Phishing is [still] the primary way attackers either commit a primary criminal act (i.e. phish a target to, say, install ransomware) or is the initial vehicle used to gain a foothold in an organization so they can perform other criminal operations to achieve some goal. As such, security teams, vendors and active members of the cybersecurity community work diligently to neutralize phishing campaigns as quickly as possible.

One popular community tool/resource in this pursuit is [PhishTank](https://www.phishtank.com/) which *is a collaborative clearing house for data and information about phishing on the Internet. Also, PhishTank provides an open API for developers and researchers to integrate anti-phishing data into their applications at no charge.*

While the PhishTank API is useful for real-time anti-phishing operations the data is also useful for security researchers as we work to understand the ebb, flow and evolution of these attacks. One avenue of research is to track the various features associated with phishing campaigns which include (amongst many other elements) network (internet) location of the phishing site, industry being targeted, domain names being used, what type of sites are being cloned/copied and a feature we’ll be looking at in this post: what percentage of new phishing sites use SSL encryption and — of these — which type of SSL certificates are “en vogue”.

Phishing sites are increasingly using and relying on SSL certificates because we in the information security industry spent a decade instructing the general internet surfing population to trust sites with the green lock icon near the location bar. Initially, phishers worked to compromise existing, encryption-enabled web properties to install phishing sites/pages since they could leech off of the “trusted” status of the associated SSL certificates. However, the advent of services like [Let’s Encrypt](https://letsencrypt.org/) have made it possible for attacker to setup their own phishing domains that look legitimate to current-generation internet browsers and prey upon the decade’s old “trust the lock icon” mantra that most internet users still believe. We’ll table that path of discussion (since it’s fraught with peril if you don’t support the internet-do-gooder-consequences-be-darned cabal’s personal agendas) and just focus on how to work with PhishTank data in R and take a look at the most prevalent SSL certs used in the past week (you can extend the provided example to go back as far as you like provided the phishing sites are still online).

**Accessing PhishTank From R**

You can use the aquarium package [[GL](https://gitlab.com/hrbrmstr/aquarium)|[GH](https://gihub.com/hrbrmstr/aquarium)] to gain access to the data provided by PhishTank’s API (you need to sign up for access and put you API key into the PHISHTANK\_API\_KEY environment variable which is best done via your ~/.Renviron file).

Let’s setup all the packages we’ll need and cache a current copy of the PhishTank data. The package forces you to utilize your own caching strategy since it doesn’t make sense for it to decide that for you. I’d suggest either using the time-stamped approach below or using some type of database system (or, say, Apache Drill) to actually manage the data.

Here are the packages we’ll need:

library(psl) # git[la|hu]b/hrbrmstr/psl

library(curlparse) # git[la|hu]b/hrbrmstr/curlparse

library(aquarium) # git[la|hu]b/hrbrmstr/aquarium

library(gt) # github/rstudio/gt

library(furrr)

library(stringi)

library(openssl)

library(tidyverse)

NOTE: The psl and curlparse packages are optional. Windows users will find it difficult to get them working and it may be easier to review the functions provided by the urlparse package and substitute equivalents for the domain() and apex\_domain() functions used below. Now, we get a copy of the current PhishTank dataset & cache it:

if (!file.exists("~/Data/2018-12-23-fishtank.rds")) {

xdf <- pt\_read\_db()

saveRDS(xdf, "~/Data/2018-12-23-fishtank.rds")

} else {

xdf <- readRDS("~/Data/2018-12-23-fishtank.rds")

}

Let’s take a look:

glimpse(xdf)

## Observations: 16,446

## Variables: 9

## $ phish\_id "5884184", "5884138", "5884136", "5884135", ...

## $ url "http://internetbanking-bancointer.com.br/lo...

## $ phish\_detail\_url "http://www.phishtank.com/phish\_detail.php?p...

## $ submission\_time 2018-12-22 20:45:09, 2018-12-22 18:40:24, 2...

## $ verified "yes", "yes", "yes", "yes", "yes", "yes", "y...

## $ verification\_time 2018-12-22 20:45:52, 2018-12-22 21:26:49, 2...

## $ online "yes", "yes", "yes", "yes", "yes", "yes", "y...

## $ details [<209.132.252.7, 209.132.252.0/24, 7296 468...

## $ target "Other", "Other", "Other", "PayPal", "Other"...

The data is really straightforward. We have unique ids for each site/campaign the URL of the site along with a URL to extra descriptive info PhishTank has on the site/campaign. We also know when the site was submitted/discovered and other details, such as the network/internet space the site is in:

glimpse(xdf$details[1])

