish between thinking time and the end of the current burst of activity, setting an excessively short timeout will greatly inconvenience the user, as the machine will switch off in mid-operation. This is yet another trade-off between security and usability.

Assuming an authentication mechanism that allows the verification to be performed without disturbing the user, the dependency between sleep and authentication should be reversed: instead of triggering authentication when the device comes out of sleep, the device should continuously verify the presence

of the user[4] and lock itself up, as well as going to sleep[5], when the user is not present.

2.1 Token

The PDA could verify that it is being held by its owner by checking for the proximity of a token held by the owner, for example a special microchip embedded in a ring or in the strap of a wristwatch. The quality of this solution can be measured along several dimensions, including at least the following.

Communication technology. The communication between the token and the PDA may require precise contact between two physical devices (electrical connection, e.g. iButton; mechanical coupling, as with lock and key); loose contact between two designated areas (inductive coupling); it may be contactless but still require physical alignment (infrared pulses, barcode and other optical methods); and it may be completely unconstrained within a certain range (radio, e.g. Bluetooth or RFID). The latter solutions in this spectrum are almost as convenient as telepathic interaction but may be subject to eavesdropping.

Protocol. The most trivial authentication procedure is for the token to present a previously agreed secret to the verifying PDA. This solution (RFID, barcode) is cheap because it only requires a static, read-only token and unidirectional communication from token to PDA. If the communication channel is subject to eavesdropping, though, or if the attacker can once obtain access to the token, then replay attacks are possible. To prevent these, the responses from the token must not be predictable. Common solutions include one-time passwords, subject to synchronization problems, and challenge-response, requiring the additional device costs of bidirectional communication between token and PDA and active cryptographic capabilities in the token (iButton, wireless dongle). Furthermore, if an active adversary can insert itself in the channel between token and PDA and masquerade as one to the other and vice versa, then a man-in-the-middle attack may be possible. If all the man-in-the-middle does is tunnel data (to pretend that the token is near the PDA even while it isn't, for example to unlock a stolen PDA) then the only countermeasure may be a distance-bounding protocol [4].

Binding. All the authentication procedure can do is give the verifier some confidence that the *token* is nearby. To infer from this that the *owner* of the token is nearby requires a reason to believe that the token is still physically with the owner. If the token itself may be lost or stolen, the problem

[4] For example by polling several times per minute—although the ideal would be to have an "interrupt" generated (but by what?) as soon as the owner is no longer there.

[5] Sending the machine to sleep, as well as locking it, has however the disadvantage that it prevents the user from running unattended background computations. For some applications, e.g. backup, it may be more convenient to lock user I/O but only force sleep after the currently running applications have completed their task.

only goes up one level. While it is true that the user is less likely to lose a wristwatch-bound token than a pocket-heavy PDA, there are scenarios (e.g. sports practice) in which the user will have to take off both the watch and the PDA, and even have to store them together in a clothes bag or in a flimsy locker.

More complicated countermeasures could be envisaged to cover this threat, such as a token that deactivates itself when the watch strap is opened and which requires a PIN for reactivation; but the complication makes the implementation more expensive and goes against the original goal of usability. In particular: the armoured chip must sense the open-closed status of the strap; there must be a way to enter a PIN in the chip—and this data entry peripheral, being outside the tamper-proof enclosure, could be abused to brute-force the PIN; finally, the user has to re-enter the PIN whenever the watch strap is undone for any reason—this, although probably less frequent, is not an enormous improvement over having to type the password in the PDA in the first place.

An original idea on the theme of binding, although definitely not the solution we would hope to see deployed, is suggested in an excellent and thought-provoking "ubicomp envisionment" video co-produced by Keio University and Tokyo University[6], which features a teenager from the near future whose wireless e-wallet is embedded in a hard-to-misplace tongue piercing.

