Contents

Load data set and packages

```
source('ens-init.R')
```

- We’ve loaded an articulated dataset generated by QuickBlocks. Included in the data set is any transaction or trace from May ’17 - Dec. ’17 that mentions the ENS-Registry `0x6090a6e47849629b7245dfa1ca21d94cd15878ef` contract address.
- Function arguments are “exploded” into columns.
- Articulated data for ENS-EthNameService `0x314159265dD8dbb310642f98f50C066173C1259b` is yet to be integrated.

I. ENS-Registry contract: Top 10 most active users (“Superusers”), May-Dec ’17

First, we analyze activity from the top 10 most active addresses ranked by by non-error ENS transactions during this ~8 month period. Reading some threads and also the original EIPs, it’s clear that name squatting is a priority concern for the ENS designers. From an ENS EIP:

> In order to maximize utility and adoption of a new namespace, the registrar should mitigate speculation and “name squatting”, however the best approach for mitigation is unclear. Thus an “initial” registrar is proposed, which implements a simple approach to name allocation. [. . .] This Initial Registrar contract will be replaced with a permanent registrar contract. The Permanent Registrar will increase the available namespace, and incorporate lessons learned from the performance of the Initial Registrar. – maurelian, EIP #162

For brevity’s sake, we’ll call each address a “user,” and use the name “superuser” to mean an address with contract interaction count in the top 10.

i. ENS-Registry contract: Superuser interaction count

Superuser vs. non-superuser contract interaction count

```
base.exploded %>%
  filter(to == special.addr$`Contract: ENS-Registrar`,
         is_error == 0,
         traceid == 0) %>%
  mutate(is.superuser = from%in% top.10) %>%
  group_by(is.superuser) %>%
  summarize(n = n()) %>%
  mutate(pct = n/sum(n))
```

```
## # A tibble: 2 x 3
##   is.superuser      n      pct
##          <lgl>  <int>    <dbl>
## 1        FALSE 824648 0.787007
## 2         TRUE 223180 0.212993
```

Superuser contract interaction detail

```r
base.exploded %>%
  filter(to == special.addr$`Contract: ENS-Registrar`,
         is_error == 0,
         traceid == 0,
         from %in% top.10) %>%
  group_by(from, fn.name) %>%
  summarize(n = n()) %>%
  spread(key = 'fn.name', val = 'n') %>%
    mutate_all(funs(ifelse(is.na(.), 0, .))) %>%
  mutate(auction.starts = startAuction + startAuctions + startAuctionsAndBid,
         bids = startAuctionsAndBid + newBid) %>%
  select(auction.starts, bids, unsealBid, finalizeAuction)
```

```
## # A tibble: 10 x 5
## # Groups:   from [10]
##                                           from auction.starts  bids
##                                          <chr>          <dbl> <dbl>
##  1 0x000fb8369677b3065de5821a86bc9551d5e5eab9           3887  3616
##  2 0x001e28376ebe0982a50b0ad4a076a39aa0264bcc          12661 12635
##  3 0x002acd20810b405fc4d01896871a6a7ba4b279fa           5483  5483
##  4 0x009fde04525832da85a240d68c82421ca249a5b8           5384  1779
##  5 0x00ab424d2019bc0f4c648232c6e7d181a34034b8           9387  9385
##  6 0x00bda1105ce38848b890c138d3d23a0435790a39          16614 14281
##  7 0x216fc0aa752393f8f215b603b4156987d3d8bbbf           2019  2057
##  8 0xa7f3659c53820346176f7e0e350780df304db179            571 26461
##  9 0xd2fa59b040852952bf4b4639edd4d8a718a4598a           2405  2513
## 10 0xf5f700e1912b93ad09597bfa22484e01c0035b04              0  5831
## # ... with 2 more variables: unsealBid <int>, finalizeAuction <dbl>
```

Interpretation:

Despite their relatively large activity numbers, some superuser addresses never called `finalizeAuction()` within the timeframe of this analysis. From the ENS docs:

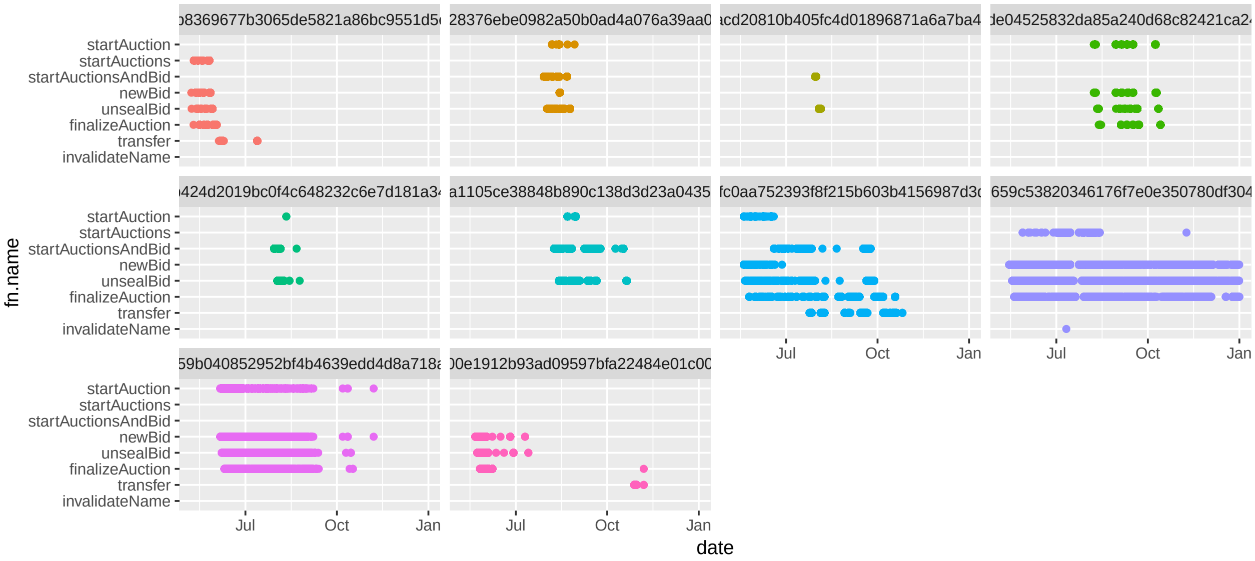
> Once the auction has completed, it must be finalized in order for the name to be assigned to the winning bidder. Only the winning bidder can do this.

If a user can afford to delay her/his `finalizeAuction()` calls for months, it reinforces a suspicion of name squatting.

ii. ENS-Registry contract: Superuser interaction timelines

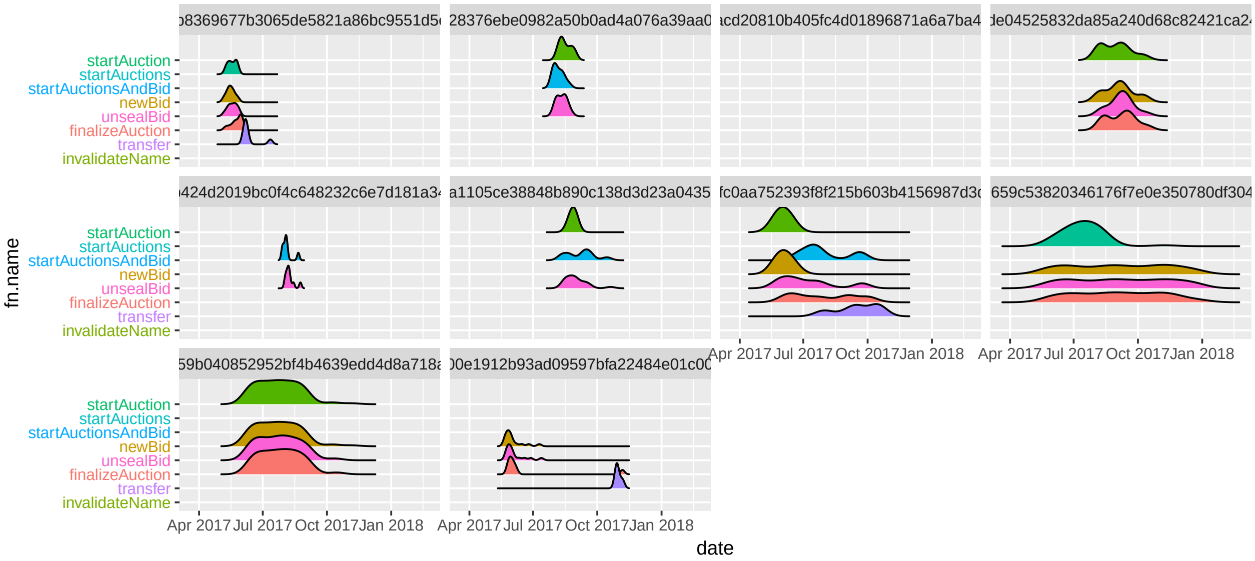
Simple interaction timeline per superuser

```r
base.exploded %>%
  filter(to == special.addr$`Contract: ENS-Registrar`,
         from %in% top.10,
         is_error == 0,
         traceid == 0) %>%
  ggplot(aes(x=date, y=fn.name, color=from)) +
  geom_point() +
  facet_wrap(facets = 'from') +
  scale_y_discrete(limits=rev(c('startAuction', 'startAuctions', 'startAuctionsAndBid', 'newBid', 'unsea
  theme(legend.position="none")
```



