

RE: mlock(1)

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-09/8636.html>

From: Robert White (*rwhite_at_casabyte.com*)

Date: 09/29/04

To: <jonathan@jonmasters.org>

Date: Tue, 28 Sep 2004 20:46:20 -0700

(Sorry I am using outlook here at work by corporate mandate so my responses are tediously at the top... 8-)

I guess the first question to ask is who or what are you actually protecting the laptop against, what is their level of commitment, and what on the laptop are you trying to protect.

So here are my assumptions for the model:

- 1) You have set your laptop not to boot from CD etc, and protected that option with a "bios settings password".
- 2) You are making intelligent decisions about "how secure" your computer needs to be.
- 3) The thing(s) we are trying to protect in the swap space are the ram images of the running programs (etc) because the running system is secure to some acceptable degree.
- 4) The system doesn't have to be "perfect" (because none are) it just has to be "better" than the amount of effort the attacker is willing to expend.
- 5) Once the system is running again, security isn't "our problem" with respect to ingress. That is, we cannot really protect against a hardware-level attack (like the "monitor boards" that can halt a CPU and let you read through memory using the add-in card.
- 6) Once the system is running again, security isn't "our problem" with respect to the software that is now "restored"; that is, if you are running "gaping security hole 2007" on the active screen when you initiate suspend, when the laptop is resumed it is Somebody Else's Problem(TM) to lock the screen or kill the app or something.
- 7) Not everybody feels the same need for the same level of security. There are people who think their data is worth a power-on password, some would only presume it is worth a "resume" password, and some think a locking screen-saver is an onerous burden. So the "measured response" has to be one that each user must be able to live with. That is, there **IS** nothing wrong with requiring a resume password... except

for the people who don't think their data is worth the hassle.

8) This is **NOT** a general startup security solution, it is targeted to cryptographically encoding swap and preventing some of the most obvious restore boo-boos. That is, there is no attempt in `_this_` code to keep a user away from the login prompt etc.

9) It is desirable to know that when you resume a kernel (etc) that you are resuming the same kernel, and that it was resumed on the same box, etc.

10) A smart guy with a good soldering iron and an eprom burner and a bunch of time will be able to defeat any of this because he can simply read the memory of the live system by adding in any piece of hardware that can do a bus-fetch. (e.g. PCI bus-mastering etc).

11) In a system that never has to "resume from suspend" you can create a unique cryptoswap key at every boot and then throw it away.

12) Any "restore" methodology can be attacked by definition because the attacker can probably contrive to get an executable into the FS image of something "long enough" to run the `setuid` program he added to the disassembled system. So without a cryptographical file system, you don't get **too** much from restore protection.

13) The "worst case" of a failed authentication run is a normal boot, so the storage requirements are **quite** ephemeral.

14) The only "real" goal for a protected restore is to attempt to make sure that the restore operation is taking place in a context that is identical to the context of the suspend. So your real goal is to try to make sure that this boot is just like the last.

15) This would only be one link in a secure chain. The weakest link is preventing low-level hardware access.

16) It is pretty safe to assume that the average non-government-agency non-industrial-espionage "involuntary computer reassignment" will not involve someone who knows how to replace the clock chip without losing the bios config contained there-in. (etc.)

... That Said ... 8-)

You mis-apprehend (or I mis-communicate 8-) the key generation a bit.

In a system that never does a "resume" the meta-boot-block key generation tidbit never reaches the disk. A region of memory is filled with a template bit of code which reads ranges of memory and disk, combines the data read with a bit of entropy, and, for the thorough, asks the user for a password. The data thus harvested is used to compose a key. The clever bit is that the entropy is added to the block externally before it is invoked to do its job. So we have this little applet that we use once and set aside.

Linux-Kernel: RE: mlock(1)

If follows that on any box that "never" does a suspend operation, you get a random key for your cryptoswap at every boot, and that key is never saved anywhere. (So a feature, but at fairly high cost... 8-)

During a system suspend, the "last thing" that happens is that the applet block is saved to the swap system in a known place.

It is instantly and admittedly obvious that a person could take apart the computer and harvest this block and all the other regions of memory [if they are static] and create a complete key, and then use this key. So the "default" level of security will not keep the NSA or a good pre-made hardware assault out of this computer's saved/suspended swap space image. It will however thwart any attacker who doesn't have the tools to harvest all the hardware information and then compose a new computer using the old one.

You have to sort of run the model in your head...

CPU ID fetch: static means.

Clock Chip etched serial number: static location.

Bios Data: static location with identical bios setup password and maybe boot password.

Invocation Command Line: so no funky trick arguments.

Same root device: (mistake proofing, not so much anti-theft. 8-)

["heck maybe" the Bios Image Checksum or whatever to prevent hanky panky?]

Unique Entropy: from the meta-block.