It is interesting to observe how, in a spectrum of "tightness of binding of token to owner" that might span such sample points as key fob, watch strap, ring, tongue piercing and subcutaneous RFID implant [5], the "something you have" token blurs into a "something you are" quasi-biometric, with the corresponding consequences on availability (the more tightly bound tokens are harder to forget at home or lose), transferability (the ability to sign a document in one's name, for example, is easier to delegate if the signature is affixed with a stamp, as commonly done in Japan, rather than as a handwritten scribble), deniability (the tighter the binding, the harder to pretend that the owner was not with the token) and privacy (particularly when recognition happens without the consent or knowledge of the owner).

The nicest existing implementation of token-based locking of a personal device that we know of is that of Corner and Noble [6]. The device being locked is a laptop, while the prototype token is a PDA, which the authors consider representative of what might be embedded in a watch in the near future. The two devices communicate over an 802.11 radio link in ad-hoc networking mode. The laptop, whose file system is encrypted, constantly polls for the presence of the token. When the token goes out of radio range, all the data in the file cache is re-encrypted within five seconds. When the token returns, decryption back to the original state occurs in just over six seconds.

[6] The short video, *A Love Triangle*, is part of a set of three "Small stories in 2008". At the time of writing, a streaming feed for the videos was available at `http://stoneroom.mlab.t.u-tokyo.ac.jp/vrep/archives/2003_12_08.html`.

We note in passing that, while in principle losing the token leads to inaccessibility of the PDA, if the binding between the two items is Duckling-compliant, then the PDA will remain accessible, because the second of the four principles of the Duckling policy [7, section 4.2] prescribes that the imprinting key be backed up. Alternative tokenless solutions might also be adopted, such as unlocking the PDA by typing a password into it. It would however be prudent to verify with care whether the chosen alternative is completely equivalent to the solution described by the Duckling policy and to look in detail at any differences.

2.2 Biometric

The biometric authenticator could be a fingerprint. There is indeed a model of cellular phone, first offered for sale in Japan in 2003, that incorporates a fingerprint reader and uses it to lock the address book and call history.

It is of course impossible to forget one's fingerprint at home, which is an advantage compared to the token, but it is not entirely impossible to have it stolen. Apart from the gory but thankfully infrequent scenario in which the actual finger is chopped off, it has been shown that most fingerprint readers can be fooled by a rubber-like mould [8]; all that remains is for the adversary who found or stole the PDA to get hold of a fingerprint from which to fashion the rubber finger. The James Bond approach might be to offer the victim a Martini and keep the glass; a more practical solution, which does not require interaction with the former owner, is to look for fingerprints on the PDA itself—it is, after all, a "handheld" device.

The biometric could also be a voice print. From the usability viewpoint this may be less convenient than silent forms of authentication: unlocking the PDA would embarrassingly attract the attention of anyone nearby. Compared to many other biometrics, though, it would offer the advantage of being better suited to a challenge-response protocol ("please read out this randomly-generated sentence") that might thwart replay attacks. Handwriting recognition, though probably still immature for authentication, is another biometric method that would share this property.

Another well known class of problems opened up by biometrics as opposed to other forms of authentication is that of the inevitability of false positives (accepting an impostor) and false negatives (wrongly rejecting the owner). The biometric scheme with the best performance along this dimension is currently iris recognition [9]. This scheme could be readily adopted for our application, particularly with the current trend towards camera-equipped phones. But here too, especially for a known victim, it wouldn't be impossible for the illegitimate holder of a lost PDA to find a photo of good enough quality for a replay attack, as suggested by the story of the Afghan woman featured on the cover of a 1984 National Geographic, identified 18 years later by comparing her iris codes to those of the photo [10]. To counter this threat one should resort to a verification procedure that ascertains the presence of a live iris, for example by measuring the pupil's dilation in response to varying illumination. The general consensus is

that it is hard to eliminate this kind of false positives if the attacker can perform the authentication without supervision.

