



MinorFS & AppArmor

Taming mutable state for filesystem access.



MinorFS & AppArmor

- Shared mutable state
- Emakers
- AppArmor
- FUSE
- MinorFs
- Bringing things together.

A powerful program

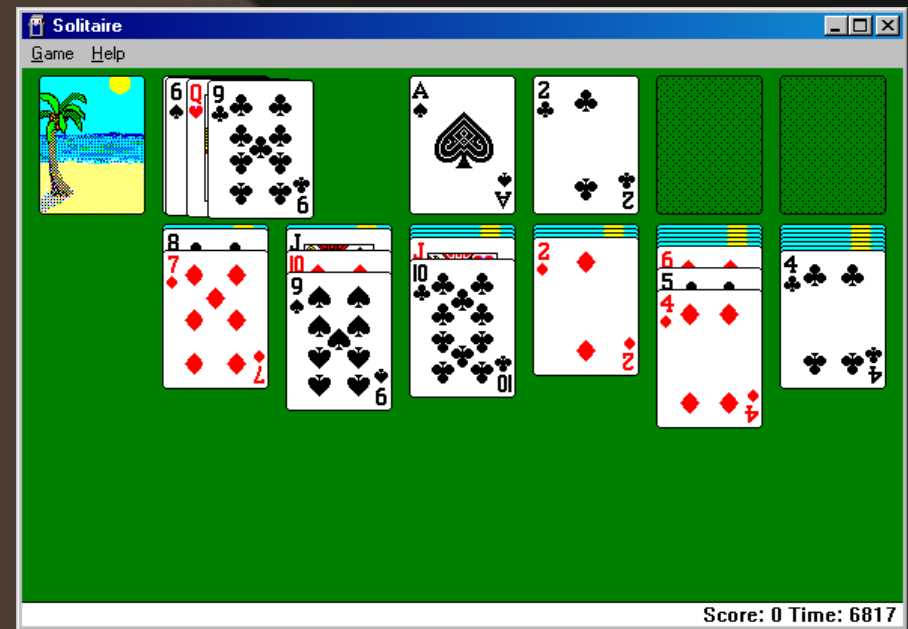
A powerful program

- Read your confidential files.
- Mail them to the competition.
- Delete or compromise your files.
- Initiate a networking tunnel allowing your competitor into your corporate network.



A powerful program

- Read your confidential files.
- Mail them to the competition.
- Delete or compromise your files.
- Initiate a networking tunnel allowing your competitor into your corporate network.



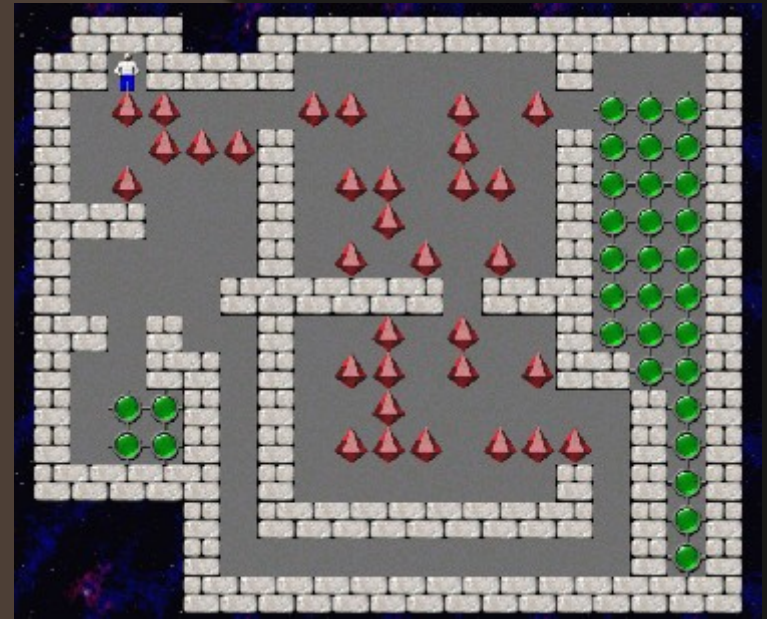
A powerful program

- Read your confidential files.
- Mail them to the competition.
- Delete or compromise your files.
- Initiate a networking tunnel allowing your competitor into your corporate network.
- Not (just) MS !!