Activity timeline showing relative interaction density per superuser

```
base.exploded %>%
  filter(to == special.addr$`Contract: ENS-Registrar`,
         from %in% top.10,
         is_error == 0,
         traceid == 0) %>%
  mutate(date = as.Date(date)) %>%
  group_by(date, from, fn.name) %>%
  summarize(n = n()) %>%
  ggplot(aes(x=date, y=fn.name, fill=fn.name)) +
  geom_density_ridges() +
  facet_wrap(facets = 'from') +
  theme(axis.text.y = element_text(colour = custom)) +
  scale_y_discrete(limits=rev(c('startAuction', 'startAuctions', 'startAuctionsAndBid', 'newBid', 'unsea
  theme(legend.position="none")
```



II. ENS-Registry contract: Simple non-error auction action counts, all addresses, May-Dec '17

i. Non-error action counts per month

```
base.exploded %>%
  filter(to == special.addr$`Contract: ENS-Registrar`,
         is_error == 0,
         traceid == 0) %>%
  mutate(month = format(date, '%Y-%m')) %>%
  group_by(month, fn.name) %>%
  summarize(n = n()) %>%
  spread(key = 'fn.name', value = 'n') %>%
  mutate_each(funs(ifelse(is.na(.), 0, .))) %>%
  mutate(bids = newBid + startAuctionsAndBid,
         start.auction = startAuction + startAuctions + startAuctionsAndBid) %>%
  select(bids, start.auction, unsealBid, finalizeAuction)
```

```
## # A tibble: 8 x 5
## # Groups:   month [8]
##     month   bids start.auction unsealBid finalizeAuction
##     <chr>  <int>         <int>     <int>           <int>
## 1 2017-05  92866         77935     75343           38479
## 2 2017-06 104364        100624     96351           59931
## 3 2017-07  69184         60051     54593           46931
## 4 2017-08  49794         48269     61704           22264
## 5 2017-09  14034         15136     13800            9158
## 6 2017-10  14635         18940     11713           10445
## 7 2017-11  20266         23973     17831           13448
## 8 2017-12   6823          8234      6854            4205
```

ii. Median, mean, and standard deviation of # bids unsealed per auction

```
base.exploded %>%
  filter(to == special.addr$`Contract: ENS-Registrar`,
         is_error == 0,
         traceid == 0,
         fn.name == 'unsealBid') %>%
  mutate(month = format(date, '%Y-%m')) %>%
  group_by(month, args1) %>%
  summarize(n = n()) %>%
  summarize(median = median(n), mean = mean(n), sd = sd(n))
```

```
## # A tibble: 8 x 4
##     month median     mean        sd
##     <chr>  <dbl>    <dbl>     <dbl>
## 1 2017-05      1 1.604647 2.4514543
## 2 2017-06      1 1.407015 1.6701722
## 3 2017-07      1 1.075194 0.4514676
## 4 2017-08      1 1.020609 0.1503326
## 5 2017-09      1 1.057877 0.2509264
## 6 2017-10      1 1.063948 0.3006444
## 7 2017-11      1 1.051728 0.3162589
## 8 2017-12      1 1.040534 0.2134903
```

