If you've been following me around the internets for a while you've likely heard me pontificate about the need to be aware of and reduce — when possible — your personal "cyber" attack surface. One of the ways you can do that is to install as few applications as possible onto your devices and make sure you have a decent handle on those you've kept around are doing or capable of doing.

On macOS, one application attribute you can look at is the set of "entitlements" apps have asked for and that you have actioned on (i.e. either granted or denied the entitlement request). If you have Developer Tools or Xcode installed you can use the `codesign` utility (it may be usable w/o the developer tools, but I never run without them so drop a note in the comments if you can confirm this) to see them:

```
$ codesign -d --entitlements :- /Applications/RStudio.app
Executable=/Applications/RStudio.app/Contents/MacOS/RStudio
```

```
  com.apple.security.device.camera

  com.apple.security.device.audio-input

  com.apple.security.cs.disable-library-validation

  com.apple.security.cs.disable-executable-page-protection

  com.apple.security.cs.allow-dyld-environment-variables

  com.apple.security.automation.apple-events
```

The output is (ugh) XML, and don't think that all app developers are as awesome as RStudio ones since those comments are pseudo-optional (i.e. you can put junk in them). I'll continue to use RStudio throughout this example just for consistency.

Since you likely have better things to do than execute a command line tool multiple times and do significant damage to your eyes with all those pointy tags we can use R to turn the apps on our filesystem into data and examine the entitlements in a much more dignified manner.

First, we'll write a function to wrap the `codesign` tool execution (and, I've leaked how were going to eventually look at them by putting all the library calls up front):

```
library(XML)
library(tidyverse)
library(igraph)
library(tidygraph)
library(ggraph)

# rewriting this to also grab the text from the comments is an exercise
left to the reader

read_entitlements <- function(app) {

  system2(
    command = "codesign",
    args = c(
      "-d",
      "--entitlements",
      ":-",
      gsub(" ", "\\\\ ", app)
    ),
    stdout = TRUE
  ) -> x

  x <- paste0(x, collapse = "\n")

  if (nchar(x) == 0) return(tibble())

  x <- XML::xmlParse(x, asText=TRUE)
  x <- try(XML::readKeyValueDB(x), silent = TRUE)

  if (inherits(x, "try-error")) return(tibble())

  x <- sapply(x, function(.x) paste0(.x, collapse=";"))

  if (length(x) == 0) return(tibble())

  data.frame(
    app = basename(app),
    entitlement = make.unique(names(x)),
    value = I(x)
  ) -> x

  x <- tibble::as_tibble(x)

  x

}
```

Now, we can slurp up all the entitlements with just a few lines of code:

```
my_apps <- list.files("/Applications", pattern = "\\.app$", full.names
= TRUE)

my_apps_entitlements <- map_df(my_apps, read_entitlements)

my_apps_entitlements %>%
  filter(grepl("RStudio", app))
## # A tibble: 6 x 3
##   app         entitlement
value
##                                                                  >
## 1 RStudio.app com.apple.security.device.camera
TRUE
## 2 RStudio.app com.apple.security.device.audio-input
TRUE
## 3 RStudio.app com.apple.security.cs.disable-library-validation
TRUE
## 4 RStudio.app com.apple.security.cs.disable-executable-page-
protection TRUE
## 5 RStudio.app com.apple.security.cs.allow-dyld-environment-variables
TRUE
## 6 RStudio.app com.apple.security.automation.apple-events
TRUE
```

