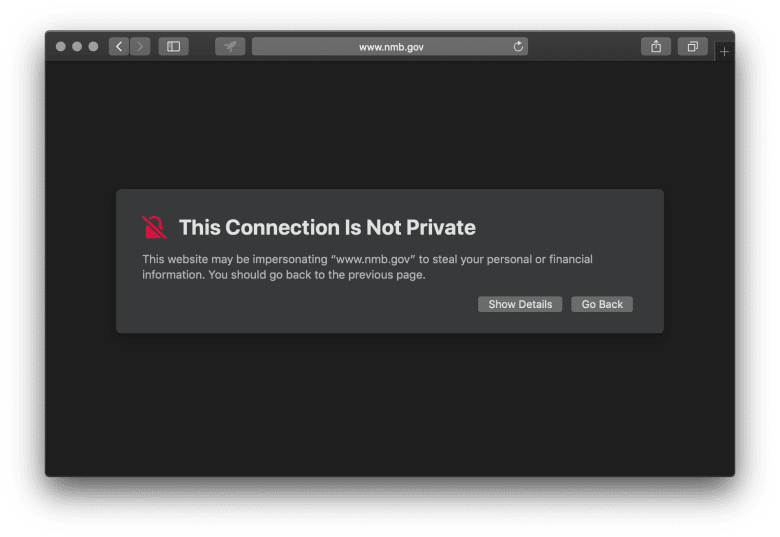
By now, even remote villages on uncharted islands in the Pacific know that the U.S. is in the midst of a protracted partial government shutdown. It’s having real impacts on the lives of Federal government workers but they aren’t the only ones. Much of the interaction Federal agencies have with the populace takes place online and the gateway to most of these services/information is a web site.

There are Federal standards that require U.S. government web sites to use SSL/TLS certificates and those certificates have something in common with, say, a loaf of bread you buy at the store: they expire. In all but the best of orgs — or we zany folks who use L e t ‘ s E n c r y p t and further propel internet denizens into a false sense of safety & privacy — renewing certificates involves manual labor/human intervention. For a good chunk of U.S. Federal agencies, those particular humans aren’t around. If a site’s SSL certificate expires and isn’t re-issued, it causes browsers to do funny things, like this:

[](https://rud.is/b/2019/01/10/roll-your-own-federal-government-shutdown-caused-ssl-certificate-expiration-monitor-in-r/screen-shot-2019-01-10-at-17-18-56/)

Now, *some* of these sites are configured improperly in many ways, including them serving pages on both http and https (vs redirecting to https immediately upon receiving an http connection). But, browsers like Chrome will generally try https first and scare you into not viewing the site.

But, how big a problem could this really be? We can find out with a fairly diminutive R script that:

* grabs a list of Federal agency domains (thanks to [the GSA](https://github.com/GSA/data))
* tries to make a SSL/TLS connection (via the openssl package) to the apex domain or www. prefixed apex domain
* find the expiration date for the cert
* do some simple date math

I’ve commented the script below pretty well so I’ll refrain from further blathering:

library(furrr)

library(openssl)

library(janitor)

library(memoise)

library(hrbrthemes)

library(tidyverse)

# fetch the GSA CSV:

read\_csv(

file = "https://raw.githubusercontent.com/GSA/data/master/dotgov-domains/current-federal.csv",

col\_types = "ccccccc"

) %>%

janitor::clean\_names() -> xdf

# make openssl::download\_ssl\_cert calls safer in the even there

# are network/connection issues

.dl\_cert <- possibly(openssl::download\_ssl\_cert, otherwise = NULL)

# memoise the downloader just in case we need to break the iterator

# below or another coding error causes it to break (the cached values

# will go away in a new R session or if you manually purge them)

dl\_cert <- memoise::memoise(.dl\_cert)

# we'll do this in parallel to save time (~1,200 domains)

plan(multiprocess)