On the other hand, for anything short of a really smart guy with a soldering iron, we now "know", as part of our restore, that we are using our CPU, our clock chip, our bios configuration (so, if set, our setup and our boot passwords) and our root file system and boot arguments (so no lilo-time override or installer disk). That's a heck of a lot of the chain "verified" for us as part of the "do we restore or reboot?" task.

OK, so, you know, "pretty good", keeps out most of the script kiddies and anybody who isn't willing to physically ruin the box... Yea, if you knew the version of linux that the box used and some of the memory map bits and you restored the bios image that you already saved right after boot, you might be able to make a custom kernel and get the box to restore.

The important thing is that the system is "pretty good" and is extensible because the text part of the template can be replaced, or selected from one of several compiled-in options, by each admin at non-restore boot time. (You *would* probably have to reboot to change templates, as the system is "on the wrong side of the fence" with respect to loaded modules and remapped memory when the meta-boot-block text is invoked.)

So with "just" a configuration change, the user may "switch" to the template that demands a password from the keyboard (and then reboot). Now you have moved up "one notch" in the world, but you haven't had to change your kernel out or anything.

Linux-Kernel: RE: mlock(1)

If you are adding safety instead of security maybe you compose a block that also checksums the sector of the disk that "just happens" to contain your /etc/fstab.

Maybe you want to write your own boot-block-template that reads a pluggable dongle that will act as a key. Again, no change to core kernel semantics, just a different 4k template, but now that template needs the hardware dongle you carry in your pocket to make the swap key. But it doesn't need it all the time, just when you boot, so you can put it in, boot, then take it out. It's not even a sophisticated device; you don't have to run a smart card algo, just read some region of a prom or something. You *could* run a smart-card algo though, if you had the room.

And like a chained boot loader, the text could load in a bigger program "hidden" on your disk to run an arbitrarily large task. That would be very advanced 8-), but now you have room to make a specialty smart-card or biometric check.

The fact of the matter is that this is "broken glass" security and detection. That is, if you break the glass by failing to have all the "sensitive" elements aligned for even one boot attempt, the swap images are invalidated and lost forever. (Yes, as admitted, it is imperfect, but its imperfections are quite stable and it can fit nicely into a "security strategy".) (Yes, you could save the swap image before you try, and then restore it if it didn't work and try again, but that restore ups the brute-force single iteration cost fantastically.)

Finally, it is notable that a thief could keep suspending and resuming the laptop for ever. If they always did an orderly suspend, they would always get an orderly resume. That's the point. That's also a good reason *not* to have your power button trigger a suspend, use an app... And if you don't have a locking screen saver, then you didn't do your part of the "securing my system" task.

Why the chain?

So if you want to "resume" your swap regions contain the secret key needed to use your `_encrypted_` file system. If you have "broken glass" security on your swap encrypted swap space, then you can easily resume and still be quite safe. If the thief "breaks the glass" by doing a bad suspend or a bogus resume, all the doors come down automatically, which is about as much as you can expect for "safe resume" protection.

And as mentioned before, if you never suspend, the whole thing hides its default-configured head, and you never knew it was there, and the key for your swap is never saved; so after any kind of outage or shutdown other than a resume, your sensitive data was scrambled by a key that will never see the light of day again (as long as your entropy wasn't "return 0;" 8-).

So someone who breaks into your office and steals the company server, better be ready to haul it off without unplugging it from your UPS.

Some of the features are very like "trusted computing", in that we are keeping track of key box-specific configuration and hardware information in the name of "known state" but we *want* known state for a safe resume anyway. Still the default only needs to go so far to be effective enough to be very useful. The advanced person can

RE: mlock(1)

Linux-Kernel: RE: mlock(1)

easily add passwords and dongles and who knows what other kind of box or platform specific nonsense, all without having to alter the kernel intrinsics or semantics.

Rob White,
Casabyte, Inc.

-----Original Message-----

From: Jon Masters [mailto:jonmasters@gmail.com]

Sent: Tuesday, September 28, 2004 6:16 PM

To: Robert White

Cc: Andrea Arcangeli; Nigel Cunningham; Alan Cox; Chris Wright; Jeff Garzik; Linux Kernel Mailing List; Andrew Morton

Subject: Re: mlock(1)

On Tue, 28 Sep 2004 16:26:16 -0700, Robert White <rwhite@casabyte.com> wrote:

> *If you are concerned about the stolen laptop scenario you would use the (theoretical)*

> *boot block that required a boot/restore password*

[This is not a flame honestly. Just some points about your idea.]

I don't see in your argument how this is meant to be cryptographically secure. Nor do I see from any of the original mail an idea which does anything more than offer a fake promise of security to those who are willing to assume only dumb criminals steal their laptop. This is worse than no security at all and renders the idea of encrypting swap completely useless.