Having these entitlement strings is great, but what do they mean? Unfortunately, Apple, frankly, *sucks* at developer documentation, and this suckage shines especially bright when it comes to documenting *all* the possible entitlements. We *can* retrieve some of them from the online documentation, so let's do that and re-look at RStudio:

```
# a handful of fairly ok json URLs that back the online dev docs; they
have ok, but scant entitlement definitions
c(
  "https://developer.apple.com/tutorials/data/documentation/bundleresources/entitlements.
json",
  "https://developer.apple.com/tutorials/data/documentation/security/app_sandbox.json",
  "https://developer.apple.com/tutorials/data/documentation/security/hardened_runtime.json",
  "https://developer.apple.com/tutorials/data/documentation/bundleresources/entitlements/
system_extensions.json"
) -> entitlements_info_urls

extract_entitlements_info <- function(x) {

  apple_ents_pg <- jsonlite::fromJSON(x)

  apple_ents_pg$references %>%
    map_df(~{

      if (!hasName(.x, "role")) return(tibble())
      if (.x$role != "symbol") return(tibble())

      tibble(
        title = .x$title,
```

```
        entitlement = .x$name,
        description = .x$abstract$text %||% NA_character_
      )

    })

}

entitlements_info_urls %>%
  map(extract_ents_info) %>%
  bind_rows() %>%
  distinct() -> apple_entitlements_definitions

# look at rstudio again ------------------------------
---------------------

my_apps_entitlements %>%
  left_join(apple_entitlements_definitions) %>%
  filter(grepl("RStudio", app)) %>%
  select(title, description)
## Joining, by = "entitlement"
## # A tibble: 6 x 2
##   title                            description
##
## 1 Camera Entitlement               A Boolean value that indicates
whether the app may capture movies…
## 2 Audio Input Entitlement          A Boolean value that indicates
whether the app may record audio u…
## 3 Disable Library Validation Enti… A Boolean value that indicates
whether the app may load arbitrary…
## 4 Disable Executable Memory Prote… A Boolean value that indicates
whether to disable all code signin…
## 5 Allow DYLD Environment Variable… A Boolean value that indicates
whether the app may be affected by…
## 6 Apple Events Entitlement         A Boolean value that indicates
whether the app may prompt the use…
```

It might be interesting to see what the most requested entitlements are:

```
my_apps_entitlements %>%
  filter(
    grepl("security", entitlement)
  ) %>%
  count(entitlement, sort = TRUE)
## # A tibble: 60 x 2
##    entitlement                                          n
##
##  1 com.apple.security.app-sandbox                      51
##  2 com.apple.security.network.client                   44
##  3 com.apple.security.files.user-selected.read-write   35
##  4 com.apple.security.application-groups               29
##  5 com.apple.security.automation.apple-events          26
```

```
##  6 com.apple.security.device.audio-input                19
##  7 com.apple.security.device.camera                     17
##  8 com.apple.security.files.bookmarks.app-scope         16
##  9 com.apple.security.network.server                    16
## 10 com.apple.security.cs.disable-library-validation     15
## # … with 50 more rows
```

Playing in an app sandbox, talking to the internet, and handling files are unsurprising in the top three slots since that's how most apps get stuff done for you.

There are a few entitlements which increase your attack surface, one of which is apps that use untrusted third-party libraries:

```
my_apps_entitlements %>%
  filter(
    entitlement == "com.apple.security.cs.disable-library-validation"
  ) %>%
  select(app)
## # A tibble: 15 x 1
##    app
##
##  1 Epic Games Launcher.app
##  2 GarageBand.app
##  3 HandBrake.app
##  4 IINA.app
##  5 iStat Menus.app
##  6 krisp.app
##  7 Microsoft Excel.app
##  8 Microsoft PowerPoint.app
##  9 Microsoft Word.app
## 10 Mirror for Chromecast.app
## 11 Overflow.app
## 12 R.app
## 13 RStudio.app
## 14 RSwitch.app
## 15 Wireshark.app
```

('Tis ironic that one of Apple's own apps is in that list.)

What about apps that listen on the network (i.e. are also servers)?

```
## # A tibble: 16 x 1
##    app
##
##  1 1Blocker.app
##  2 1Password 7.app
##  3 Adblock Plus.app
##  4 Divinity - Original Sin 2.app
##  5 Fantastical.app
##  6 feedly.app
##  7 GarageBand.app
##  8 iMovie.app
##  9 Keynote.app
```

```
## 10 Kindle.app
## 11 Microsoft Remote Desktop.app
## 12 Mirror for Chromecast.app
## 13 Slack.app
## 14 Tailscale.app
## 15 Telegram.app
## 16 xScope.app
```

You should read through the retrieved definitions to see what else you may want to observe to be an informed macOS app user.