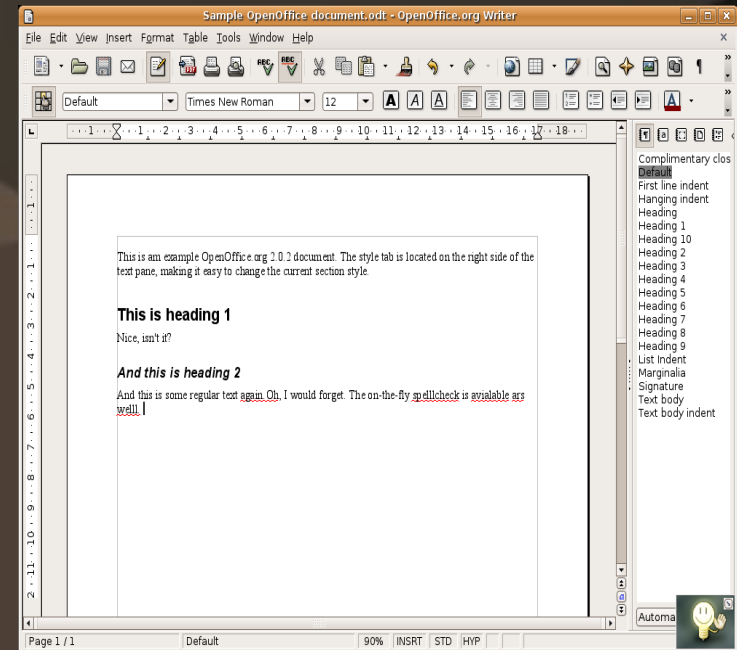
Who/what do we trust?

- Do we trust Sokoban with all this power?
- How many people would have been able to trojanize Sokoban?
- How about firefox?
- Exploitable bugs?
- More programs?
- Lines of code ?



Two sides of the coin

- Sokoban has the power to read your secrets.
- Openoffice Writer does NOT have the power to protect your secrets.
- Your programs could use some privacy.



mutable state



vs



Global mutable state & filesystems

- \$TEMP
- \$HOME
- Other (obscure?) places

Problems with global mutable state

- Can potentially be modified from anywhere
- Any subsystem may rely on it.
- Potential for creating mutual dependencies
- Makes composite systems hard to analyze or review.
- Makes composite systems hard to test.
- High potential for violating the principle of least authority.

*Shared mutable state,
can we fix it?*



*Shared mutable state,
can we fix it?*



Didn't they fix this already ??

Shared mutable state & programming

- Global variables
- Static member variables (OO)
- Singletons (OO)

Solutions from OO programming

- Basic OO: Encapsulation/Data hiding.
 - Private member variables.
 - Static member variables (the lesser evil)
 - Singletons (the hidden lesser evil)
- Dependency injection.
 - Avoid the lesser evils for testability purposes.
- Object capabilities.
 - Avoid the lesser evils for least authority purposes.



POLA example: emakers

- E-Language
 - Memory safe programming language
 - Asynchronous language running on JVM.
 - Code in the *main* program code runs with the program's full authority.
 - Subsystems implemented as emaker run with dynamic least authority.
 - Fewer *powerful code* lines.



POLA example: emakers

- Subsystems implemented as **emaker** only have authority:
 - That was handed to it at construction time.
 - To objects created by the subsystem itself.
 - That is passed to it explicitly.

Shared mutable state, can we fix it?



Lets use these principles to fix
access control for the File system.

Basic principles

- Avoid implicitly shared mutable state.
- Instead use decomposition, attenuation and EXPLICIT delegation.
- Use the OO paradigm at all abstraction levels.
- Minimize knowledge, minimize authority.
- Minimize the size of the trusted code base.

Avoid implicitly shared mutable state

- 3 ways to gain access to resources:
 - Parenthood
 - Initial conditions.
 - Passed (delegated) explicitly.

*Shared mutable state,
can we fix it?*



First the foundation.



Initial conditions: AppArmor

- Path based access control for Suse & Ubuntu.
- Allows profiles to be created for applications.
 - Authority handed to process at construction time.
 - A strict profile makes a program into a coarse-grained static equivalent of the emulator.
 - For POLA-based file system access we need to make this equivalence dynamic.
- AppArmor takes away ambient authority to data, now let's replace that with private data and explicit designation.

FUSE

- File systems as user space
- Library and kernel module
- Allows a user space program to expose a file system abstraction to the system.