> *or read the password out of your bios or something.*

This assumes some kind of trusted BIOS running on the machine which can obtain this password in some secure fashion – but we don't have this on most laptops. You pointed out the idea of DRM and noted that this is in fact quite horrible and I agree :-).

> *No zero-password/pass-device restore has any right to be expected*

> *to be any more secure than walking away from a running console.*

I fail to see anything wrong with the asking password approach – if I use a Microsoft box of evil(TM) then it'll ask me for a password on boot and on restore, most folks are happy to type their password in to any box which asks for one (whether or not the actual password dialog box) so they'll be happy to do so here. Just like having xscreensaver or xlock come on when you resume your laptop.

> *As stated, the idea is pretty basic, but if you have a computer you are worried might*

> *be stolen and compromised at this level, you presumably have set your bios passwords*

> *and such. If the "non-time" section of the bios config ram is one of the*

RE: mlock(1)

composition

- > *key elements, the act of clearing the bios to clear the boot password would*
- > *invalidate the data that the key generation block uses to recreate the key.*

Yes great. But:

- 1). I open the laptop up (I'm allowed to do that if I've already nicked it :P).
- 2). I take a copy of the BIOS.
- 3). I replace the BIOS with a hardware configuration (however done – perhaps hot swapping chips, perhaps some simple logic device helps me) in which the original BIOS is available once booting begins.
- 4). That part of the security model was just destroyed.

You can't trust anything you read from hardware. Ever. Perhaps you might choose to trust hardware containing burried memory under layers of metal (thus introducing an expense element to obtaining it with a tunneling electron microscope) which required some public key/other mechanism for communication that didn't just allow raw access to it.

- > *A "normal" investigator using a normal level of attack would be thwarted.*

That's not the point. Once a method exists for obtaining information from stolen laptops, that ends up not far from a simple google search near you.

- > *It's like your house keys, your house is secure unless someone steals your keys...*

I think the difference here is that if your house is broken in to, you presumably know what might have been stolen, but if your laptop is stolen it might be harder to guess exactly what can be retrieved from it afterwards. Personally I'm not currently paranoid enough to use encryption on the laptop but might do so when I have some free time to get around to it.

When my laptop went back to Apple I didn't do anything more than a simple delete on the files within it (so they could have grabbed bits of data from it or trogated it – but I do have copies so I can create checksums of the files which were on it), but I'm just not that worried that they secretly have a super villan on their repair team who'd try something on ;-).

The point is however – if I do decide to use encryption to protect some of my personal data (and I don't care about credit cards – if my laptop is nicked then I can cancel the card) then I want to be sure that the mechanism is actually sound. There's no point to any system which simply claims to be secure (like certain USB devices) but isn't.

- > *But for that "casually" stolen laptop or computer, if they boot from a CD (having*
- > *this mecanisim) or use alternate boot parameters, or restore once and the hit the*
- > *power switch because they couldn't trigger the suspend, the swap image is*
- scrambled.*

RE: mlock(1)

Linux-Kernel: RE: mlock(1)

- > *If they use the system "vergin", without changing a thing, then they will get your*
- > *normal boot, which is no more or less secure than you have set it up to be via the*
- > *front door services (login. FTP, etc).*

I see your point but don't think it's good to assume anything about an attacker. While it's unlikely that someone is going to have the technical expertise to thwart your model, I'd only want to use something which relied on more than attacker knowledge. The reality is however that someone stealing my laptop will see an xscreensaver, reboot, possibly see Linux booting and then get really confused because a). it's a Mac so it's not one of those PC things that you can stick a bootleg Windows XP on. b). It's a Mac running something weird so it'll need a copy of that Mac software thing to make it work. They might be smart enough to visit a bookshop and read the page in the Hayes Apple manual on removing the hard disk before flogging it on ebay "for parts".

- > *And being extensible, you could have your very-own secret, home-made*
- > *key-generation-block template that you wrote yourself that nobody else in the world*
- > *knows about and makes restore (swap decryption) predicate on anything your heart*
- can*
- > *envision and you can provide in a timely manner during startup... Like the*
- > *super-secret RFID tag in your favorite pair of wall-mart sneakers.*

...or my tux boxers. They'd have to be very determined to go there for my passphrase.

- > *Also, the attacker gets only one chance not to screw up because the non-decrypt*
- > *startup sequence destroys the original key generation block and reinitializes the*
- > *swap headers. (Yes, he could save the swap image and try again but that is a long*
- > *brute-force cycle. 8-)*

First thing I'd do when messing with a laptop would be to take a copy of the disk.

- > *but I think you will find it a lot more secure than you might think after just one*
- read.*

I'll read it again and see what I missed then – you're probably right there.

Jon.

–

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@vger.kernel.org
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>