

dns_achados

Rafilx

2022-04-14

R Markdown

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##   filter, lag

## The following objects are masked from 'package:base':
##
##   intersect, setdiff, setequal, union

## Loading required package: viridisLite

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##   date, intersect, setdiff, union
```

- Busca os dados no banco com o parse do DNS ja realizado, então temos:
 - qname que é o domínio
 - qtype tipo da query
 - query_id ID da transação definido pelo atacante
 - year_period ano e trimestre em que ocorreu o ataque exemplo “20212” o ataque ocorreu no segundo trimestre do 2021

```
db <- dbConnect(RSQLite::SQLite(), dbname="./dnstor_statistics_dns.sqlite")

data_unfetch <-dbSendQuery(db, "
  SELECT *, CAST(CAST(year AS text) || CAST(period AS text) as integer) as year_period
  FROM DNS_ANALYSIS
  JOIN DNS_ANALYSIS_QUESTION
    ON DNS_ANALYSIS.id = DNS_ANALYSIS_QUESTION.dns_analysis_id
  WHERE QTYPE != 0
")
data <- fetch(data_unfetch)
```

```

dns_data_unfetch <- dbSendQuery(db, "
  SELECT count(*) as countGrouped, year, period, CAST(CAST(year AS text) || CAST(period AS text) as int)
  FROM DNS_ANALYSIS
  JOIN DNS_ANALYSIS_QUESTION
  ON DNS_ANALYSIS.id = DNS_ANALYSIS_QUESTION.dns_analysis_id
  WHERE QTYPE != 0
GROUP BY year_period, year, period, qname, qtype
ORDER BY quantity DESC;
")

```

Warning: Closing open result set, pending rows

```

dns_data_fetched <- fetch(dns_data_unfetch)

dns_data_overlap_unfetch <- dbSendQuery(db, "
  SELECT *
  FROM TB_DATE_OVERLAP_QUERYID
ORDER BY amount_overlap;
")

```

Warning: Closing open result set, pending rows

```

dns_data_overlap_fetched <- fetch(dns_data_overlap_unfetch)

dbDisconnect(db)

```

Warning in connection_release(conn@ptr): There are 1 result in use. The
connection will be released when they are closed

- Primeiro separa todos os registros por trimestre

```

data_split_year_period = data %>%
  group_split(year_period)

```

- Gerando um total de 6 trimestres

```

N=10

period_query_id = data.frame()
for (i in c(1:length(data_split_year_period))) {
  query_id_frequency = data_split_year_period[[i]] %>%
    count(query_id)

  query_id_frequency['year_period'] = data_split_year_period[[i]]$year_period[1]

  period_query_id = rbind(period_query_id, head(query_id_frequency[order(-query_id_frequency$n),], N) )
}

```

Os 10 query_id mais utilizados divididos por período e ordenados pela frequência em que apareceram no período

```
period_query_id %>%
  group_split(year_period)
```

```
## <list_of<
##   tbl_df<
##     query_id    : integer
##     n           : integer
##     year_period: integer
##   >
## >[6]>
## [[1]]
## # A tibble: 10 x 3
##   query_id      n year_period
##   <int> <int>    <int>
## 1  17767  1917    20204
## 2  16049  1060    20204
## 3  56064  1049    20204
## 4  63710   782    20204
## 5  59797   741    20204
## 6  63374   729    20204
## 7  59378   723    20204
## 8  31694   718    20204
## 9  13304   715    20204
## 10 46512   707    20204
##
## [[2]]
## # A tibble: 10 x 3
##   query_id      n year_period
##   <int> <int>    <int>
## 1  17767 67047    20211
## 2  28940   418    20211
## 3  13551   318    20211
## 4  50265   305    20211
## 5  19592   277    20211
## 6  45810   214    20211
## 7  57166   197    20211
## 8  43855   168    20211
## 9  56643   125    20211
## 10 56686   124    20211
##
## [[3]]
## # A tibble: 10 x 3
##   query_id      n year_period
##   <int> <int>    <int>
## 1  26566   5090    20212
## 2  17767   3748    20212
## 3  13551   348    20212
## 4  50265   118    20212
## 5  37845    81    20212
## 6      1     65    20212
```

```

## 7      36379      60      20212
## 8      45810      59      20212
## 9       1525      47      20212
## 10     40074      38      20212
##
## [[4]]
## # A tibble: 10 x 3
##   query_id      n year_period
##   <int> <int>    <int>
## 1         1  6859    20213
## 2      17767  5838    20213
## 3      13551   783    20213
## 4         27   403    20213
## 5      59252   253    20213
## 6      60765   220    20213
## 7      13143   212    20213
## 8      53342   157    20213
## 9      65372   102    20213
## 10     14262   100    20213
##
## [[5]]
## # A tibble: 10 x 3
##   query_id      n year_period
##   <int> <int>    <int>
## 1     26566 29963    20214
## 2      17767  8479    20214
## 3          1   677    20214
## 4      13551   566    20214
## 5         27   102    20214
## 6     28826    70    20214
## 7     36609    69    20214
## 8       3803    60    20214
## 9     47132    58    20214
## 10     50265    51    20214
##
## [[6]]
## # A tibble: 10 x 3
##   query_id      n year_period
##   <int> <int>    <int>
## 1     26566  5876    20221
## 2      17767   895    20221
## 3      13551   124    20221
## 4          1   120    20221
## 5     64206    63    20221
## 6     28826    41    20221
## 7         27    29    20221
## 8     14602    19    20221
## 9     50293    15    20221
## 10          6    12    20221

```

- Dessa forma é possível observar que o mesmo query_id é utilizado várias vezes durante o mesmo trimestre, como no primeiro trimestre de 2021 que o query_id 17767 foi utilizado em 67047 ataques

Os 10 query_id mais utilizados

```
data %>%
  count(query_id) %>%
  arrange(desc(n)) %>%
  head(N)
```

```
##      query_id      n
## 1      17767 87924
## 2      26566 40940
## 3         1  7803
## 4      13551  2347
## 5      16049  1062
## 6      56064  1054
## 7      63710   785
## 8      59797   746
## 9      63374   738
## 10     46512   728
```

Os 10 query_id mais utilizados em cada período ordenados pela frequência em que apareceram levando em consideração todos os períodos

```
period_query_id %>%
  arrange(desc(n)) %>%
  head(N)
```

```
## # A tibble: 10 x 3
##   query_id      n year_period
##   <int> <int>    <int>
## 1    17767 67047    20211
## 2    26566 29963    20214
## 3    17767  8479    20214
## 4         1  6859    20213
## 5    26566  5876    20221
## 6    17767  5838    20213
## 7    26566  5090    20212
## 8    17767  3748    20212
## 9    17767  1917    20204
## 10   16049  1060    20204
```

Os query_id que apareceram com maior frequência entre os top 10 em todos os períodos

- Caso o query_id 13213 fosse top 1 em 20204 e top 3 em 20211 e não aparecer em mais nenhum outro período seu “n” seria 2

```
period_query_id %>%
  count(query_id) %>%
  arrange(desc(n)) %>%
  filter(n > 1)
```

```
## # A tibble: 8 x 2
##   query_id    n
##   <int> <int>
## 1   17767     6
## 2   13551     5
## 3      1     4
## 4     27     3
## 5   26566     3
## 6   50265     3
## 7   28826     2
## 8   45810     2
```

- Isso apresenta que um mesmo query_id esteve no top 10 em “n” trimestres diferentes, o que levanta a possibilidade de que atacantes diferentes possam estar utilizando a mesma ferramenta para realizar ataques

Dados agrupados por qname, query_id, período e qtype

```
period_query_id_qname = data.frame()
for (i in c(1:length(data_split_year_period))) {
  query_id_qname_frequency = data_split_year_period[[i]] %>%
    count(qname, qtype, query_id, year_period, sort = TRUE) %>%
    filter(n > 1)

  period_query_id_qname = rbind(period_query_id_qname, head(query_id_qname_frequency, N) )
}
```

Os 10 query_id, qname, qtype mais utilizados divididos por período e ordenados pela frequência em que apareceram no período