Yet another problem of using a biometric measurement instead of a password or token is the impossibility to revoke it. You can't get new irises, except by following the grisly procedure demonstrated in Spielberg's *Minority Report*.

Perhaps the strongest threat for biometrics, therefore, is the current worldwide trend, under strong pressure from the US government, towards embedding biometric information such as fingerprints or iris codes in machine-readable state-issued identity documents. Ignoring a number of implementation details, this is more or less equivalent to having your non-changeable password printed in cleartext in your passport.

Much of the justified stigma associated with biometrics, though, comes from the Orwellian overtones of a Big Brother entity menacingly collecting privacy-threatening information about our activities and whereabouts. The case of PDA authentication, however, is different: here the verifier is under our control, works for our benefit and won't be reporting us to a global aggregating observer.

After pointing out the problems of tokens and biometrics it is only fair to emphasize that passwords, too, are far from perfect. Apart from the fundamental usability problem of having to enter the password at every interaction, which motivated the search for alternatives in the first place, passwords are often poorly chosen—easy to guess and, worse, recycled across systems, so that the discovery of one (which can be trivial for the provider of a password-protected web service) means the compromise of several accounts. Except for long and entirely random ones, passwords are subject to brute force and smart dictionary attacks (including "variations on a theme")[7]. When they are hard to guess they are also hard to type and remember, sometimes denying access to the device until the user can get hold of a backup (if one exists). The backup itself, when it exists, is usually better protected from the viewpoint of availability than from that of confidentiality and is therefore a vulnerability of its own. Finally, if fingers can be chopped off, so passwords can be tortured out of victims.

Just as most of us still confidently rely on passwords on a daily basis despite all of the above, the problems previously pointed out for tokens and biometrics should not be taken as implying that those solutions are entirely hopeless. Let us therefore suspend disbelief on the assumption that a non-password method can be sufficiently secure for protecting the confidentiality and integrity of the data in the PDA against the envisaged threats, and that the additional hardware cost is justified by the improvement in usability.

[7] On a PDA, though, where the attacker cannot crack a password file offline, brute-force attacks are hard to mount unless one interfaces an automaton to the keyboard, either mechanically or electrically. Even then, a throttling mechanism can easily reduce the frequency of attempts after only a small number of consecutive failures, making brute force impossible.

2.3 But do we need authentication?

The main thesis so far is that passwords are a nuisance from the usability viewpoint and that a user-friendly PDA should adopt some different mechanism for user authentication. This thesis is easy to defend but not particularly novel intellectually.

Another take on this problem is to step back and ask whether authentication is really needed. Some of the PDA functions we use most frequently, such as the multilingual dictionary, gain no benefit whatsoever from password protection. Yet we are forced to keep the password feature active in order to protect our diary, our personal notes and our contact list.

In previous work on authentication and authorization for ubiquitous computing we introduced the Big Stick principle [7, section 4.2.8], one of the most robust security policy models you can adopt: "whoever currently has physical control of the device is allowed to take it over". This is the security policy governing access to your lawnmower, your fountain pen and your English dictionary. In the world of electronic devices, it's the policy of your watch and pocket calculator, for which it is highly appropriate. It is also the policy of your digital camera, for which it is perhaps less so. The strength of this policy is that it is a very close model of what usually happens anyway; it is therefore less likely to be violated than others that attempt to impose unnatural behaviours on the real world: when the real and the ideal world are at odds, the real world usually wins.

The significance of Big Stick in the current context is that there are parts of a PDA that operate as devices for which that policy model is appropriate: the calculator application, the Japanese-English dictionary application, the multimedia encyclopedia, the World Time application, as well as most games and puzzles, carry no state and have no confidentiality concerns. If the PDA only offered those, Big Stick would be a perfectly suitable way to manage it and there would be no need to look for alternatives to passwords because one could dispense with authentication altogether[8]. The PDA would be ready for use all the time by the bearer with no access control restrictions. This is indeed what happens for separate appliances offering these functions—the stand-alone pocket calculators, electronic dictionaries and world clocks never ask you to log in, and are therefore much more convenient to use than the corresponding function of your PDA.