FUSE

*Shared mutable state,
can we fix it?*



Now for the carrying walls

MinorFs: parenthood and delegation

- MinorCapFS:
 - Unguessable tokens for directory access
 - Toplevel mountpoint appears empty.
 - /mnt/minorfs/cap is an empty directory.
 - /mnt/minorfs/cap/1a12bd48fa710276432a986865876fcd4587873d discloses a directory tree.
- MinorViewFS
 - Delegation of private directories to processes.
 - /mnt/minorfs/priv/tmp delegates private storage to non persistent processes.
 - /mnt/minorfs/priv/home delegates private storage to pseudo persistent processes.



MinorFS and \$TMP usage

- `/mnt/minorfs/priv/tmp` is a symbolic link.
- For each process id the link points at a different private storage dir.
- When a process dies, MinorViewFS deletes its private storage dir.
- Processes can safely store private data in temporary files.
- Good alternative for `/tmp/` usage without the problems of global or static mutable state.



Pseudo persistent processes

- Non persistent processes don't survive reboots.
- Often their functionality **does or should**.
- Part of (or all) the program state can be serialized to disk.
- Persistent **global or static mutable storage** is used to allow restoring state after reboot.
- Using the program identity and a slot system, we can define pseudo persistent processes.



Pseudo persistent processes

(MinorFS style)

- Pseudo Persistent Process Unique ID's (PPPID):
 - The path of the executable.
 - The invocation chain with all parent processes up to init.
 - The loaded libraries in the invocation chain.
 - The user id the process runs under.
 - Often the command line / environment variables
 - A slot, identifying the n-th process invoked in this way.
- If after a reboot a process claims a PPPID, we shall address it as the same pseudo persistent process as the one holding the PPPID before reboot.



MinorFS and \$HOME

- /mnt/minorfs/priv/home is a symbolic link.
- For each PPPID the link points at a different private storage dir.
- When a process dies, MinorViewFS retains the storage for the next incarnation.
- Processes can safely store persistent private data, including its own serialization.
- Good alternative for \$HOME usage without the problems of global or static mutable state.

*Shared mutable state,
can we fix it?*



Lets go multi-storey



Persistent processes

- AppArmor takes away excess authority.
- MinorFS delegates private storage to pseudo persistent processes.
- The E-Language allows a so called VAT to be made persistent by coupling to on-disk storage.
- Combining these 3 allows for the creation of persistent processes without using global or static mutable state.
- Combining these 3 uses the same abstractions at multiple granularities.

Global versus private, example:

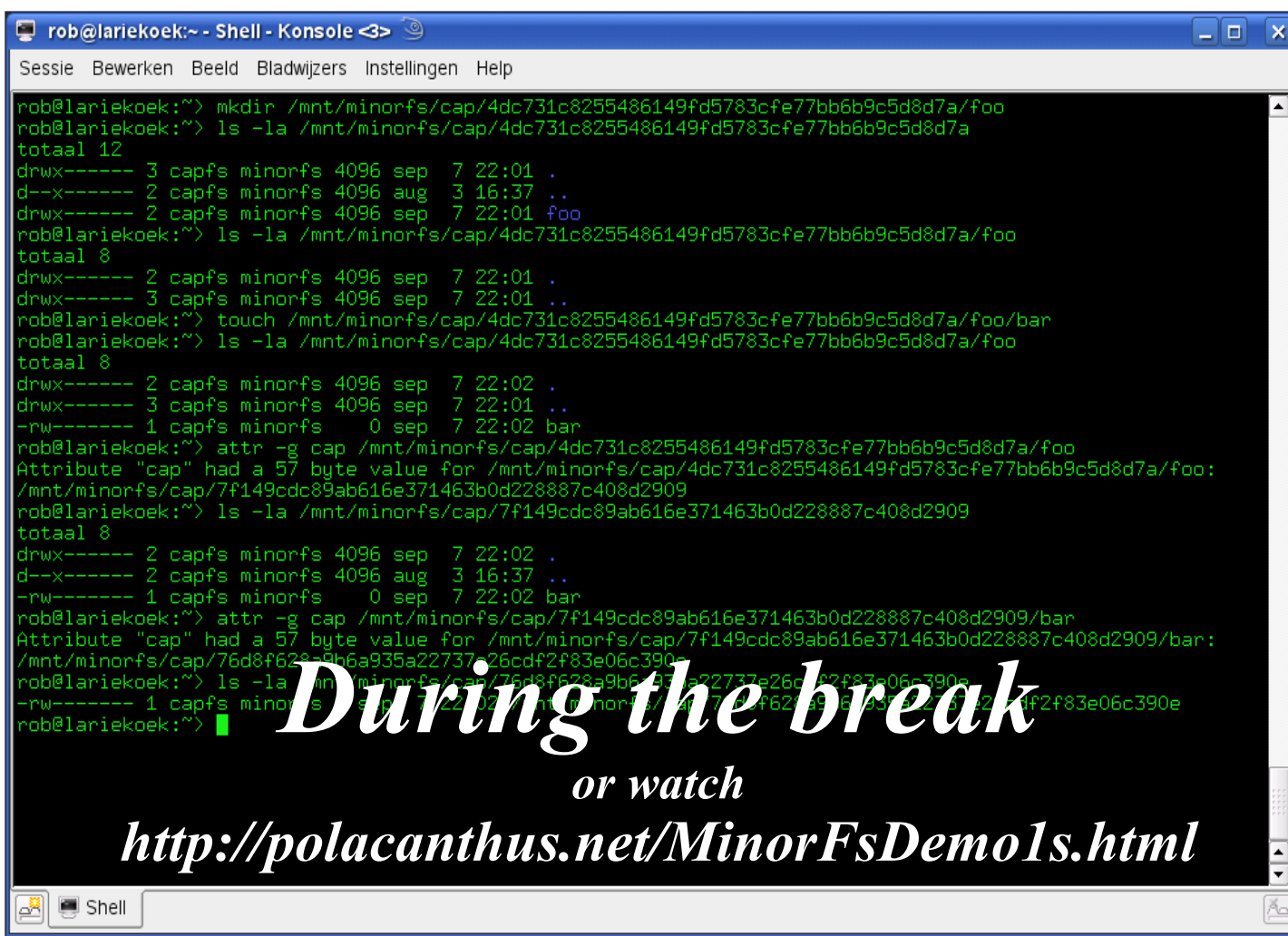
- Shared \$HOME
- Shared \$TMP
- 20 'trusted' programs
- Use memory unsafe languages for all programs
- 50.000 of lines of 'trusted' code per application.
- 1.000.0000 lines of trusted code.
- Private \$HOME
- Private \$TMP
- 2 'trusted' programs.
- Use capability secure languages for trusted programs.
- 1000 lines of 'trusted' code per trusted application.
- 2.000 lines of trusted code.

DEMO

```
rob@lariekoek:~ - Shell - Konsole <>
Sessie Bewerken Beeld Bladwijzers Instellingen Help

rob@lariekoek:~> mkdir /mnt/minorfs/cap/4dc731c8255486149fd5783cfe77bb6b9c5d8d7a/foo
rob@lariekoek:~> ls -la /mnt/minorfs/cap/4dc731c8255486149fd5783cfe77bb6b9c5d8d7a
totaal 12
drwx----- 3 capfs minorfs 4096 sep  7 22:01 .
d--x----- 2 capfs minorfs 4096 aug  3 16:37 ..
drwx----- 2 capfs minorfs 4096 sep  7 22:01 foo
rob@lariekoek:~> ls -la /mnt/minorfs/cap/4dc731c8255486149fd5783cfe77bb6b9c5d8d7a/foo
totaal 8
drwx----- 2 capfs minorfs 4096 sep  7 22:01 .
drwx----- 3 capfs minorfs 4096 sep  7 22:01 ..
rob@lariekoek:~> touch /mnt/minorfs/cap/4dc731c8255486149fd5783cfe77bb6b9c5d8d7a/foo/bar
rob@lariekoek:~> ls -la /mnt/minorfs/cap/4dc731c8255486149fd5783cfe77bb6b9c5d8d7a/foo
totaal 8
drwx----- 2 capfs minorfs 4096 sep  7 22:02 .
drwx----- 3 capfs minorfs 4096 sep  7 22:01 ..
-rw----- 1 capfs minorfs    0 sep  7 22:02 bar
rob@lariekoek:~> attr -g cap /mnt/minorfs/cap/4dc731c8255486149fd5783cfe77bb6b9c5d8d7a/foo
Attribute "cap" had a 57 byte value for /mnt/minorfs/cap/4dc731c8255486149fd5783cfe77bb6b9c5d8d7a/foo:
/mnt/minorfs/cap/7f149cdc89ab616e371463b0d228887c408d2909
rob@lariekoek:~> ls -la /mnt/minorfs/cap/7f149cdc89ab616e371463b0d228887c408d2909
totaal 8
drwx----- 2 capfs minorfs 4096 sep  7 22:02 .
d--x----- 2 capfs minorfs 4096 aug  3 16:37 ..
-rw----- 1 capfs minorfs    0 sep  7 22:02 bar
rob@lariekoek:~> attr -g cap /mnt/minorfs/cap/7f149cdc89ab616e371463b0d228887c408d2909/bar
Attribute "cap" had a 57 byte value for /mnt/minorfs/cap/7f149cdc89ab616e371463b0d228887c408d2909/bar:
/mnt/minorfs/cap/76d8f628a9b6a935a22737e26cdf2f83e06c390e
rob@lariekoek:~> ls -la /mnt/minorfs/cap/76d8f628a9b6a935a22737e26cdf2f83e06c390e
-rw----- 1 capfs minorfs 0 sep  7 22:02 /mnt/minorfs/cap/76d8f628a9b6a935a22737e26cdf2f83e06c390e
rob@lariekoek:~> █
```