```
period_query_id_qname %>%
  group_split(year_period)
```

```
## <list_of<
##   tbl_df<
##     qname      : character
##     qtype      : character
##     query_id   : integer
##     year_period: integer
##     n          : integer
##   >
## >[6]>
## [[1]]
## # A tibble: 10 x 5
##   qname    qtype query_id year_period    n
##   <chr>   <chr>   <int>      <int> <int>
## 1 isc.org. ANY      17767      20204  1141
## 2 irs.gov. ANY      16049      20204  1060
## 3 irs.gov. ANY      56064      20204  1049
## 4 irs.gov. ANY      63710      20204   782
```

```
## 5 sl. ANY 17767 20204 764
## 6 irs.gov. ANY 59797 20204 741
## 7 irs.gov. ANY 63374 20204 728
## 8 irs.gov. ANY 59378 20204 721
## 9 irs.gov. ANY 31694 20204 718
## 10 irs.gov. ANY 13304 20204 713
```

```
##
```

```
## [[2]]
```

```
## # A tibble: 10 x 5
```

```
##   qname      qtype query_id year_period    n
##   <chr>      <chr>   <int>      <int> <int>
## 1 isc.org.    ANY      17767      20211 56753
## 2 sl.         ANY      17767      20211 10260
## 3 .           ANY      28940      20211  417
## 4 VERSION.BIND. TXT      13551      20211  314
## 5 .           ANY      19592      20211  273
## 6 .           ANY      57166      20211  196
## 7 .           ANY      43855      20211  164
## 8 fe18.ru.    ANY      56643      20211  124
## 9 fe18.ru.    ANY      56686      20211  122
## 10 .          ANY      10000      20211  117
```

```
##
```

```
## [[3]]
```

```
## # A tibble: 10 x 5
```

```
##   qname      qtype query_id year_period    n
##   <chr>      <chr>   <int>      <int> <int>
## 1 peacecorps.gov. ANY      26566      20212  5090
## 2 sl.         ANY      17767      20212  3739
## 3 VERSION.BIND. TXT      13551      20212  346
## 4 213.1.168.192.in-addr.arpa. PTR      37845      20212   80
## 5 com.        ANY       1525      20212   46
## 6 67b.org.    AAAA     40074      20212   38
## 7 hcc.nl.     ANY         3      20212   33
## 8 version.bind. TXT         6      20212   30
## 9 pizzaseo.com. RRSIG         1      20212   29
## 10 200-19-107-238.measurebr.xiaofengtest.com. A      50265      20212   24
```

```
##
```

```
## [[4]]
```

```
## # A tibble: 10 x 5
```

```
##   qname      qtype query_id year_period    n
##   <chr>      <chr>   <int>      <int> <int>
## 1 pizzaseo.com. RRSIG         1      20213  6236
## 2 sl.         ANY      17767      20213  5764
## 3 VERSION.BIND. TXT      13551      20213   783
## 4 pizzaseo.com. ANY         27      20213   403
## 5 .           ANY      59252      20213   252
## 6 .           ANY      60765      20213   219
## 7 .           ANY      13143      20213   212
## 8 .           ANY      53342      20213   155
## 9 .           ANY      65372      20213   102
## 10 .          ANY      14262      20213   100
```

```
##
```

```
## [[5]]
```

```
## # A tibble: 10 x 5
```

```
##      qname          qtype query_id year_period      n
##      <chr>          <chr>    <int>    <int> <int>
## 1 peacecorps.gov. ANY      26566      20214 29963
## 2 sl.              ANY      17767      20214 8433
## 3 VERSION.BIND.    TXT      13551      20214 564
## 4 pizzaseo.com.    ANY       27      20214 101
## 5 pizzaseo.com.    RRSIG     1      20214 97
## 6 .               ANY     36609      20214 69
## 7 ip.parrotdns.com. A       28826      20214 68
## 8 .               ANY     47132      20214 58
## 9 .               ANY     3803      20214 57
## 10 .              ANY     20986      20214 47
##
## [[6]]
## # A tibble: 10 x 5
##      qname          qtype query_id year_period      n
##      <chr>          <chr>    <int>    <int> <int>
## 1 peacecorps.gov. ANY      26566      20221 5876
## 2 sl.              ANY      17767      20221 841
## 3 VERSION.BIND.    TXT      13551      20221 122
## 4 ip.parrotdns.com. A       28826      20221 41
## 5 dnsscan.shadowserver.org. A     64206      20221 33
## 6 version.bind.    TXT      64206      20221 30
## 7 pizzaseo.com.    ANY       27      20221 29
## 8 isc.org.          ANY     17767      20221 24
## 9 pizzaseo.com.    RRSIG     1      20221 21
## 10 clients1.google.com. A     14602      20221 19
```

- Ao observar esses registros, é possível verificar que alguns deles se repetem durante o tempo utilizando o mesmo qname, qtype e query_id, e que possivelmente não fizeram nenhuma alteração na ferramenta de ataque durante o período observado que iniciou no ultimo trimestre de 2020 até o primeiro trimestre de 2022.

Os query_id que apareceram com maior frequência entre os top 10 em todos os períodos, agrupados por qname e qtype

- Caso o query_id 13213 de qname = “isc.org.” e qtype = “ANY” fosse top 1 em 20204 e top 3 em 20211 e não aparecer em mais nenhum outro período seu “n” seria 2

```
top_queryid_qname = period_query_id_qname %>%
  count(query_id, qtype, qname) %>%
  arrange(desc(n)) %>%
  filter(n > 1)

top_queryid_qname
```

```
## # A tibble: 7 x 4
##      query_id qtype qname      n
##      <int> <chr> <chr>    <int>
## 1    17767 ANY    sl.        6
## 2    13551 TXT    VERSION.BIND. 5
## 3         1 RRSIG pizzaseo.com. 4
```



```
## 4      27 ANY    pizzaseo.com.      3
## 5     17767 ANY   isc.org.          3
## 6     26566 ANY   peacecorps.gov.      3
## 7     28826 A     ip.parrotdns.com.  2
```

- O ataque do tipo ANY de qname “sl.” e query_id “17767” apareceu no top 10 6x, ou seja em todo trimestre esse foi um dos ataques mais realizados que passaram pelos honeypots

Top 10 consultas que receberam a maior quantidade de requisições por períodos

```
dns_data.year_period.ungrouped <- group_split(dns_data_fetched, year_period)

dns_data.topNconsultas <- head(dns_data.year_period.ungrouped[[1]], N)
dns_data.year_period.ungrouped.len = length(dns_data.year_period.ungrouped)

dns_columns = c('year_period', 'qtype', 'quantity', 'qname')
select(dns_data.topNconsultas, dns_columns)
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use 'all_of(dns_columns)' instead of 'dns_columns' to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
## # A tibble: 10 x 4
##   year_period qtype quantity qname
##   <int> <chr>    <int> <chr>
## 1     20204 ANY     19005578 peacecorps.gov.
## 2     20204 ANY       816242 lavrov.in.
## 3     20204 ANY       779892 sl.
## 4     20204 ANY       652325 irs.gov.
## 5     20204 ANY       569411 fe18.ru.
## 6     20204 ANY        12296 .
## 7     20204 ANY       10248 isc.org.
## 8     20204 A         8467 20200328132334-cq9bm.1dd.sohu.com.
## 9     20204 RRSIG      6176 jp.
## 10    20204 A         4953 500940734da64dde863b257c9c12c03d.apigw.ap-southea~
```

```
select(head(dns_data.year_period.ungrouped[[2]], N), dns_columns)
```

```
## # A tibble: 10 x 4
##   year_period qtype quantity qname
##   <int> <chr>    <int> <chr>
## 1     20211 ANY    32698124 peacecorps.gov.
## 2     20211 ANY    3032399 sl.
## 3     20211 ANY    2418859 isc.org.
## 4     20211 ANY     941083 fe18.ru.
## 5     20211 ANY     463904 wzb.eu.
## 6     20211 ANY     132970 .
## 7     20211 A       20998 mirrorlist.centos.org.
## 8     20211 A       10698 hotspot.accesscam.org.
## 9     20211 MX        8014 pwad.gov.ae.
## 10    20211 A       3882 theguardian.webredirect.org.
```