Especially with passwords, that emphasize the cost of every authentication operation, it is quite frustrating to have to pay the penalty of unlocking the PDA just in order to access a function, such as looking up a word in the dictionary, for which the access control restriction is meaningless in the first place.

[8] The only viewpoint from which Big Stick might be inappropriate is if one wanted to make the device inoperable for a thief—in other words, if one wanted to protect the hardware. In this paper we assume that the hardware will continue to drop in cost and that its value is insignificant compared to that of the data. If you lose your pocket calculator, even a scientific pocket calculator, you just buy a new one without great regrets.

2.4 Lending your machine—or part of it

There is more: if, seeing that you have just used an interesting-looking dictionary on your PDA, your dinner party companions ask you whether they can borrow the device and look up some words of their own, under the current state of affairs you can only hand them a usable device if you give it to them unlocked, i.e. in the state in which you have already logged in. At this point you must blindly trust your friends to access only the dictionary and not, while the device is in their hands and too far for you to actually see[9], the content of your diary, personal finance spreadsheet or secret business plan.

Things would be much nicer if we could draw a kind of security perimeter ("private area") around the data and applications of which we wanted to protect confidentiality and integrity, while leaving the rest of the functionality of the PDA in the "public area" (or perhaps "DMZ" in firewall terms) outside the security perimeter. Then anything in the public area could be accessed and run without authentication. Authentication would only be required for entering the private area. With this arrangement we could hand over the PDA without having logged in, thereby giving access to all applications in the public area but preventing access to anything in the private area.

Assigning applications to the private or the public area involves some subtlety. While the calculator can be made public with no second thoughts, the web browser requires some attention to detail: despite being essentially stateless, and therefore a good candidate for the public area, it might still contain sensitive information in the cache, history, bookmarks, cookies, "remembered fields" and so on. This kind of application should therefore be allowed to run in both areas, but it should access a different set of preferences in each. There should also be a way to erase in a global way any preferences left by any applications in the public area. Finally a third type of applications, such as backup and restore, should only be allowed to run in the private area.

The most obvious way to implement this split using standard operating system facilities is by making use of two user accounts, one for the private area and one for the public area, the latter equipped with an empty password so as to require no credentials for authentication.

As we noted earlier, the PDA is the archetypal *single-user* machine. Many PDAs don't even have a concept of user accounts in their operating system—my first one, an HP, just ran DOS. There are, however, PDAs based on a multi-user OS (my current one, a Sharp Zaurus, runs Linux), even though the supplied system software pretends and assumes that there is only ever going to be one user.

Is it then sufficient to pick a PDA with a multi-user OS, create a user called "public" (with empty password) and one called "private", and assign the right permissions to the relevant directories and applications?

Not quite. Because, if "private" is logged in and the PDA goes to sleep, the machine will be unusable until "private" types in his password to wake it

[9] And you are too polite to watch over their shoulder like a prison guard anyway.

up, so you can't suddenly open the PDA as "public" if you just want to use the calculator. Besides, if you are working as "private" with half a dozen applications open in useful places, you don't want to have to log out in order to be able to lend the machine in "public" mode to a friend who wishes to check the web briefly.

Would this be something solved by the unix "su" facility, or its Win32 equivalent "run as"? Not quite, because both of these are nested invocations: the person running the application under the new userid can always close the application and therefore switch back to the previous userid, the one that invoked the "su" or "run as" command. So this would perhaps be suitable for going from "public" to "private", but never vice versa.

Interestingly, any good solution to the sharing problem highlighted here would also fix the problem of the previous section (2.3): if you wanted to access the dictionary or any other application in the public area, you now would be able to do so without password.