# now follow the process described in the bullet points

future\_map\_dfr(xdf$domain\_name, ~{

who <- .x

crt <- dl\_cert(who)

if (!is.null(crt)) {

# shld be the first cert and expires is second validity field

expires <- crt[[1]]$validity[2]

} else {

crt <- dl\_cert(sprintf("www.%s", who)) # may be on www b/c "gov"

if (!is.null(crt)) {

expires <- crt[[1]]$validity[2]

} else {

expires <- NA\_character\_

}

}

# keep a copy of the apex domain, the expiration field and the cert

# (in the event you want to see just how un-optimized the U.S. IT

# infrastructure is by how many stupid vendors they use for certs)

tibble(

who = who,

expires = expires,

cert = list(crt)

)

}) -> cdf

Now, lets make strings into proper dates, count only the dates starting with the date of the shutdown to the end of 2019 (b/c the reckless human at the helm is borderline insane enough to do that) and plot the timeline:

filter(cdf, !is.na(expires)) %>%

mutate(

expires = as.Date(

as.POSIXct(expires, format="%b %d %H:%M:%S %Y")

)

) %>%

arrange(expires)

count(expires) %>%

filter(

expires >= as.Date("2018-12-22"),

expires <= as.Date("2019-12-31")

) %>%

ggplot(aes(expires, n)) +

geom\_vline(

xintercept = Sys.Date(), linetype="dotted", size=0.25, color = "white"

) +

geom\_label(

data = data.frame(),

aes(x = Sys.Date(), y = Inf, label = "Today"),

color = "black", vjust = 1

) +

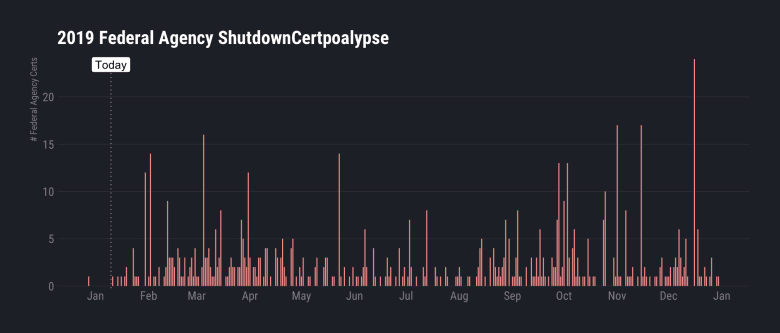
geom\_segment(aes(xend=expires, yend=0), color = ft\_cols$peach) +

scale\_x\_date(name=NULL, date\_breaks="1 month", date\_labels="%b") +

scale\_y\_comma("# Federal Agency Certs") +

labs(title = "2019 Federal Agency ShutdownCertpoalypse") +

theme\_ft\_rc(grid="Y")

[](https://rud.is/b/2019/01/10/roll-your-own-federal-government-shutdown-caused-ssl-certificate-expiration-monitor-in-r/plot_zoom_png-14/)

Now, I’m unwarrantedly optimistic that this debacle could be over by the end of January. How many certs (by agency) could go bad by then?

left\_join(cdf, xdf, by=c("who"="domain\_name")) %>%

mutate(

expires = as.Date(

as.POSIXct(expires, format="%b %d %H:%M:%S %Y")

)

) %>%

filter(

expires >= as.Date("2018-12-22"),

expires <= as.Date("2019-01-31")

) %>%

count(agency, sort = TRUE)

## # A tibble: 10 x 2

## agency n

##

## 1 Government Publishing Office 8

## 2 Department of Commerce 4

## 3 Department of Defense 3

## 4 Department of Housing and Urban Development 3

## 5 Department of Justice 3

## 6 Department of Energy 1

## 7 Department of Health and Human Services 1

## 8 Department of State 1

## 9 Department of the Interior 1

## 10 Department of the Treasury 1

Ugh.

**FIN**

Not *every* agency is fully shutdown and not *all* workers in charge of cert renewals are furloughed (or being forced to work without pay). But, this one other area shows the possible unintended consequences of making rash, partisan decisions (something both Democrats & Republicans excel at).