Emacs Configuration

deepwaterooo
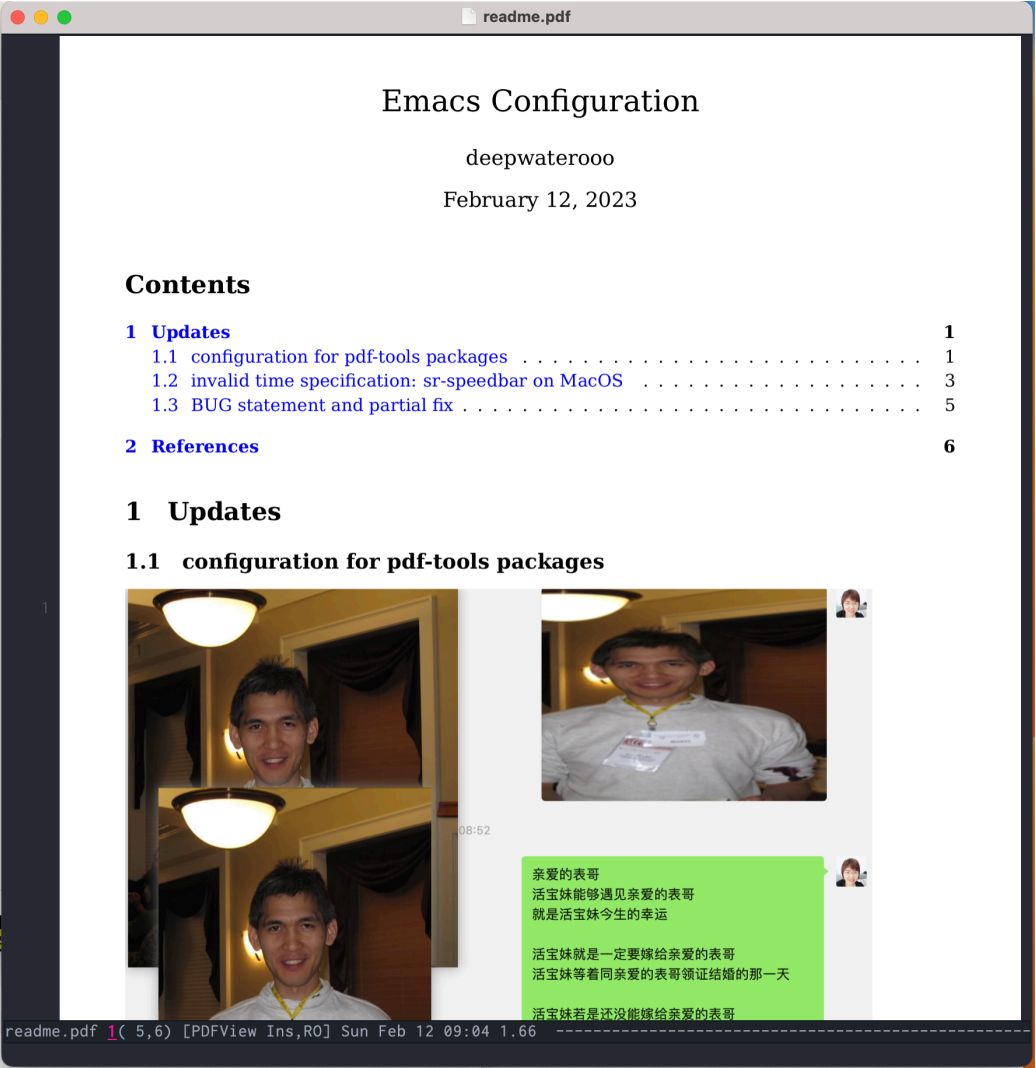
February 12, 2023

Contents

1 Updates

1.1 configuration for pdf-tools packages

- pdf viewer noter <==> Skim bi-directional linking configuration on the way, most emacs work is done. Need to learn how to use them though.