DEMO



```
rob@lariekoek:~ - Shell - Konsole <>
Sessie Bewerken Beeld Bladwijzers Instellingen Help

rob@lariekoek:~> mkdir /mnt/minorfs/cap/4dc731c8255486149fd5783cfe77bb6b9c5d8d7a/foo
rob@lariekoek:~> ls -la /mnt/minorfs/cap/4dc731c8255486149fd5783cfe77bb6b9c5d8d7a
totaal 12
drwx----- 3 capfs minorfs 4096 sep  7 22:01 .
d--x----- 2 capfs minorfs 4096 aug  3 16:37 ..
drwx----- 2 capfs minorfs 4096 sep  7 22:01 foo
rob@lariekoek:~> ls -la /mnt/minorfs/cap/4dc731c8255486149fd5783cfe77bb6b9c5d8d7a/foo
totaal 8
drwx----- 2 capfs minorfs 4096 sep  7 22:01 .
drwx----- 3 capfs minorfs 4096 sep  7 22:01 ..
rob@lariekoek:~> touch /mnt/minorfs/cap/4dc731c8255486149fd5783cfe77bb6b9c5d8d7a/foo/bar
rob@lariekoek:~> ls -la /mnt/minorfs/cap/4dc731c8255486149fd5783cfe77bb6b9c5d8d7a/foo
totaal 8
drwx----- 2 capfs minorfs 4096 sep  7 22:02 .
drwx----- 3 capfs minorfs 4096 sep  7 22:01 ..
-rw----- 1 capfs minorfs    0 sep  7 22:02 bar
rob@lariekoek:~> attr -g cap /mnt/minorfs/cap/4dc731c8255486149fd5783cfe77bb6b9c5d8d7a/foo
Attribute "cap" had a 57 byte value for /mnt/minorfs/cap/4dc731c8255486149fd5783cfe77bb6b9c5d8d7a/foo:
/mnt/minorfs/cap/7f149cdc89ab616e371463b0d228887c408d2909
rob@lariekoek:~> ls -la /mnt/minorfs/cap/7f149cdc89ab616e371463b0d228887c408d2909
totaal 8
drwx----- 2 capfs minorfs 4096 sep  7 22:02 .
d--x----- 2 capfs minorfs 4096 aug  3 16:37 ..
-rw----- 1 capfs minorfs    0 sep  7 22:02 bar
rob@lariekoek:~> attr -g cap /mnt/minorfs/cap/7f149cdc89ab616e371463b0d228887c408d2909/bar
Attribute "cap" had a 57 byte value for /mnt/minorfs/cap/7f149cdc89ab616e371463b0d228887c408d2909/bar:
/mnt/minorfs/cap/76d8f628a9b6a935a22737c26cdf2f83e06c390e
rob@lariekoek:~> ls -la /mnt/minorfs/cap/76d8f628a9b6a935a22737c26cdf2f83e06c390e
-rw----- 1 capfs minorfs    0 sep  7 22:02 /mnt/minorfs/cap/76d8f628a9b6a935a22737c26cdf2f83e06c390e
rob@lariekoek:~>
```

During the break

or watch

<http://polacanthus.net/MinorFsDemo1s.html>

Resources

- The E-Language
 - <http://www.erights.org/>
- AppArmor
 - <http://en.opensuse.org/AppArmor>
- FUSE
 - <http://fuse.sourceforge.net/>
- MinorFS
 - <http://minorfs.polacanthus.net/>
 - <http://www.linuxjournal.com/article/10199>