```
select(head(dns_data.year_period.ungrouped[[3]], N), dns_columns)
```

```
## # A tibble: 10 x 4
##   year_period qtype quantity qname
##   <int> <chr>   <int> <chr>
## 1     20212 ANY    13183512 peacecorps.gov.
## 2     20212 ANY    1337802 sl.
## 3     20212 ANY    534815 irs.gov.
## 4     20212 ANY    220674 isc.org.
## 5     20212 ANY    124579 fe18.ru.
## 6     20212 ANY     90999 .
## 7     20212 MX     21895 dpc.ae.
## 8     20212 ANY    11229 hcc.nl.
## 9     20212 A      10965 dji.gov.ae.
## 10    20212 A       9144 emaratalyoum.com.
```

```
select(head(dns_data.year_period.ungrouped[[4]], N), dns_columns)
```

```
## # A tibble: 10 x 4
##   year_period qtype quantity qname
##   <int> <chr>   <int> <chr>
## 1     20213 RRSIG    324789 pizzaseo.com.
## 2     20213 ANY    178363 sl.
## 3     20213 ANY    165932 .
## 4     20213 A       5925 www.ac.my.blastodermic-swimmable.info.
## 5     20213 A       5291 tmall.com.
## 6     20213 A       4848 www.ac.my.superability-kooka.info.
## 7     20213 A       4655 2015annualreport.bloomberg.org.
## 8     20213 A       2794 lpnkuearwljpwqbwz.tmall.com.
## 9     20213 MX      1915 rt.com.
## 10    20213 MX      1888 nawahprogram.ae.
```

```
select(head(dns_data.year_period.ungrouped[[5]], N), dns_columns)
```

```
## # A tibble: 10 x 4
##   year_period qtype quantity qname
##   <int> <chr>   <int> <chr>
## 1     20214 ANY    4844082 peacecorps.gov.
## 2     20214 ANY    620249 sl.
## 3     20214 A      19541 www.ac.my.blastodermic-swimmable.info.
## 4     20214 A      17848 www.ac.my.superability-kooka.info.
## 5     20214 A      13595 www.ndnslab.com.
## 6     20214 ANY    11073 .
## 7     20214 RRSIG    8499 pizzaseo.com.
## 8     20214 MX     6670 nih.gov.
## 9     20214 A      5932 2015annualreport.bloomberg.org.
## 10    20214 MX     4680 nawahprogram.ae.
```

```
select(head(dns_data.year_period.ungrouped[[6]], N), dns_columns)
```

```
## # A tibble: 10 x 4
```

```
##      year_period qtype quantity qname
##      <int> <chr>    <int> <chr>
##  1      20221 ANY      2614699 peacecorps.gov.
##  2      20221 A        21200 admin.asry.net.
##  3      20221 ANY      19737 sl.
##  4      20221 A        18629 www.ndnslab.com.
##  5      20221 A        11635 ftp.ebisb.com.
##  6      20221 MX       7821 bankfab.com.
##  7      20221 A        6091 vpn.qatarsteel.com.qa.
##  8      20221 MX       6025 zayed.org.ae.
##  9      20221 A        5766 moi.gov.kw.
## 10      20221 MX       5077 mopa.ae.
```

```
for (i in c(2: dns_data.year_period.ungrouped.len)) {
  dns_data.topNconsultas <- rbind(dns_data.topNconsultas, head(dns_data.year_period.ungrouped[[i]], N))
}
```

```
## ----- Quantos ataques com cada tipo de qtype foi utilizado, por trimestre ? -----
#dns_data_fetched

dns_data_fetched.quarter_type_quantity = select(dns_data_fetched, c('year_period', 'qtype', 'quantity'))

dns_data_fetched.sum_attacks_quarterly = dns_data_fetched.quarter_type_quantity %>%
  group_by(qtype, year_period) %>%
  summarise(quantity = sum(quantity))
```

'summarise()' has grouped output by 'qtype'. You can override using the
'.groups' argument.

```
#dns_data_fetched.sum_attacks_quarterly %>%
# mutate(year_period=as.factor(year_period)) %>%
# ggplot(aes(x = year_period, y = quantity, color = qtype)) +
# geom_line()

#ggplot(data = dns_data_fetched.sum_attacks_quarterly, aes(x = year_period, y = quantity)) +
#   geom_line() +
#   facet_wrap(facets = vars(qtype))

#dns_data_fetched.sum_attacks_quarterly %>%
# filter(qtype != "ANY") %>%
# ggplot(aes(x = year_period, y = quantity)) +
#   geom_line() +
#   facet_wrap(facets = vars(qtype))

# ----- quantity with percentage

dns_data_fetched.sum_attacks_quarterly.sum_period_quantity = dns_data_fetched.sum_attacks_quarterly %>%
  group_by(year_period) %>%
  summarise(sum_period_quantity = sum(quantity), qtype=qtype, quantity=quantity)
```

'summarise()' has grouped output by 'year_period'. You can override using the
'.groups' argument.

```
dns_data_fetched.sum_attacks_quarterly.sum_period_quantity['quantity_percentage'] = (dns_data_fetched.s
```

```
#dns_data_fetched.sum_attacks_quarterly.sum_period_quantity %>%
# filter(quantity_percentage > 0.001) %>%
# filter(quantity_percentage > 0.1) %>%
#ggplot(aes(x = year_period, y = quantity_percentage)) +
# geom_line() +
#facet_wrap(facets = vars(qtype))

#dns_data_fetched.sum_attacks_quarterly.sum_period_quantity %>%
# filter(qtype != "ANY") %>%
# ggplot(aes(x = year_period, y = quantity_percentage)) +
#   geom_line() +
#   facet_wrap(facets = vars(qtype))

#dns_data_fetched.sum_attacks_quarterly.sum_period_quantity %>%
#mutate(year_period=as.factor(year_period)) %>%
# filter(quantity_percentage > 0.1) %>%
# ggplot(aes(x = year_period, y = quantity_percentage, color = qtype)) +
# geom_line()

# ----- filter any
```

```
dns_data_fetched.sum_attacks_quarterly.sum_period_quantity.filter_any = dns_data_fetched.sum_attacks_qu
  group_by(year_period) %>%
  filter(qtype != "ANY") %>%
  summarise(sum_period_quantity = sum(quantity), qtype=qtype, quantity=quantity)
```

'summarise()' has grouped output by 'year_period'. You can override using the
'.groups' argument.

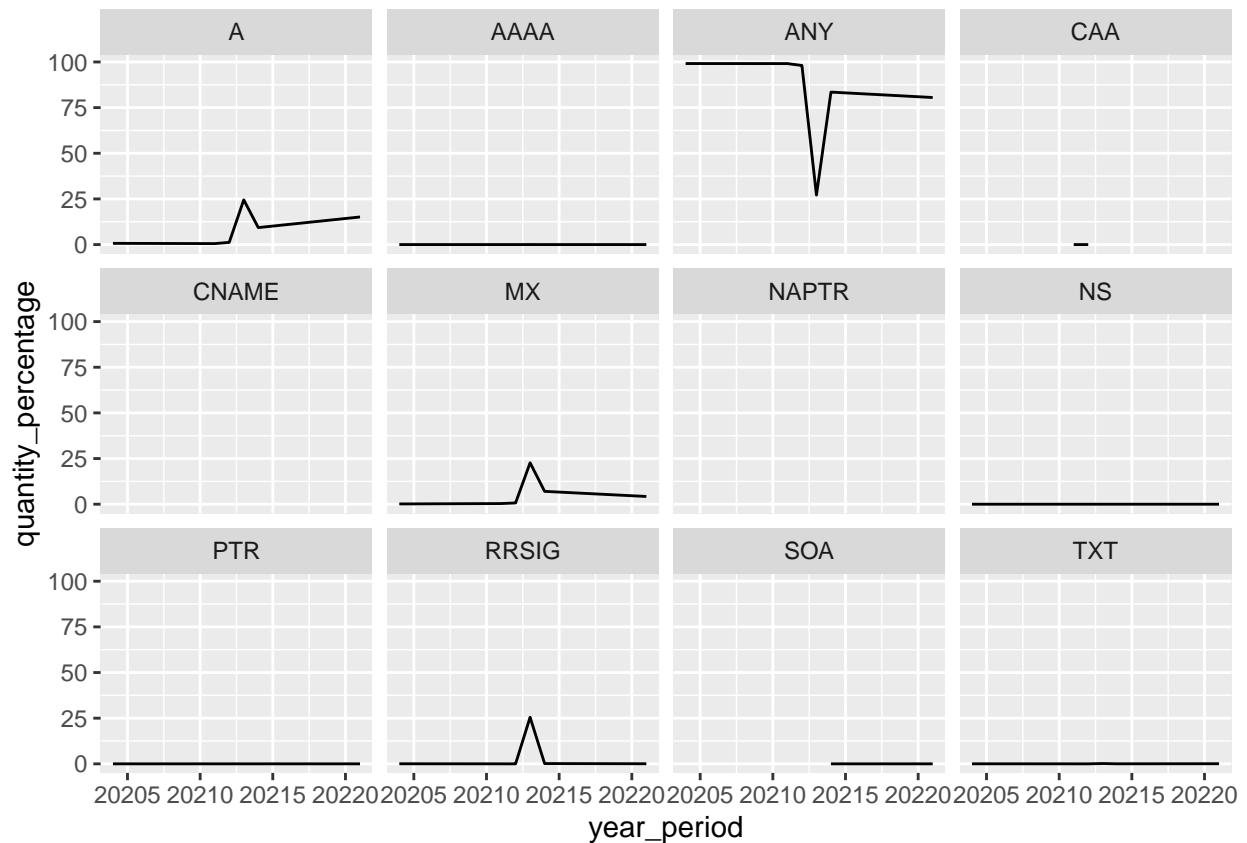
```
dns_data_fetched.sum_attacks_quarterly.sum_period_quantity.filter_any['quantity_percentage'] = (dns_data