hhj@Heyans-MacBook-Pro:~/.emacs.d/elpa/pdf-tools

 ~/.emacs.d/e/pdfBK-tools on master ~/.emacs.d/elpa/pdfBK-tools/server/autobuild -i ~/.emacs.d/elpa/pdfBK-tools/

```
---------------------------
   Installing packages
---------------------------
Skipping package installation (already installed)

---------------------------
 Configuring and compiling
---------------------------
./configure -q  && make clean && make -s

Is case-sensitive searching enabled ?    yes
Is modifying text annotations enabled ?   yes
Is modifying markup annotations enabled ? yes

test -z "epdfinfo" || rm -f epdfinfo
test -z "libsynctex.a" || rm -f libsynctex.a
rm -f *.o
epdfinfo.c:354:19: warning: 'tempnam' is deprecated: This function is provided for compatibility reasons only.  Due to security concerns inher
ent in the design of tempnam(3), it is highly recommended that you use mkstemp(3) instead. [-Wdeprecated-declarations]
     filename = tempnam(NULL, "epdfinfo");
                ^
/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX.sdk/usr/include/stdio.h:312:1: note: 'tempnam' has
been explicitly marked deprecated here
__deprecated_msg("This function is provided for compatibility reasons only.  Due to security concerns inherent in the design of tempnam(3), it
 is highly recommended that you use mkstemp(3) instead.")
^
/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDKs/MacOSX.sdk/usr/include/sys/cdefs.h:215:48: note: expanded
from macro '__deprecated_msg'
        #define __deprecated_msg(_msg) __attribute__((__deprecated__(_msg)))
                                                      ^
epdfinfo.c:1548:5: warning: 'poppler_page_get_selection_region' is deprecated: Use poppler_page_get_selected_region instead [-Wdeprecated-decl
arations]
    poppler_page_get_selection_region (page, 1.0, selection_style, region);
    ^
/usr/local/Cellar/poppler/23.02.0/include/poppler/glib/poppler-page.h:75:134: note: 'poppler_page_get_selection_region' has been explicitly ma
rked deprecated here
GList *poppler_page_get_selection_region(PopplerPage *page, gdouble scale, PopplerSelectionStyle style, PopplerRectangle *selection) G_GNUC_DE
PRECATED_FOR(poppler_page_get_selected_region);
                                                                                                                                     ^
/usr/local/Cellar/glib/2.74.5/include/glib-2.0/glib/gmacros.h:694:18: note: expanded from macro 'G_GNUC_DEPRECATED_FOR'
    __attribute__((deprecated("Use " #f " instead")))     \
                 ^
2 warnings generated.

===========================
   Build succeeded. :O)
===========================
```
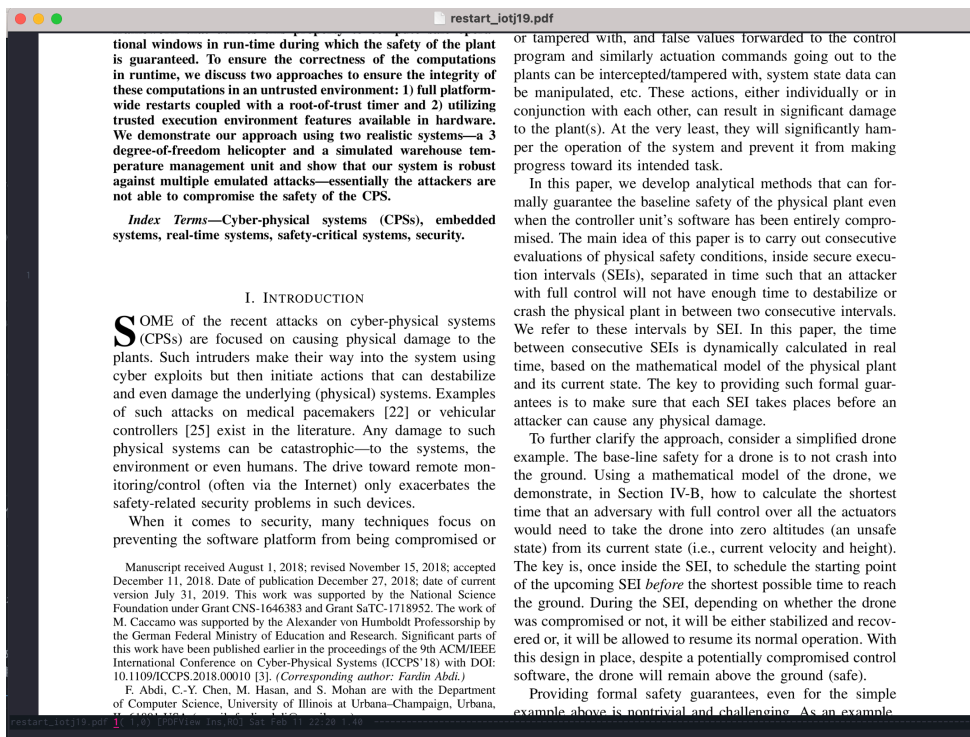
- It has to be configured for M1. But I am still not getting any .tar file yet.