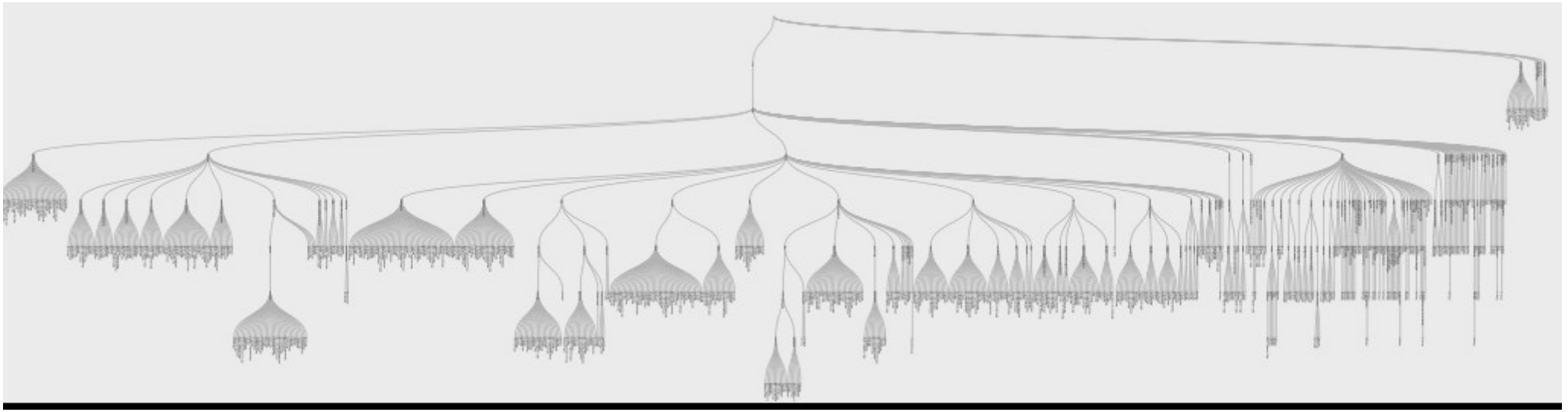
The Big Picture

Looking at individual apps is great, but why not look at *them all*? We can build a large, but searchable network graph hierarchy if we output it as PDf, so let's do that:

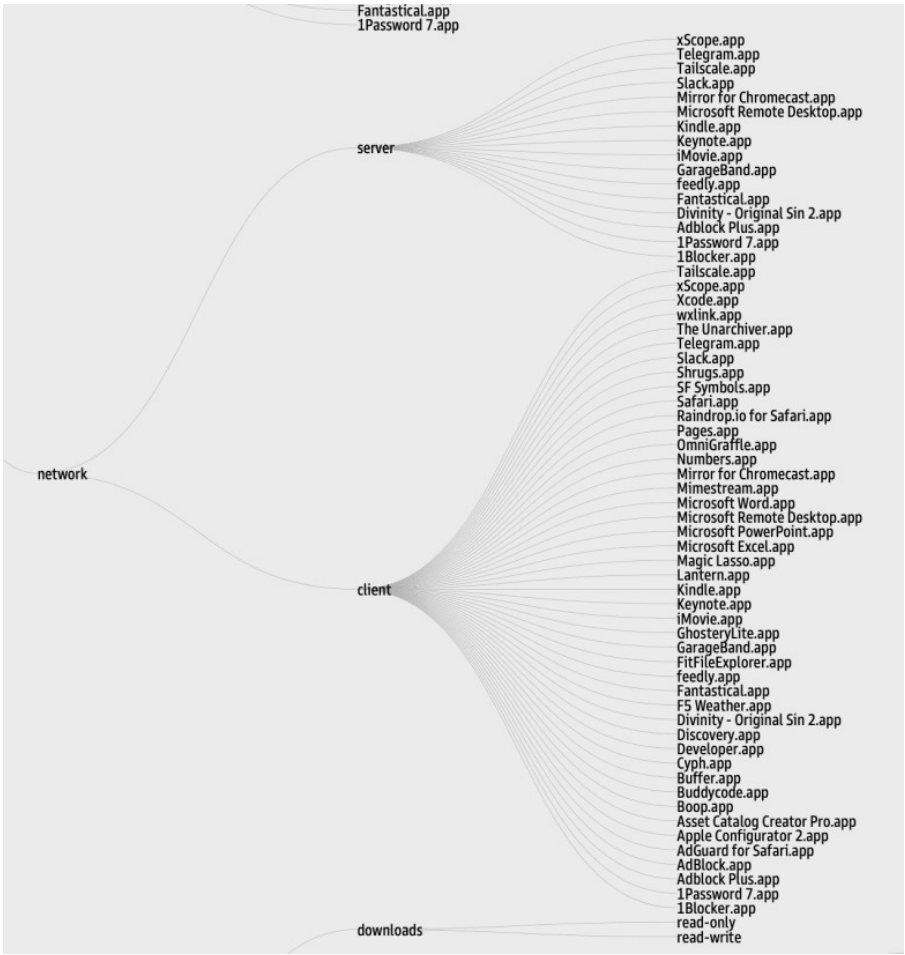
```
# this is just some brutish force code to build a hierarchical edge
list

my_apps_entitlements %>%
  distinct(entitlement) %>%
  pull(entitlement) %>%
  stri_count_fixed(".") %>%
  max() -> max_dots

my_apps_entitlements %>%
  distinct(entitlement, app) %>%
  separate(
    entitlement,
    into = sprintf("level_%02d", 1:(max_dots+1)),
    fill = "right",
    sep = "\\."
  ) %>%
  select(
    starts_with("level"), app
  ) -> wide_hierarchy

bind_rows(

  distinct(wide_hierarchy, level_01) %>%
    rename(to = level_01) %>%
    mutate(from = ".") %>%
    select(from, to) %>%
    mutate(to = sprintf("%s_1", to)),

  map_df(1:nrow(wide_hierarchy), ~{

    wide_hierarchy[.x,] %>%
      unlist(use.names = FALSE) %>%
      na.exclude() -> tmp

    tibble(
      from = tail(lag(tmp), -1),
```

```
      to = head(lead(tmp), -1),
      lvl = 1:length(from)
    ) %>%
      mutate(
        from = sprintf("%s_%d", from, lvl),
        to = sprintf("%s_%d", to, lvl+1)
      )

  }) %>%
    distinct()

) -> long_hierarchy

# all that so we can make a pretty graph!
---------------------------------

g <- graph_from_data_frame(long_hierarchy, directed = TRUE)

ggraph(g, 'partition', circular = FALSE) +
  geom_edge_diagonal2(
    width = 0.125, color = "gray70"
  ) +
  geom_node_text(
    aes(
      label = stri_replace_last_regex(name, "_[[:digit:]]+$", "")
    ),
    hjust = 0, size = 3, family = font_gs
  ) +
  coord_flip() -> gg

# saving as PDF b/c it is ginormous, but very searchable

quartz(
  file = "~/output-tmp/.pdf", # put it where you want
  width = 21,
  height = 88,
  type = "pdf",
  family = font_gs
)
print(gg)
dev.off()
```