#dns_data_fetched.sum_attacks_quarterly.sum_period_quantity.filter_any %>%
# ggplot(aes(x = year_period, y = quantity_percentage)) +
#   geom_line() +
#   facet_wrap(facets = vars(qtype))
```

- A quantidade de requests em % por trimestre por qtype

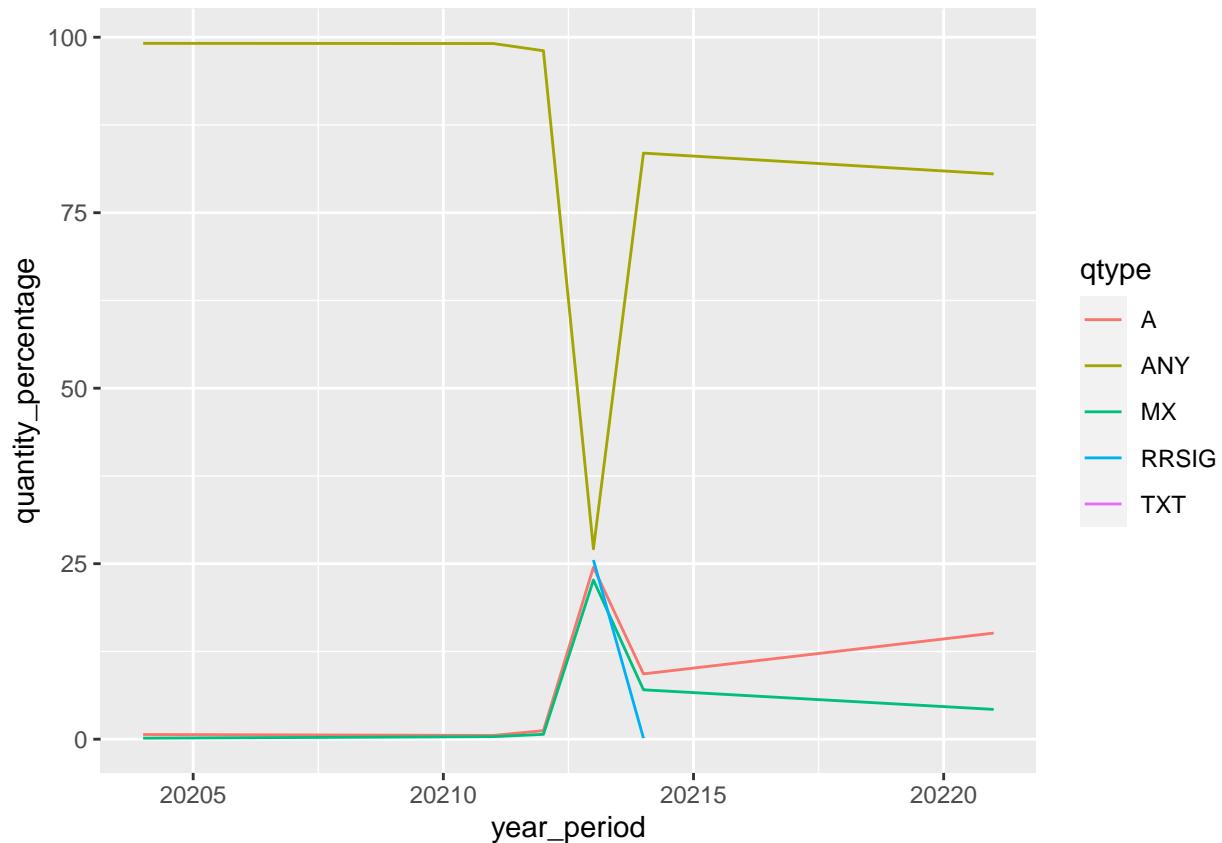
```
dns_data_fetched.sum_attacks_quarterly.sum_period_quantity %>%
ggplot(aes(x = year_period, y = quantity_percentage)) +
  geom_line() +
  facet_wrap(facets = vars(qtype))
```

geom_path: Each group consists of only one observation. Do you need to adjust
the group aesthetic?
geom_path: Each group consists of only one observation. Do you need to adjust
the group aesthetic?



- A quantidade de requests em % por trimestre por qtype
 - lembrando que em 20213 teve um problema em armazenar os dados, por isso talvez essa discrepância

```
dns_data_fetched.sum_attacks_quarterly.sum_period_quantity %>%
  #mutate(year_period=as.factor(year_period)) %>%
  filter(quantity_percentage > 0.1) %>%
  ggplot(aes(x = year_period, y = quantity_percentage, color = qtype)) +
  geom_line()
```



- Com esse gráfico é possível observar a grande quantidade de ataques do tipo ANY sendo extremamente superior aos demais, até mesmo no trimestre em que houve problema de armazenamento e alguns registros foram perdidos

- Quantos qnames e qtypes novos aparecem em cada trimestre

```
# ----- Quantos qtypes novos aparecem em cada trimestre -----
# > Diferenças percentuais são mais relevantes que absolutas

quarter_qtype_aux = dns_data.year_period.ungrouped[[1]] %>%
  group_by(qtype) %>%
  summarise(quantity = sum(quantity))

#quarter_qtype_2 = dns_data.year_period.ungrouped[[2]] %>%
# group_by(qtype) %>%
# summarise(quantity = sum(quantity))

#quarter_qtype_2
#merged = merge(x = quarter_qtype_aux, y = quarter_qtype_2, by = "qtype", all = TRUE)
#merged.new_quantity = merged$quantity.x - merged$quantity.y
#merged

quarter_new_qtype = data.frame()
for (i in c(2:dns_data.year_period.ungrouped.len)) {
  quarter_qtype = dns_data.year_period.ungrouped[[i]] %>%
    group_by(qtype) %>%
```

```

    summarise(quantity = sum(quantity))

merged = merge(x = quarter_qtype_aux, y = quarter_qtype, by = "qtype", all = TRUE)
merged.new_quantity = merged$quantity.x - merged$quantity.y

perio_to_period = paste(head(dns_data.year_period.ungrouped[[i - 1]]['year'], 1), '.', head(dns_data
quarter_new_qtype <- rbind(quarter_new_qtype, data.frame(quarter_to_quarter=perio_to_period, merged$q

quarter_qtype_aux = quarter_qtype
}

#quarter_new_qtype
#head(na.omit(quarter_new_qtype[order(-quarter_new_qtype$quantity_percentage),]))