## List of 1

## $ :'data.frame': 1 obs. of 6 variables:

## ..$ ip\_address : chr "209.132.252.7"

## ..$ cidr\_block : chr "209.132.252.0/24"

## ..$ announcing\_network: chr "7296 468"

## ..$ rir : chr "arin"

## ..$ country : chr "US"

## ..$ detail\_time : chr "2018-12-23T01:46:16+00:00"

We’re going to focus on recent phishing sites (in this case, ones that are less than a week old) and those that use SSL certificates:

filter(xdf, verified == "yes") %>%

filter(online == "yes") %>%

mutate(diff = as.numeric(difftime(Sys.Date(), verification\_time), "days")) %>%

filter(diff <= 7) %>%

{ all\_ct <<- nrow(.) ; . } %>%

filter(grepl("^https", url)) %>%

{ ssl\_ct <<- nrow(.) ; . } %>%

mutate(

domain = domain(url),

apex = apex\_domain(domain)

) -> recent

Let’s ee how many are using SSL:

(ssl\_ct)

## [1] 383

(pct\_ssl <- ssl\_ct / all\_ct)

## [1] 0.2919207

This percentage is lower than a recent “50% of all phishing sites use encryption” statistic going around of late. There are many reasons for the difference:

* PhishTank doesn’t have *all* phishing sites in it
* We just looked at a week of examples
* Some sites were offline at the time of access attempt
* Diverse attacker groups with varying degrees of competence engage in phishing attacks

Despite the 20% deviation, 30% is still a decent percentage, and a green, “everything’s ” icon is a still a valued prize so we shall pursue our investigation.

Now we need to retrieve all those certs. This can be a slow operation that so we’ll grab them in parallel. It’s also quite possible the “online”status above data frame glimpse is inaccurate (sites can go offline quickly) so we’ll catch certificate request failures with safely() and cache the results:

cert\_dl <- purrr::safely(openssl::download\_ssl\_cert)

plan(multiprocess)

if (!file.exists("~/Data/recent.rds")) {

recent <- mutate(recent, cert = future\_map(domain, cert\_dl))

saveRDS(recent, "~/Data/recent.rds")

} else {

recent <- readRDS("~/Data/recent.rds")

}

Let see how many request failures we had:

(failed <- sum(map\_lgl(recent$cert, ~is.null(.x$result))))

## [1] 25

(failed / nrow(recent))

## [1] 0.06527415

As noted in the introduction to the blog, when attackers want to use SSL for the lock icon ruse they can either try to piggyback off of legitimate domains or rely on Let’s Encrypt to help them commit crimes. Let’s see what the top p”apex” domains](https://help.github.com/articles/about-supported-custom-domains/#apex-domains) were in use in the past week:

count(recent, apex, sort = TRUE)

## # A tibble: 255 x 2

## apex n

##

## 1 000webhostapp.com 42

## 2 google.com 17

## 3 umbler.net 8

## 4 sharepoint.com 6

## 5 com-fl.cz 5

## 6 lbcpzonasegurabeta-viabcp.com 4

## 7 windows.net 4

## 8 ashaaudio.net 3

## 9 brijprints.com 3

## 10 portaleisp.com 3

## # ... with 245 more rows

We can see that a large hosting provider (000webhostapp.com) bore a decent number of these sites, but Google Sites (which is what the full domain represented by the google.com apex domain here is usually pointing to) Microsoft SharePoint (sharepoint.com) and Microsoft forums (windows.net) are in active use as well (which is smart give the pervasive trust associated with those properties). There are 241 distinct apex domains in this 1-week set so what is the SSL cert diversity across these pages/campaigns?

We ultimately used openssl::download\_ssl\_cert to retrieve the SSL certs of each site that was online, so let’s get the [issuer and intermediary certs](https://letsencrypt.org/certificates/) from them and look at the prevalence of each. We’ll extract the fields from the issuer component returned by openssl::download\_ssl\_cert then just do some basic maths:

filter(recent, map\_lgl(cert, ~!is.null(.x$result))) %>%

mutate(issuers = map(cert, ~map\_chr(.x$result, ~.x$issuer))) %>%

mutate(

inter = map\_chr(issuers, ~.x[1]), # the order is not guaranteed here but the goal of the exercise is

root = map\_chr(issuers, ~.x[2]) # to get you working with the data vs build a 100% complete solution

) %>%

mutate(

inter = stri\_replace\_all\_regex(inter, ",([[:alpha:]])+=", ";;;$1=") %>%

stri\_split\_fixed(";;;") %>% # there are parswers for the cert info fields but this hack is quick and works

map(stri\_split\_fixed, "=", 2, simplify = TRUE) %>%

map(~setNames(as.list(.x[,2]), .x[,1])) %>%

map(bind\_cols),

root = stri\_replace\_all\_regex(root, ",([[:alpha:]])+=", ";;;$1=") %>%

stri\_split\_fixed(";;;") %>%

map(stri\_split\_fixed, "=", 2, simplify = TRUE) %>%

map(~setNames(as.list(.x[,2]), .x[,1])) %>%

map(bind\_cols)