restart_iotj19.pdf

tional windows in run-time during which the safety of the plant is guaranteed. To ensure the correctness of the computations in runtime, we discuss two approaches to ensure the integrity of these computations in an untrusted environment: 1) full platform-wide restarts coupled with a root-of-trust timer and 2) utilizing trusted execution environment features available in hardware. We demonstrate our approach using two realistic systems—a 3 degree-of-freedom helicopter and a simulated warehouse temperature management unit and show that our system is robust against multiple emulated attacks—essentially the attackers are not able to compromise the safety of the CPS.

Index Terms—Cyber-physical systems (CPSs), embedded systems, real-time systems, safety-critical systems, security.

I. INTRODUCTION

SOME of the recent attacks on cyber-physical systems (CPSs) are focused on causing physical damage to the plants. Such intruders make their way into the system using cyber exploits but then initiate actions that can destabilize and even damage the underlying (physical) systems. Examples of such attacks on medical pacemakers [22] or vehicular controllers [25] exist in the literature. Any damage to such physical systems can be catastrophic—to the systems, the environment or even humans. The drive toward remote monitoring/control (often via the Internet) only exacerbates the safety-related security problems in such devices.

When it comes to security, many techniques focus on preventing the software platform from being compromised or

F. Abdi, C.-Y. Chen, M. Hasan, and S. Mohan are with the Department of Computer Science, University of Illinois at Urbana–Champaign, Urbana,

or tampered with, and false values forwarded to the control program and similarly actuation commands going out to the plants can be intercepted/tampered with, system state data can be manipulated, etc. These actions, either individually or in conjunction with each other, can result in significant damage to the plant(s). At the very least, they will significantly hamper the operation of the system and prevent it from making progress toward its intended task.

In this paper, we develop analytical methods that can formally guarantee the baseline safety of the physical plant even when the controller unit's software has been entirely compromised. The main idea of this paper is to carry out consecutive evaluations of physical safety conditions, inside secure execution intervals (SEIs), separated in time such that an attacker with full control will not have enough time to destabilize or crash the physical plant in between two consecutive intervals. We refer to these intervals by SEI. In this paper, the time between consecutive SEIs is dynamically calculated in real time, based on the mathematical model of the physical plant and its current state. The key to providing such formal guarantees is to make sure that each SEI takes places before an attacker can cause any physical damage.

To further clarify the approach, consider a simplified drone example. The base-line safety for a drone is to not crash into the ground. Using a mathematical model of the drone, we demonstrate, in Section IV-B, how to calculate the shortest time that an adversary with full control over all the actuators would need to take the drone into zero altitudes (an unsafe state) from its current state (i.e., current velocity and height). The key is, once inside the SEI, to schedule the starting point of the upcoming SEI *before* the shortest possible time to reach the ground. During the SEI, depending on whether the drone was compromised or not, it will be either stabilized and recovered or, it will be allowed to resume its normal operation. With this design in place, despite a potentially compromised control software, the drone will remain above the ground (safe).