# ----- Quantos qname novos aparecem em cada trimestre -----

quarter_qname_aux = dns_data.year_period.ungrouped[[1]] %>%
  group_by(qname) %>%
  summarise(quantity = sum(quantity))

quarter_new_qname = data.frame()
for (i in c(2:dns_data.year_period.ungrouped.len)) {
  quarter_qname = dns_data.year_period.ungrouped[[i]] %>%
    group_by(qname) %>%
    summarise(quantity = sum(quantity))

merged = merge(x = quarter_qname_aux, y = quarter_qname, by = "qname", all = TRUE)
merged.new_quantity = merged$quantity.x - merged$quantity.y

period_to_period = paste(head(dns_data.year_period.ungrouped[[i - 1]]['year'], 1), '.', head(dns_data
quarter_new_qname <- rbind(quarter_new_qname, data.frame(quarter_to_quarter=period_to_period, merged$q

quarter_qname_aux = quarter_qname
}

#quarter_new_qname
#head(na.omit(quarter_new_qname[-order(quarter_new_qname$quantity_percentage_diff),]))

```

- Top 10 novos qtypes por trimestre

```

quarter_new_qtype %>%
  arrange(desc(sum_quantity)) %>%
  select('quarter_to_quarter', 'merged.qtype', 'sum_quantity') %>%
  head(N)

```

```

##      quarter_to_quarter merged.qtype sum_quantity
## 1  2020 . 4 -> 2021 . 1          ANY    17841217
## 2  2021 . 3 -> 2021 . 4          ANY     5133467
## 3  2021 . 2 -> 2021 . 3        RRSIG     325120
## 4  2021 . 3 -> 2021 . 4           A      297381
## 5  2021 . 2 -> 2021 . 3           MX     180707
## 6  2021 . 3 -> 2021 . 4           MX     171803
## 7  2021 . 2 -> 2021 . 3           A      121656
## 8  2020 . 4 -> 2021 . 1           MX     111066

```

```
## 9 2020 . 4 -> 2021 . 1      A      67349
## 10 2021 . 4 -> 2022 . 1     TXT      962
```

- Nessa tabela deveríamos desconsiderar todos os registros relacionados ao período 2021.3, então o registro mais relevante é o top 1 que indica que tiveram 17841217 novas requisições do tipo ANY do ultimo trimestre de 2020 para o primeiro trimestre de 2021
- Top 10 novos qnames por trimestre

```
quarter_new_qname %>%
  arrange(desc(sum_quantity)) %>%
  select('quarter_to_quarter', 'merged.qname', 'sum_quantity') %>%
  head(N)
```

```
##      quarter_to_quarter merged.qname sum_quantity
## 1 2020 . 4 -> 2021 . 1 peacecorps.gov. 13689398
## 2 2021 . 3 -> 2021 . 4 peacecorps.gov. 4844048
## 3 2020 . 4 -> 2021 . 1      isc.org. 2408612
## 4 2020 . 4 -> 2021 . 1      sl. 2252507
## 5 2021 . 1 -> 2021 . 2      irs.gov. 534808
## 6 2021 . 3 -> 2021 . 4      sl. 441886
## 7 2020 . 4 -> 2021 . 1      fe18.ru. 371672
## 8 2021 . 2 -> 2021 . 3 pizzaseo.com. 323981
## 9 2020 . 4 -> 2021 . 1      . 120727
## 10 2021 . 2 -> 2021 . 3      . 76494
```

-
- Gráfico de barras da porcentagem de qtypes por trimestre

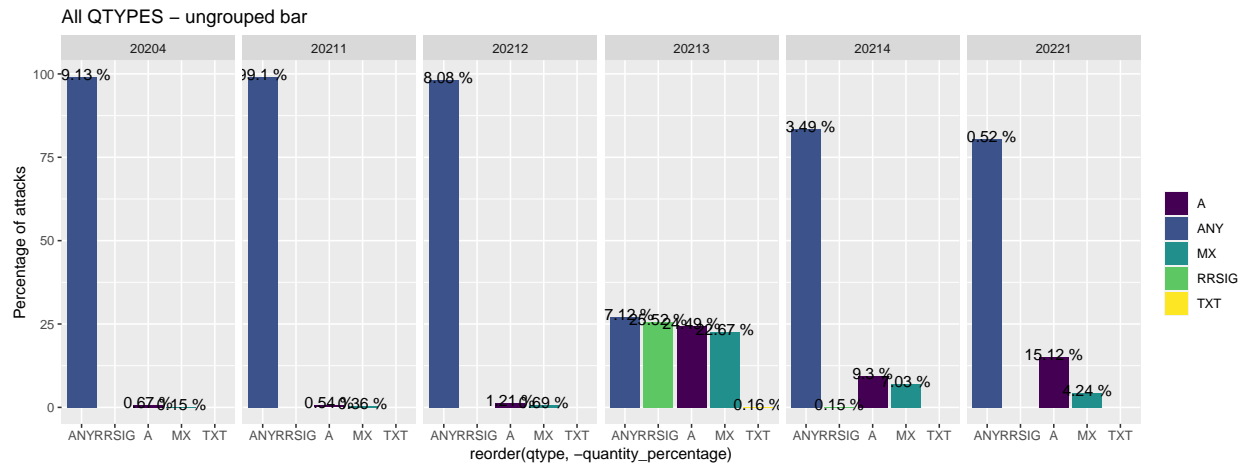
```
dns_data_fetched.sum_attacks_quarterly.sum_period = dns_data_fetched.sum_attacks_quarterly %>%
  group_by(year_period) %>%
  summarise(period_quantity = sum(quantity), qtype=qtype, quantity=quantity)
```

```
## 'summarise()' has grouped output by 'year_period'. You can override using the
## '.groups' argument.
```

```
dns_data_fetched.sum_attacks_quarterly.sum_period['quantity_percentage'] = (dns_data_fetched.sum_attacks_quarterly.sum_period %>%
  mutate(quantity_percentage = period_quantity / sum(quantity)) %>%
  filter(quantity_percentage > 0.1) %>%
  ggplot(aes(x=reorder(qtype, -quantity_percentage), y=quantity_percentage, fill=qtype)) +
  geom_bar(stat="identity", position="dodge") +
  scale_fill_viridis(discrete=TRUE, name="") +
  geom_text(aes(label = paste(round(quantity_percentage, 2), "%")), vjust = +0.25, ) +
  facet_grid(~year_period) +
  ylab("Percentage of attacks") +
  ggtitle("All QTYPES - ungrouped bar")
```

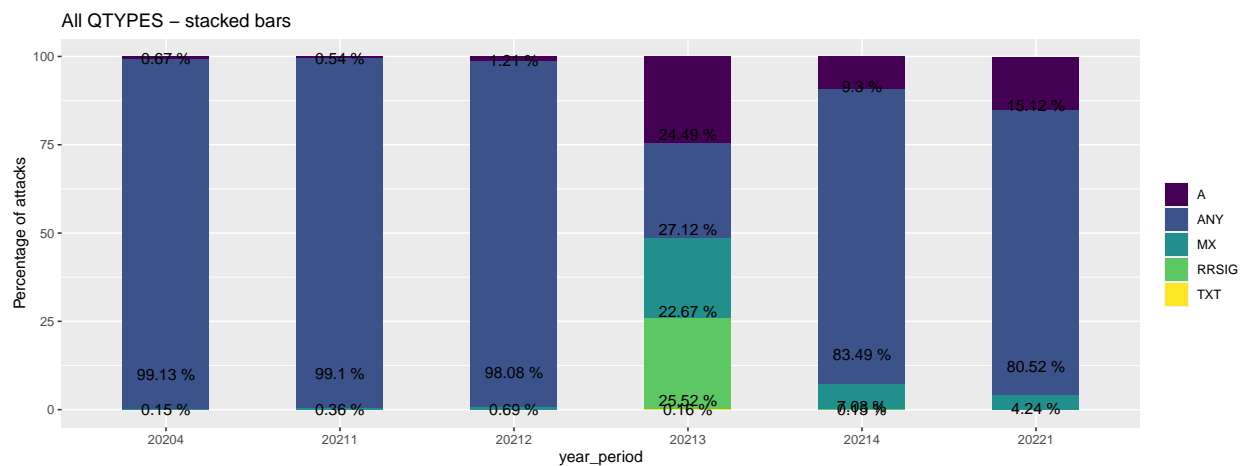
- A porcentagem calculada pela quantidade de requisições em cada período por cada qtype