3 One user, many hats

To solve the problem just presented in section 2.4, a different facility is needed: a multi-user OS that can keep several sessions (several "desktops") open at the same time and allow random-access, non-nested switching between them. This is similar to the CTRL-ALT-F1 console switching facility of Linux, with the two important differences that it should work for graphical sessions and that switching to a different user should require that user's credentials again. The closest approximation to this facility in a current desktop OS is Windows XP's Fast User Switching.

In order not to invalidate too many of the assumptions on which existing applications may be based we accept the restriction that, at any given time, each userid can have at most one active desktop. But several desktops may simultaneously be active, so long as they belong to different userids. One of these will be the one and only "public" userid—the one requiring no credentials. There may be several other accounts, all private in the sense of requiring credentials, each with a different set of privileges: this one can access the whole home directory but cannot access the network, this one can access the network but can only access a chrooted jail of the file system, this one can view Word files but can't do anything else, and so on. The underlying idea is sandboxing. This is clearly an arrangement that could be useful also outside the realm of PDAs. In fact, most laptops are just as single-user as PDAs.

The main point to note is that all of the "private" accounts actually belong to the same real-world user—the owner of the machine. These accounts form a group of different userids (we could say "personalities") for the same user. This is still a machine with a single real-world user, and yet there are several userids that are logged in simultaneously. One user, many hats. The hat can be viewed as a personality and also as a credential: the machine sees you wear the fireman's hat and recognizes you as a fireman; it sees you wear the Robin Hood hat and

recognizes you as Robin Hood, giving you access to Robin Hood's files and privileges. And sometimes it sees you come with no hat, as an anonymous user, and grants you the (non-null) privileges of an anonymous guest, which include looking up words in the dictionary, playing stateless videogames and surfing the web. Hence the title of this paper: *One user, many hats; but, sometimes, no hat.*

When the machine comes out of sleep, all active "private" sessions are locked—meaning that you need to exhibit the corresponding credentials to access them. The "public" session requires no credentials and therefore is never locked. When the machine comes out of sleep, you don't have to log back into the same session that was active when the machine went to sleep; you can instead switch to another session and log into that, leaving the original one locked. You can also start an entirely new session under an unused userid, assuming there are sufficient resources left to allow that. You can also atomically close all the sessions and reboot the machine without having to show any credentials. This is in accordance with the Big Stick principle, since you could always do that by removing the battery anyway.

This "switcher" accessory of the operating system can be operated without any credentials and always allows you to get back to the "public" session. It is invoked automatically when the machine is turned on or woken up, but it can also be recalled explicitly at any other time. In a well-designed PDA, the facility to switch to the "public" session would even be given its own physical button, which for the user would have the semantics of "turn on the machine *right now*[10], without any of that password hassle".

The "public" session, usable by anyone to whom the owner may lend the machine, does not retain any permanent state (browser history etc). When closed, it forgets everything. It is also always active: whenever closed, it reopens itself automatically, so that there is always an open "public" session to switch to.

With fingerprints as authenticators, one could even have a different userid for every fingerprint, with appropriate mnemonics: index to search Google, middle finger to read the dreaded Word attachments and so forth.

The baseline requirement, however, and probably the most useful configuration, is simply one private area and the common public area.

To get maximum protection from the sandboxing idea one might have to combine the OS facility described above with a carefully thought out mandatory access control policy. The goal might be to ensure that, for example, a virus coming through the mail program can never read the appointments in your diary. This, incidentally, will result in restrictions that prevent you from cutting and pasting data between the diary and the mail application. The need for smooth inter-application communication, not just through the file system but especially via higher level mechanisms such as the clipboard, is probably going to be the main constraint limiting excessive subdivision of the private area into a fine-grained multitude of separate userids.

[10] The "right now" may require some RAM preallocation, otherwise swapping may induce a substantial delay.