Providing formal safety guarantees, even for the simple example above is nontrivial and challenging. As an example

- could customer F5 toggle sr-speedbar, and make sis-mode work. But I do NOT really need

sis-mode, only needed macism command line. to help [LOVE MY DEAR COUSIN!!!]

- finally sync up with csharp-mode with tree-sitter, and fixed added the other's that gio-mode etc.

- Permanently diabled speedbar-edit-file's set-timer function call from Resource files. Do NOT want to see such a bug, don't know how to fix, but disable it and walk around.

- Now have a relatively barable and stable colorful emacs code editor now, at least for csharp-mode. Relatively satisfied now. Could sit it aside for a while now to focus on projects.

1.2   invalid time specification: sr-speedbar on MacOS

- I don't like this bug, and I belive I do NOT really using any timer for auto-refresh in my speedbar. So I ended up by disabling the (speedbar-edit-file() func, which is frequently bug trigered) setting timer part from /Applications/Emacs.app/Contents/Resources/lisp/speedbar. and recompile the file. The bug was gone. And I could deal with csharp-mode's fontify bug.

- newer debugging infos, concernibg about sr-speedbar.el file. Have NOT been able to trace down for today.

- 好像是 macOS 系统常常存在的 bug, 两年了关了又开，开了再关......

- https://github.com/remacs/remacs/issues/845

- get cmake work later, not urgent though.Complete

- babel org-mode so I don't have to copy from specific babeled source org-mode files in order for chinese characters to work.

- Installed my emacs of version 28.1. But there is a bug of any verison emacs > 27.1, and I am NOT able to find a installable emacs 27.1 version any more.

- 不同电脑架构上可能因为架构的不同，可以可能可以有某些优化. 又照一个单做了一遍，似乎没有出错. https://goykhman.ca/gene/blog/2022/2022-04-10-emacs-28.1-on-m1.html