) -> recent

Let’s take a look at roots:

unnest(recent, root) %>%

distinct(phish\_id, apex, CN) %>%

count(CN, sort = TRUE) %>%

mutate(pct = n/sum(n)) %>%

gt::gt() %>%

gt::fmt\_number("n", decimals = 0) %>%

gt::fmt\_percent("pct")

|  |  |  |
| --- | --- | --- |
| **CN** | **n** | **pct** |
| DST Root CA X3 | 96 | 26.82% |
| COMODO RSA Certification Authority | 93 | 25.98% |
| DigiCert Global Root G2 | 45 | 12.57% |
| Baltimore CyberTrust Root | 30 | 8.38% |
| GlobalSign | 27 | 7.54% |
| DigiCert Global Root CA | 15 | 4.19% |
| Go Daddy Root Certificate Authority – G2 | 14 | 3.91% |
| COMODO ECC Certification Authority | 11 | 3.07% |
| Actalis Authentication Root CA | 9 | 2.51% |
| GlobalSign Root CA | 4 | 1.12% |
| Amazon Root CA 1 | 3 | 0.84% |
| Let’s Encrypt Authority X3 | 3 | 0.84% |
| AddTrust External CA Root | 2 | 0.56% |
| DigiCert High Assurance EV Root CA | 2 | 0.56% |
| USERTrust RSA Certification Authority | 2 | 0.56% |
| GeoTrust Global CA | 1 | 0.28% |
| SecureTrust CA | 1 | 0.28% |

DST Root CA X3 is (wait for it) *Let’s Encrypt*! Now, Comodo is not far behind and indeed surpasses LE if we combine the extra-special “enhanced” versions they provide and it’s important for you to read the comments near the lines of code making assumptions about order of returned issuer information above. Now, let’s take a look at intermediaries:

unnest(recent, inter) %>%

distinct(phish\_id, apex, CN) %>%

count(CN, sort = TRUE) %>%

mutate(pct = n/sum(n)) %>%

gt::gt() %>%

gt::fmt\_number("n", decimals = 0) %>%

gt::fmt\_percent("pct")

|  |  |  |
| --- | --- | --- |
| **CN** | **n** | **pct** |
| Let’s Encrypt Authority X3 | 99 | 27.65% |
| cPanel\, Inc. Certification Authority | 75 | 20.95% |
| RapidSSL TLS RSA CA G1 | 45 | 12.57% |
| Google Internet Authority G3 | 24 | 6.70% |
| COMODO RSA Domain Validation Secure Server CA | 20 | 5.59% |
| CloudFlare Inc ECC CA-2 | 18 | 5.03% |
| Go Daddy Secure Certificate Authority – G2 | 14 | 3.91% |
| COMODO ECC Domain Validation Secure Server CA 2 | 11 | 3.07% |
| Actalis Domain Validation Server CA G1 | 9 | 2.51% |
| RapidSSL RSA CA 2018 | 9 | 2.51% |
| Microsoft IT TLS CA 1 | 6 | 1.68% |
| Microsoft IT TLS CA 5 | 6 | 1.68% |
| DigiCert SHA2 Secure Server CA | 5 | 1.40% |
| Amazon | 3 | 0.84% |
| GlobalSign CloudSSL CA – SHA256 – G3 | 2 | 0.56% |
| GTS CA 1O1 | 2 | 0.56% |
| AlphaSSL CA – SHA256 – G2 | 1 | 0.28% |
| DigiCert SHA2 Extended Validation Server CA | 1 | 0.28% |
| DigiCert SHA2 High Assurance Server CA | 1 | 0.28% |
| Don Dominio / MrDomain RSA DV CA | 1 | 0.28% |
| GlobalSign Extended Validation CA – SHA256 – G3 | 1 | 0.28% |
| GlobalSign Organization Validation CA – SHA256 – G2 | 1 | 0.28% |
| RapidSSL SHA256 CA | 1 | 0.28% |
| TrustAsia TLS RSA CA | 1 | 0.28% |
| USERTrust RSA Domain Validation Secure Server CA | 1 | 0.28% |
| NA | 1 | 0.28% |

LE is number one again! But, it’s important to note that these [issuer CommonNames](https://docs.oracle.com/cd/E24191_01/common/tutorials/authz_cert_attributes.html) can roll up into a single issuing organization given just how messed up integrity and encryption capability is when it comes to web site certs, so the raw results could do with a bit of post-processing for a more complete picture (an exercise left to intrepid readers).

**FIN**

There are *tons* of avenues to explore with this data, so I hope this post whet your collective appetites sufficiently for you to dig into it, especially if you have some dowm-time coming.

Let me also take this opportunity to resissue guidance I and many others have uttered this holiday season: be *super careful* about what you click on, which sites you even just visit, and just how much you *really* trust the site, provider and entity behind the form about to enter your personal information and credit card info into.