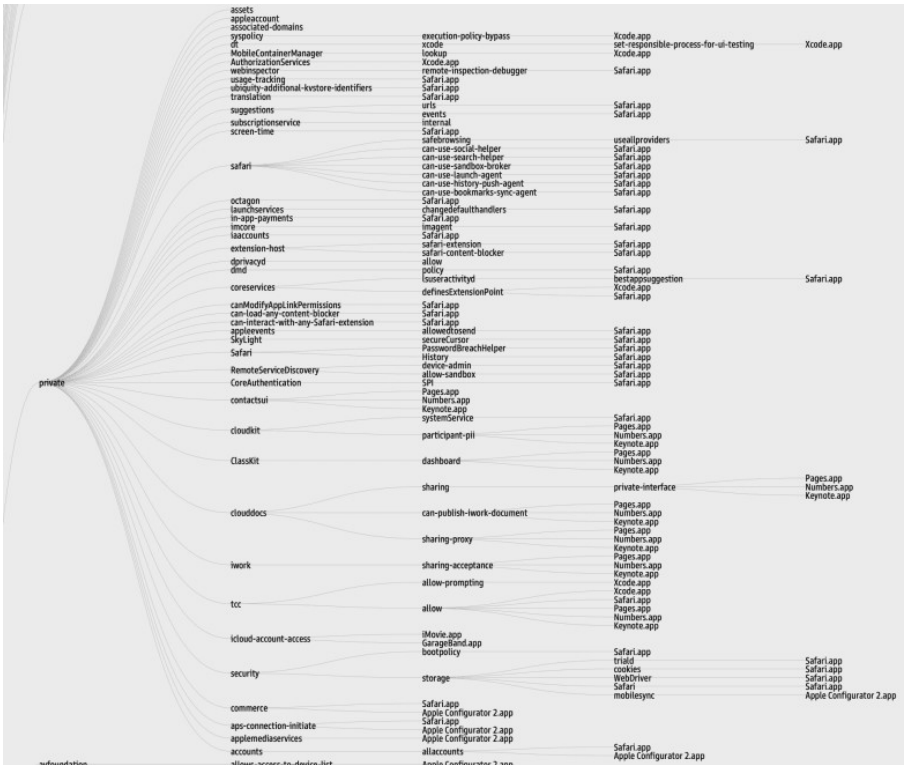
The above generates a *large* (dimension-wise; it's ~<5MB on disk for me) PDF graph that is barely viewable in thumnail mode:



Here are some screen captures of portions of it. First are all network servers and clients:



Last are seekrit entitlements only for Apple:



FIN

I'll likely put a few of these functions into {mactheknife} for easier usage.

After going through this exercise I deleted 11 apps, some for their entitlements and others that I just never use anymore. Hopefully this will help you do some early Spring cleaning as well.