- 但是我没有没能加入那个补丁包. 暂时没能想好怎么加入那个补丁包.（**todo：改天可以尝试再把这个补丁包加进去**）

- 因为构建是在原有现有的 mac 28.1.1 版本上构建的. 所以改动什么，或是不曾改变，又或者改不了不影响明显功能都是无从知晓的，但是它最后的两个步骤的验证都是成功的，应该还是构建成功了吧？

```
                /usr/bin/install -c -m 644 etc/${docfile} "/usr/local/share/emacs/28.1/etc/${docfile}"; \
                for installuser in ${LOGNAME} ${USERNAME} ${USER} `(id -u) 2> /dev/null`; do [ -n "${installuser}" ] && break ; done; installgrou
p=`(id -g) 2>/dev/null` && [ -n "$installgroup" ] && installuser=$installuser:$installgroup; \
                chown ${installuser} "/usr/local/share/emacs/28.1/etc/${docfile}" || true ; \
        else true; fi
Copying etc/DOC to /usr/local/share/emacs/28.1/etc ...
umask 022; ./build-aux/install-sh -c -d "/usr/local/bin"
/Applications/Xcode.app/Contents/Developer/usr/bin/make -C lib-src install

Installing utilities run internally by Emacs.
umask 022 && ../../build-aux/install-sh -c -d "/usr/local/libexec/emacs/28.1/aarch64-apple-darwin22.3.0"
exp_archlibdir=`cd "/usr/local/libexec/emacs/28.1/aarch64-apple-darwin22.3.0" && /bin/pwd` && \
        if [ "$exp_archlibdir" != "`/bin/pwd`" ]; then \
          for file in hexl  movemail ; do \
            /usr/bin/install -c  $file \
                "/usr/local/libexec/emacs/28.1/aarch64-apple-darwin22.3.0/$file" || exit; \
          done ; \
        fi
exp_archlibdir=`cd "/usr/local/libexec/emacs/28.1/aarch64-apple-darwin22.3.0" && /bin/pwd` && \
        if [ "$exp_archlibdir" != "`cd . && /bin/pwd`" ]; then \
          for file in rcs2log; do \
            /usr/bin/install -c ./$file \
                "/usr/local/libexec/emacs/28.1/aarch64-apple-darwin22.3.0/$file" || exit; \
          done ; \
        fi

Installing utilities for users to run.
umask 022 && ../../build-aux/install-sh -c -d "/usr/local/bin"
for file in etags ctags emacsclient  ebrowse ; do \
          /usr/bin/install -c  ${file} \
            "/usr/local/bin/"/`  \
             echo ${file} | sed -e 's/$//' -e 's,x,x,' \
          ` || exit; \
        done
if test "/Users/hhj/selfSoft/emacs-mac/mac" != "`cd mac && /bin/pwd`"; then \
          umask 022; ./build-aux/install-sh -c -d /Users/hhj/selfSoft/emacs-mac/mac/Emacs.app; \
          (cd mac/Emacs.app; (tar -chf - . | \
              (cd /Users/hhj/selfSoft/emacs-mac/mac/Emacs.app; umask 022; tar -xvf - \
                  && cat > /dev/null))) || exit 1; \
        fi
/usr/bin/install -c  src/emacs "/usr/local/bin/`echo emacs-28.1 | sed 's,x,x,'`"
/usr/bin/install -c -m 644 src/emacs.pdmp "/usr/local/libexec/emacs/28.1/aarch64-apple-darwin22.3.0"/emacs.pdmp
chmod 755 "/usr/local/bin/`echo emacs-28.1 | sed 's,x,x,'`"
rm -f "/usr/local/bin/`echo emacs | sed 's,x,x,'`"
cd "/usr/local/bin" && ln -s "`echo emacs-28.1 | sed 's,x,x,'`" "`echo emacs | sed 's,x,x,'`"
/Applications/Xcode.app/Contents/Developer/usr/bin/make -C lib-src maybe-blessmail
make[1]: Nothing to be done for `maybe-blessmail'.
 emacs-mac  on  #emacs-28.1-mac-9.0 73  em --version        ✓  took 1m 20s   at 02:16:21 PM 
GNU Emacs 28.1
Copyright (C) 2022 Free Software Foundation, Inc.
GNU Emacs comes with ABSOLUTELY NO WARRANTY.
You may redistribute copies of GNU Emacs
under the terms of the GNU General Public License.
For more information about these matters, see the file named COPYING.
```

- 今天又尝试安装 Xcode 之后再构建一遍，但是没有成功. 可能本身参考有些年代，另外自己还完全不通这个部分，所以暂时放一放. 改天有机会可以再回来研究一下，错在哪里，我如何才可能构建出自己的版本.

```
macfont.m:3482:15: warning: 'CTGetCoreTextVersion' is deprecated: first deprecated in macOS 11.0 - Use -[NSProcessInfo operatin
gSystemVersion] [-Wdeprecated-declarations]
         if (CTGetCoreTextVersion () >= kCTVersionNumber10_9)
             ^
/Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/System/Library/Frameworks/CoreText.framework/Headers/CoreText.h:61:10: note
: 'CTGetCoreTextVersion' has been explicitly marked deprecated here
uint32_t CTGetCoreTextVersion( void ) CT_DEPRECATED("Use -[NSProcessInfo operatingSystemVersion]", macos(10.5, 11.0), ios(3.2,
14.0), watchos(2.0, 7.0), tvos(9.0, 14.0));
         ^
1 warning generated.
  CC       terminfo.o
  CC       lastfile.o
/Library/Developer/CommandLineTools/usr/bin/make -C ../admin/charsets all
make[2]: Nothing to be done for `all'.
/Library/Developer/CommandLineTools/usr/bin/make -C ../admin/unidata charscript.el
make[2]: Nothing to be done for `charscript.el'.
  CCLD     temacs
../../build-aux/install-sh -c -d ../etc
/Library/Developer/CommandLineTools/usr/bin/make -C ../lisp update-subdirs
cp -f temacs bootstrap-emacs
rm -f bootstrap-emacs.pdmp
./temacs --batch  -l loadup --temacs=pbootstrap
make[1]: *** [bootstrap-emacs.pdmp] Killed: 9
make: *** [src] Error 2
 🍎  ⬛ ~/s/emacs  on git  emacs-27 !3 ?1  sudo make install              ✓  at 10:13:25 AM ⊘
Password:
/Library/Developer/CommandLineTools/usr/bin/make -C lib all
make[1]: Nothing to be done for `all'.
/Library/Developer/CommandLineTools/usr/bin/make -C lib-src all
make[1]: Nothing to be done for `all'.
/Library/Developer/CommandLineTools/usr/bin/make -C src VCSWITNESS='$(srcdir)/../.git/logs/HEAD' all
/Library/Developer/CommandLineTools/usr/bin/make -C ../admin/charsets all
make[2]: Nothing to be done for `all'.
/Library/Developer/CommandLineTools/usr/bin/make -C ../admin/unidata charscript.el
make[2]: Nothing to be done for `charscript.el'.
rm -f bootstrap-emacs.pdmp
./temacs --batch  -l loadup --temacs=pbootstrap
make[1]: *** [bootstrap-emacs.pdmp] Killed: 9
make: *** [src] Error 2
```