```
dns_data_fetched.sum_attacks_quarterly.sum_period %>%
  mutate(year_period=as.factor(year_period)) %>%
  filter(quantity_percentage > 0.1) %>%
  ggplot(aes(x=reorder(qtype, -quantity_percentage), y=quantity_percentage, fill=qtype)) +
  geom_bar(stat="identity", position="dodge") +
  scale_fill_viridis(discrete=TRUE, name="") +
  geom_text(aes(label = paste(round(quantity_percentage, 2), "%")), vjust = +0.25, ) +
  facet_grid(~year_period) +
  ylab("Percentage of attacks") +
  ggtitle("All QTYPES - ungrouped bar")
```

- Novamente a porcentagem calculada pela quantidade de requisições em cada período por cada qtype, cada barra é um trimestre

```
dns_data_fetched.sum_attacks_quarterly.sum_period %>%
  mutate(year_period=as.factor(year_period)) %>%
  filter(quantity_percentage > 0.1) %>%
  ggplot( aes(x=year_period, y=quantity_percentage, fill=qtype)) +
    geom_bar(stat="identity", width = 0.5) +
    geom_text(aes(label = paste(round(quantity_percentage, 2), "%")), position = position_stack(vjust =
    scale_fill_viridis(discrete=TRUE, name="") +
    ylab("Percentage of attacks") +
    ggtitle("All QTYPES - stacked bars")
```

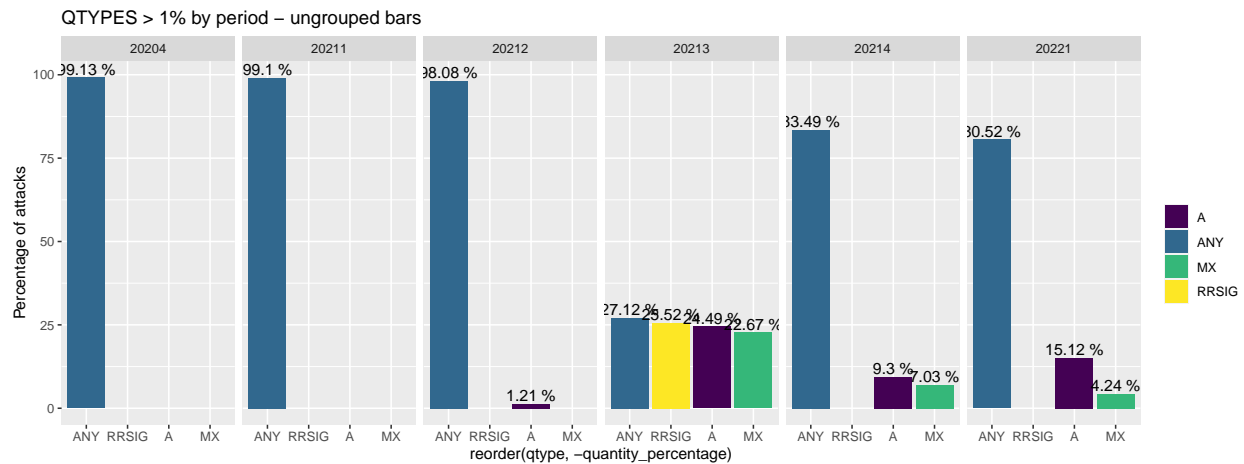


- Porcentagem da quantidade de requisições por trimestre por qtype em que tenha no mínimo 1% de requisições totais realizadas

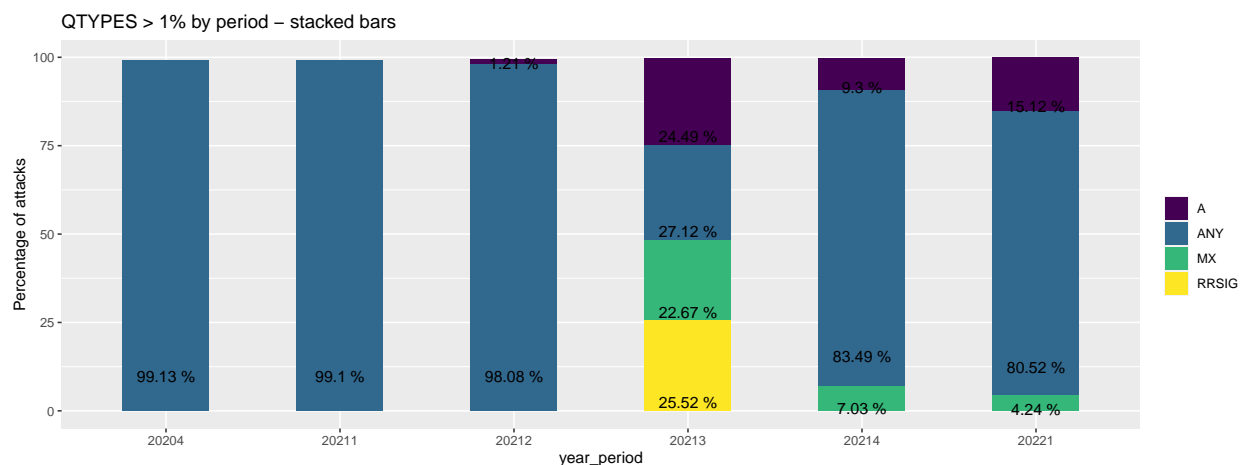
```
## Filter data using qtype quantity percentage bigger than 1

dns_data_fetched.sum_attacks_quarterly.sum_period %>%
  filter(quantity_percentage > 1) %>%
  mutate(year_period=as.factor(year_period)) %>%
  ggplot( aes(x=reorder(qtype, -quantity_percentage), y=quantity_percentage, fill=qtype)) +
    geom_bar(stat="identity", position="dodge") +
```