We fear however that, while the sandboxing idea will stop most of the malware attacks targeted at the pre-hats versions of the underlying OS, a determined attacker working specifically against the multi-hat configuration will in most cases be able to bypass the protection, possibly using techniques inspired by the API attacks community [11]. There are just too many possible interactions if the machine must still be usable and user-friendly.

We envisage a few ways of building a prototype of this functionality into a Linux-based PDA. The first is based on the X Window system. X already has the facility to host several independent graphical sessions on the same machine: some work is required to provide a suitable switcher applet, but thereafter each session will run on its own X display. A rather different approach would instead use the Xen hypervisor [12], which allows a machine to be split into several virtual machines each of which runs its own operating system. This would provide greater separation between the hats, allowing for totally separate file systems and peripherals (e.g. network). It would also require a port of Xen to the PDA's architecture and a port of the PDA's version of Linux to Xen.

4 Research questions

Once such a prototype is built, there will be further research questions to explore. It may already be possible to investigate some of them on a laptop by implementing the multi-hat strategy on top of Windows XP's Fast User Switching; however, with a closed proprietary OS, one has no freedom to adapt and possibly optimize the underlying mechanism towards the intended usage pattern and one cannot subsequently port the system from a laptop to a PDA. A solution based on free software would clearly be preferable.

Perhaps one of the most interesting research questions is how to reconcile asynchronous notifications with the security requirement of keeping everything encrypted. Imagine a PDA in which the whole file system is encrypted and data is only decrypted on the fly with a key that is forgotten as soon as the owner is no longer around, as in the cited work by Corner and Noble [6]. How will the PDA be able to wake up and beep at the next scheduled alarm event from the diary, if the diary file itself is encrypted and inaccessible while the machine is asleep? Should there be a special queue of wake-up events, kept around in unencrypted format, to which the diary application writes before going to sleep? This seems ad-hoc and inelegant.

If one could invoke the silver bullet of tamper resistance, and rely on a safeguard capable of erasing the RAM at the first attempt of tampering, the solution would be easy: the file system on secondary storage would always stay encrypted but the RAM and file cache would be unencrypted and would allow programs to run in the background (assuming they had preloaded all the disk pages they needed) even with the machine (meaning actually the user interface) locked. This may be the best compromise yet, but we have learnt to look at tamper resistance claims with some suspicion [13].

Furthermore, how would an encrypted file system interact with the multiple hats? Clearly, anything in the "public" area, including OS and applications, would have to be unencrypted. Would this then threaten the integrity of the "private" area? Should there be several complete replicas of the file system (an approach that the Xen-based implementation would support better than any other)? Is there any scope for a less drastic `chroot`-style sandboxing?

We may also have to reconcile the need for secure isolation between the hats with the potential usability requirement of transferring information between them in some controlled but convenient way. If, for example, the web browser runs only in the public area, it may be desirable to transfer a file downloaded with it into one of the "private" areas. Clearly this raises a number of issues from the field of multilevel secure systems—not trivial to begin with, but made much harder by the additional requirement of usability. Once in that realm, it will then be interesting to discover and counter the creative ways in which a no-hat user might escalate privileges.

5 Conclusions

Today, security and usability still tend to be antithetic. For a PDA, they roughly translate to "password" and "no password" respectively. Neither property is satisfactory without the other.

Performing the authentication with means other than passwords is one way to increase usability. Tokens and biometrics both have some problems, but ought to be taken more seriously. In this paper we examined a number of alternatives in some detail, exposing their security vs. usability trade-offs.

Whatever the authentication method, though, it would be nice if the machine were smarter and less pedantic. Why ask for authentication when accessing a facility that does not require protection? Our "hats" approach addresses this problem and, as a side benefit, also supports the otherwise dangerous usage pattern of temporarily lending the machine to a friend.

Even a strongly single-user machine such as the PDA will benefit from our new arrangement in which the single real-world user can have one hat, or several, for accessing protected material, and can switch hats—or take off the hat altogether—without logging out, and without having to provide any credentials when switching to "no hat".