- 上面又成为一个需要改的东西：就是系统下如何从剪贴板自动生成写入文件 org-mode M-s

- 然后看见这里说可以自己构建一个，连 Xcode 也没有安装，就跑去构建了，当然不成功。这段时间太忙，XCode 要的空间太大了，暂时还不想。等改天有机会的时候倒是可以一试的

  - https://stuff-things.net/2020/12/28/building-emacs-27-dot-1-on-macos-big-sur/

- added key-bindings for opening from VSCode/Android Studio of current emacs buffer.

  - VSC Emacs can locate to each othr to correct row and col.

  - Android Studio could open current emacs buffer. but not to the row nor col.

- I liked recently configued Visual studio 2019 one-dark-pro theme, want to configue it for emacs, but ended up with any permission denied, renaming emacs initiating bug. reverted back for daily use, and may look into that bug for later reference.

- will reconfigure one-dark-pro theme later.

- fixed legency java-mode highlighing issue which I did not fix for years. Has been able to treat java-mode as java-mode Instead of using it as csharp-mode. Can not separate java-mode snippets from csharp-mode's.

1.3   BUG statement and partial fix

- in java-mode, the code style I expected is as followed:

```java
class node {
    int v ;
    public node() {
        if (a > 0) // I don't want { } blocks when I have only one line statement inside blocks
 // a = 17;    // before fix:
            a = 17;   // now it can auto-indent
        b = 20;
        while (true) // same auto indents here
            j++;
```

```
    }
}
class dklfjdj {|} // <<==== current un-auto-expanded version, bug right now for java-mode
class dklfjdj {
    | // <<==== expected feature: once I typed '{', '}' will be autopaired(it does), but also auto-expand and cursor moves
}
```

- if while if while one line statement autoindent without {} fixed today for java-mode, but for kotlin-mode, this bug consists, make coders/programmers nuts.

```
fun getStringLength(obj: Any): Int? {
    if (obj is String)
    return obj.length  // <<<<====== BUG: need to fix auto-indent here for if else while etc without {}

    if (obj is String) {
        return obj.length
    }
    // 在离开类型检测分支后，`obj` 仍然是 `Any` 类型
    return null
}
fun dslfkj { // kotlin-mode, unlike java-mode, this feature works charming
    val a = 1720
}
```

- The {|} can NOT auto-expand still bugs me a lot, I don't want to switch back to java-mode yet unless bug fixed and it auto-pands.

- java minor bug: Debugger entered–Lisp error: (void-function company-clear-completion-rules): this bug I will look into it recently, and expect it to be fixed so I could switch java-mode from csharp-mode as soon as possible.

- **csharp-mode** has been the one that works perfectly for these two features, {} auto expand, also if while one line statement autoindent without {}, so I used csharp-mode as java mode.

2   References

- <https://github.com/redguardtoo/emacs.d>