```
geom_text(aes(label = paste(round(quantity_percentage, 2), "%")), vjust = -0.25) +
facet_grid(~year_period) +
scale_fill_viridis(discrete=TRUE, name="") +
ylab("Percentage of attacks") +
ggtitle("QTYPES > 1% by period - ungrouped bars")
```



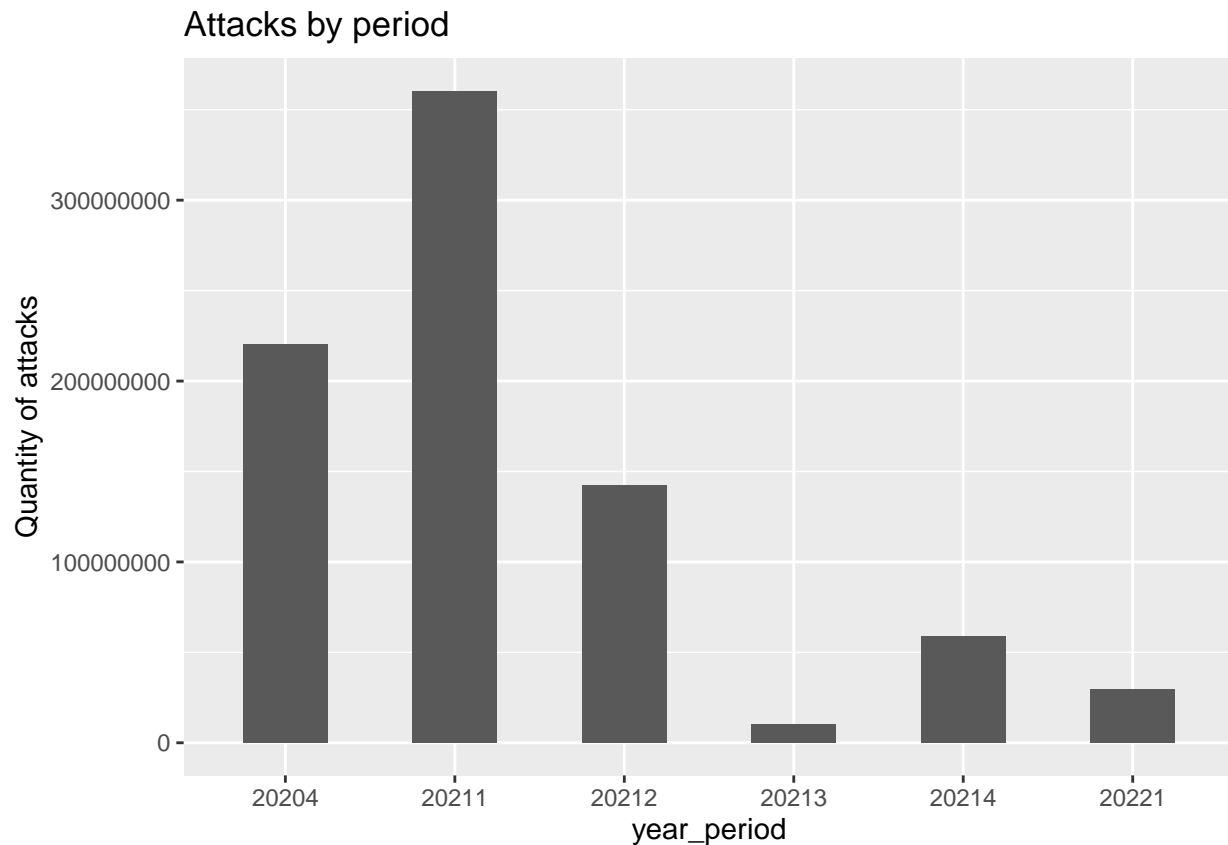
```
dns_data_fetched.sum_attacks_quarterly.sum_period %>%
  filter(quantity_percentage > 1) %>%
  mutate(year_period=as.factor(year_period)) %>%
  ggplot( aes(x=year_period, y=quantity_percentage, fill=qtype)) +
  geom_bar(stat="identity", width = 0.5) +
  geom_text(aes(label = paste(round(quantity_percentage, 2), "%")), position = position_stack(vjust =
scale_fill_viridis(discrete=TRUE, name="") +
ylab("Percentage of attacks") +
ggtitle("QTYPES > 1% by period - stacked bars")
```



- Com esse gráfico é possível observar a grande quantidade de ataques do tipo ANY sendo extremamente superior aos demais, levando a crer que a depreciação da query do tipo ANY vazia nos servidores DNS não afetou o modo em que os ataques são realizados, até mesmo porque inúmeros servidores DNS não se atualizaram e não depreciaram esse modo de realizar as queries

- Quantidade de requisições por trimestre

```
dns_data_fetched.sum_attacks_quarterly.sum_period %>%
  mutate(year_period=as.factor(year_period)) %>%
  ggplot( aes(x=year_period, y=period_quantity)) +
    geom_bar(stat="identity", width = 0.5) +
    scale_fill_viridis(discrete=TRUE, name="") +
    ylab("Quantity of attacks") +
    ggtitle("Attacks by period")
```



- Em quantidade de ataques, obtivemos um pico no primeiro trimestre de 2021, e logo vamos poder comparar com o primeiro trimestre de 2022 para verificar se geralmente o primeiro trimestre é aquele em que ocorrem a maior quantidade de ataques no ano

- Os 10 ataques em que tiveram uma maior sobreposição de tempo entre o início e o fim do ataque com um mesmo query_id

```
dns_data_overlap_fetched %>%
  arrange(desc(amount_overlap)) %>%
  head(5)
```

```
##      id tempo_inicio_datetime tempo_final_datetime query_id qtype  qname
## 1 248406 2021-02-03 11:13:01 2021-02-03 18:56:55 17767 ANY wzb.eu.
## 2 241613 2021-01-31 16:50:23 2021-01-31 22:34:50 17767 ANY sl.
## 3 243903 2021-02-01 19:25:25 2021-02-01 22:14:16 17767 ANY sl.
```

## 4	261733	2021-02-09 00:11:08	2021-02-09 05:53:30	17767	ANY	sl.
## 5	268519	2021-02-11 15:06:02	2021-02-11 17:28:15	17767	ANY	sl.
##	requests_per_attack	ip amount_overlap				
## 1	95308	78.128.114.125		658		
## 2	19754	73.95.243.160		527		
## 3	590	174.99.5.206		270		
## 4	1444	76.107.212.17		257		
## 5	1453	172.101.205.51		235		

- Isso possivelmente indica que alguns atacantes utilizaram-se da mesma ferramenta para realizar o ataque para diferentes domínios e nem sequer modificaram o query_id, ou utilizaram algum modo de incremento para ofuscar o ataque

Texto

Dados

- Esse documento teve como dados apenas os ataques que utilizam DNS. Esses ataques foram recolhidos pelo MP-H entre o período de 29/10/2020 até 24/02/2022.
- Os datasets com os ataques DNS foram processados por um script em python que faz o parse do payload para recolher informações do DNS como:
 - O query_id que é o TX ID da transação definido pelo atacante;
 - O qname que é o domínio em que o atacante realizou spoofing;
 - O qtype que é o tipo da query utilizada para realizar amplificação.

Questões

- A ideia para esse documento é realizar uma análise longitudinal da evolução dos payloads somente para evoluções relacionadas a ataques DNS.
- Uma das análises destacadas é o uso do DNS com o QTYPE ANY. Pois, historicamente ataques DRDoS com DNS utilizam majoritariamente requisições para o QTYPE ANY, que retorna todos os tipos de registros para um dado nome. Assim, será que a incidência de requisições com ANY evoluiu ao longo do tempo?
- Para contexto, requisições com o QTYPE ANY tem um grande potencial de amplificação e o seu uso legítimo é apenas para debug do servidor e para verificar o estado de um servidor DNS, por esse motivo, foi introduzida uma recomendação para que servidores DNS **reduzam ou limitem** o tamanho na resposta das consultas com QTYPE ANY (RFC8482) para evitar o uso dessas requisições em ataques de negação de serviço por reflexão ou tornar essa forma de amplificação menos útil aos atacantes.

Analises

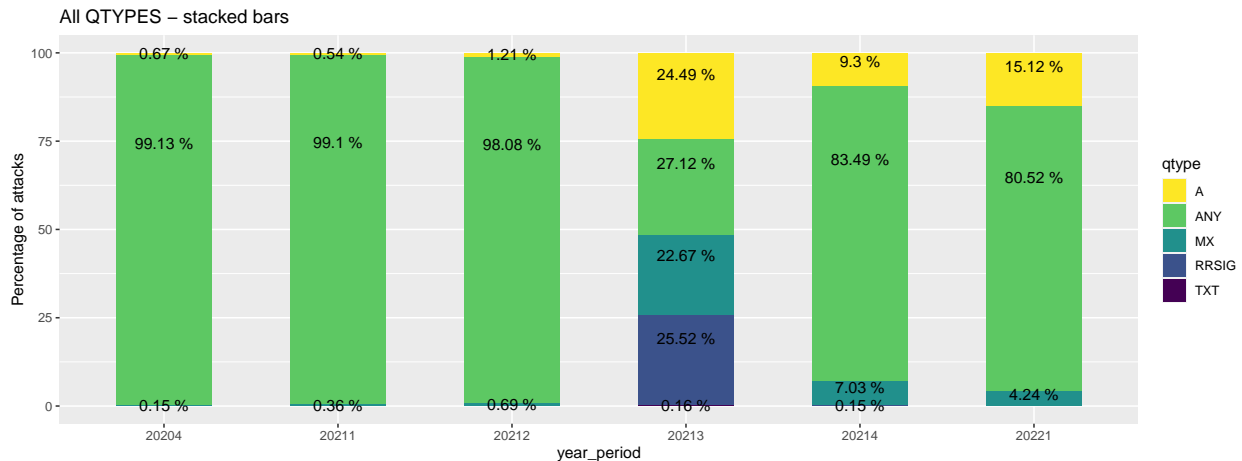
- Obs.: Tenha em mente que os dados referentes ao terceiro trimestre de 2021 podem não refletir a realidade, devido a alguns problemas de armazenamento durante esse período.

Quantidade de requisições por QTYPE e trimestres

- Para realizar a análise, foi realizado uma query agrupando os registros do DNS por (year_period, year, period, qname, qtype) e somado a quantidade total de requisições (request_per_attack) e renomeado para (quantity)

- Dessa forma, foi analisado a porcentagem de cada QTYPE dentre as consultas por trimestre de DNS já processados, em que a porcentagem foi definida pela soma total das requisições por ataques de um mesmo tipo por trimestre. Então por exemplo se em um mesmo trimestre a soma total das requisições dos ataques com o QTYPE MX for 20 e o total das requisições dos ataques do tipo ANY for 80, a porcentagem de ANY será 80% e de MX será 20% naquele trimestre.