When one day our machines feature a correctly implemented one-touch "no hat" button, we will have won a significant usability battle without having compromised security.

6 Acknowledgements

As usual, the original draft was improved thanks to the many comments received at the workshop. I am grateful to, among others, Matt Blaze, Tuomas Aura and particularly James Malcolm. After the workshop I also benefited from discussions with Matt Jackson, who has since become my PhD student.

References

1. Frank Stajano. "Will Your Digital Butlers Betray You?" In Paul Syverson and Sabrina De Capitani di Vimercati (eds.), "Proceedings of the 2004 Workshop on Privacy in the Electronic Society", pp. 37–38. ACM, Washington, DC, USA, 28 Oct 2004. ISBN 1-58113-968-3.
2. Frank Stajano and Ross Anderson. "The Resurrecting Duckling: Security Issues in Ad-Hoc Wireless Networks". In Bruce Christianson, Bruno Crispo, James A. Malcolm and Michael Roe (eds.), "Security Protocols, 7^{th} International Workshop, Proceedings", vol. 1796 of *Lecture Notes in Computer Science*, pp. 172–182. Springer, 2000. ISBN 3-540-67381-4. ISSN 0302-9743. `http://www.cl.cam.ac.uk/~fms27/duckling/`.
3. Tony Sammes and Brian Jenkinson. *Forensic Computing: A Practitioner's Guide*. Springer, 2000. ISBN 1-85233-299-9.
4. Stefan Brands and David Chaum. "Distance-Bounding Protocols (Extended Abstract)". In Tor Helleseth (ed.), "Advances in Cryptology—EUROCRYPT 93", vol. 765 of *Lecture Notes in Computer Science*, pp. 344–359. Springer-Verlag, 1994, 23–27 May 1993.
5. Julia Scheeres. "Implantable Chip, On Sale Now", 25 Oct 2002. `http://www.wired.com/news/privacy/0,1848,55999,00.html`.
6. Mark D. Corner and Brian D. Noble. "Zero-interaction authentication". In "Proceedings of the eighth Annual International Conference on Mobile Computing and Networking (MOBICOM-02)", pp. 1–11. ACM Press, New York, Sep 23–28 2002.
7. Frank Stajano. *Security for Ubiquitous Computing*. John Wiley and Sons, Feb 2002. ISBN 0-470-84493-0. `http://www.cl.cam.ac.uk/~fms27/secubicomp/`.
8. Tsutomu Matsumoto, Hiroyuki Matsumoto, Koji Yamada and Satoshi Hoshino. "Impact of Artificial Gummy Fingers on Fingerprint Systems". In "Proceedings of SPIE", vol. 4677, Optical Security and Counterfeit Deterrence Techniques IV. 2002. `http://cryptome.org/gummy.htm`.
9. John Daugman. "How Iris Recognition Works". *IEEE Transactions on Circuits and Systems for Video Technology*, **14**(1), Jan 2004. `http://www.cl.cam.ac.uk/users/jgd1000/csvt.pdf`.
10. John Daugman. "How the Afghan Girl was Identified by Her Iris Patterns", 2002. `http://www.cl.cam.ac.uk/users/jgd1000/afghan.html`.
11. Mike Bond and Ross J. Anderson. "API-Level Attacks on Embedded Systems". *IEEE Computer*, **34**(10):67–75, 2001. `http://www.cl.cam.ac.uk/users/mkb23/research/API-Attacks.pdf`.
12. Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho and Rolf Neugebauer. "Xen and the art of virtualization". In "Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP'03)", pp. 164–177. ACM, Bolton Landing, NY, USA, Oct 2003.
13. Ross Anderson and Markus Kuhn. "Tamper Resistance—A Cautionary Note". In "Proc. 2^{nd} USENIX Workshop on Electronic Commerce", 1996. ISBN 1-880446-83-9. `http://www.cl.cam.ac.uk/~mgk25/tamper.pdf`.