```
dns_data_fetched.sum_attacks_quarterly.sum_period %>%
  mutate(year_period=as.factor(year_period)) %>%
  filter(quantity_percentage > 0.1) %>%
  ggplot(aes(x=year_period, y=quantity_percentage, fill=qtype)) +
    geom_bar(stat="identity", width = 0.55) +
    geom_text(aes(label = paste(round(quantity_percentage, 2), "%")), position = position_stack(vjust =
    scale_fill_viridis(discrete=TRUE, direction = -1) +
    ylab("Percentage of attacks") +
    ggtitle("All QTYPES - stacked bars")
```



- Esse gráfico apresenta que ANY é a maioria em todos os trimestre em que os dados foram coletados com uma predominância grande (98%+) nos três primeiros trimestres (2020/Q4, 2021/Q1, 2021/Q2) e teve uma queda nos dois últimos trimestres com 83.49% em 2021/Q4 e 80.52% 2022/Q1 liberando a entrada dos QTYPES A e MX que juntos somam 16.32% em 2021/Q4 e 19.36% em 2021/Q5.
- Valores abaixo de 1.5% no gráfico foram desconsiderados

Quantidade de ataques por query_id e sobreposição

- Para iniciar essa análise de dados foi verificado que inúmeros ataques utilizavam o mesmo query_id, como apresentado no top 5 abaixo que o query_id “17767” foi utilizado em 87.924 ataques diferentes somando 9.063.853 requisições utilizando o mesmo query_id entre cerca de 30 domínios diferentes e isso levantou algumas dúvidas, pois o query_id deveria ser aleatório conforme estabelecido na RFC5452 em que resolvers DNS precisam utilizar um número de query ID aleatório entre (0-65535), totalizando 2^{16} possíveis valores para o query_id. Dessa forma, uma dúvida que surge é porque tantos ataques utilizam o mesmo query_id?
 - Uma curiosidade é que o query_id mais utilizado “17767” em hexadecimal é 0x4567 que são 4 números consecutivos

```
data %>%
  group_by(query_id) %>%
  summarise(requests_per_attack = sum(requests_per_attack), n = n(), qnames_diff = n_distinct(qname)) %>%
  arrange(desc(n)) %>%
  head(5)
```

```
## # A tibble: 5 x 4
##   query_id requests_per_attack      n qnames_diff
##   <int>          <int> <int>      <int>
## 1    17767          9063853 87924         30
## 2    26566          9019358 40940          1
## 3         1          401246  7803        1311
## 4    13551           5961  2347          3
## 5    16049          17634  1062          3
```

- Uma teoria que vem a mente é que como muitos desses ataques utilizaram o mesmo query_id, talvez eles utilizem a mesma ferramenta para realizar esses ataques e nem se deram ao trabalho de alterar o query_id ou utilizar um número randômico para realizar os ataques...
- Um destaque relacionado ao query_id é que ele deve ser aleatório, mas pode repetir pois existem apenas 65.536 possibilidades de números distintos (de 0 a 65535) que é o tamanho de um cabeçalho de 16 bits. Então, a partir de 65.536 requisições com o mesmo qname o query_id probabilisticamente iniciaria a se repetir, o que torna anômalo os casos listados a baixo em que o query_id repetiu 58.001 vezes enquanto probabilisticamente deveria repetir aproximadamente 40x (requests_per_attack / 65.536)

```
possible_query_id = 65536
```

```
data %>%
  group_by(query_id, qname) %>%
  summarise(requests_per_attack = sum(requests_per_attack), query_id_repeated = n()) %>%
  mutate(probabilist_repetition = as.integer(requests_per_attack/possible_query_id)) %>%
  arrange(desc(query_id_repeated)) %>%
  head(5)
```

```
## 'summarise()' has grouped output by 'query_id'. You can override using the
## '.groups' argument.
```

```
## # A tibble: 5 x 5
## # Groups:   query_id [4]
##   query_id qname      requests_per_attack query_id_repeated probabilist_rep~
##   <int> <chr>          <int>          <int>          <int>
## 1    17767 isc.org.          2652120          58001           40
## 2    26566 peacecorps.gov.  9019358          40940          137
## 3    17767 sl.          5947165          29801           90
## 4         1 pizzaseo.com.  331233           6442            5
## 5    13551 VERSION.BIND.    5938            2335            0
```

- Essa diferença gigante entre a quantidade de vezes que o query_id repetiu e a probabilidade em que poderia repetir auxilia no embasamento da utilização de uma mesma ferramenta pelos atacantes.
- Uma outra teoria é que se ataques com o mesmo query_id acontecem em paralelo a outro ataque com qname diferente, possivelmente existe um atacante que comanda inúmeros diferentes ataques ao mesmo tempo ou colabora para a teoria de que vários atacantes utilizam a mesma ferramenta para realizar os ataques e esses ataques são sincronizados para acontecerem em um mesmo período.

- Para verificar se ataques com um mesmo qtype e query_id acontecem em paralelo a diferentes qnames é necessário verificar se esses ataques são concomitantes, ou seja, observar se um ataque acontece enquanto outro ataque com o mesmo query_id acontece. O cálculo da quantidade de ataques concomitantes foi verificado se o tempo início de um ataque está entre o início e fim de outro ataque ou o tempo final está entre o início e fim de outro ataque, é somado 1 ao contador (amount_overlap).

```
dns_data_overlap_fetched %>%
  arrange(desc(amount_overlap)) %>%
  select('query_id', 'qtype', 'qname', 'requests_per_attack', 'ip', 'amount_overlap') %>%
  head(5)
```

##	query_id	qtype	qname	requests_per_attack	ip	amount_overlap
## 1	17767	ANY	wzb.eu.	95308	78.128.114.125	658
## 2	17767	ANY	sl.	19754	73.95.243.160	527
## 3	17767	ANY	sl.	590	174.99.5.206	270
## 4	17767	ANY	sl.	1444	76.107.212.17	257
## 5	17767	ANY	sl.	1453	172.101.205.51	235

- Com a tabela acima é possível observar que existe muita sobreposição entre os ataques que utilizam o mesmo QTYPE e o mesmo query_id com o ataque para o domínio “wzb.eu.” que teve 658 ataques com o mesmo query_id “17767” e qtype “ANY” ocorrendo durante o período em que esse ataque ocorria.

Conclusões

- Enfim, os ataques com o QTYPE “ANY” ainda são os mais utilizados em todos os períodos, indicando que a redução no tamanho da resposta desse tipo de query não modificou ou não limitou o fator amplificação o suficiente para que os atacantes parem de abusar. Assim os atacantes continuam abusando dessa query na realização de seus ataques, muito provavelmente devido ao fato de servidores DNS não realizarem atualizações para se adequarem as recomendações apresentadas na RFC8482 para limitar o tamanho da resposta e assim tornar esse modo de amplificação menos útil aos atacantes.
- Além disso, o grande número de ataques utilizando o mesmo query_id e também a quantidade de sobreposição entre os ataques indicam que muitos deles utilizaram a mesma ferramenta para realizar os ataques. Ainda mais pois os atacantes nem sequer se deram ao trabalho de alterar o query_id ou utilizar um número randômico/incremental para o query_id ser diferente.