Interpretation:

Over this timeframe, there was initially more competition over names judging from the mean

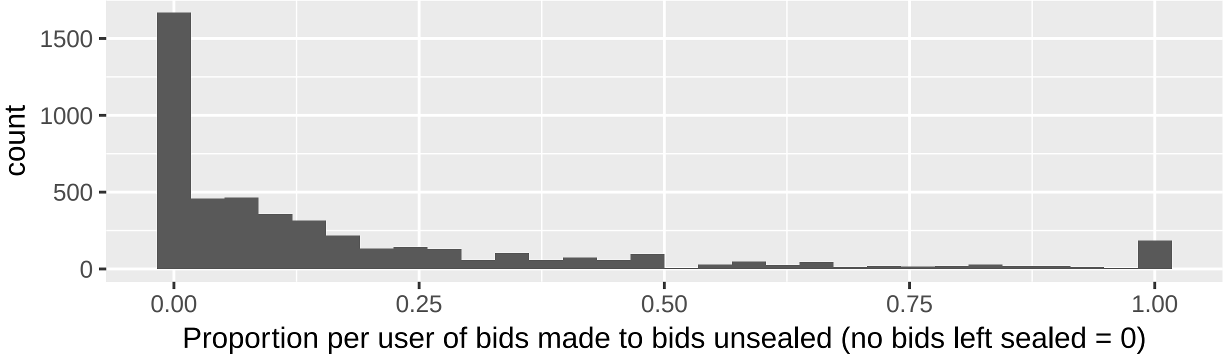
and variance

of # unsealed bids per auction. We expect that this can be explained by competitive early known name bidding. This is probably why Nick Johnson was motivated to separate out the two categories of names in his analyses. We can explore this by scraping known names from etherscan and comparing them to a group comprised of unknown namehashes (TBD).

iii. Proportion of bids made vs. bids unsealed: addresses with >5 bids

We read a lot about users who missed the reveal on their bids. What's the distribution of the proportion between bids placed and bids revealed during the analysis timeframe? We'll look at addresses who placed at least 5 bids.

```
  base.exploded %>%
    filter(to == special.addr$`Contract: ENS-Registrar`,
           is_error == 0,
           traceid == 0) %>%
    group_by(from, fn.name) %>%
    summarize(n = n()) %>%
    spread(key = c('fn.name'), value = c('n')) %>%
    mutate_all(funs(ifelse(is.na(.), 0, .))) %>%
    mutate(bid = newBid + startAuctionsAndBid,
           start.auction = startAuction + startAuctions + startAuctionsAndBid) %>%
    mutate(made.vs.unsealed = (bid - unsealBid) / bid) %>%
    select(bid, unsealBid, finalizeAuction, made.vs.unsealed) %>%
    filter(bid > 5) %>%
    ggplot(aes(x = made.vs.unsealed)) +
    geom_histogram() +
    xlab('Proportion per user of bids made to bids unsealed (no bids left sealed = 0)')
```



```
  base.exploded %>%
    filter(to == special.addr$`Contract: ENS-Registrar`,
           is_error == 0,
           traceid == 0) %>%
    group_by(from, fn.name) %>%
    summarize(n = n()) %>%
    spread(key = c('fn.name'), value = c('n')) %>%
    mutate_all(funs(ifelse(is.na(.), 0, .))) %>%
    mutate(bid = newBid + startAuctionsAndBid,
           start.auction = startAuction + startAuctions + startAuctionsAndBid) %>%
    mutate(made.vs.unsealed = (bid - unsealBid) / bid) %>%
    select(bid, unsealBid, finalizeAuction, made.vs.unsealed) %>%
    filter(bid > 5) %>%
    arrange(desc(made.vs.unsealed)) %>%
    mutate(made.vs.unsealed = format(made.vs.unsealed, digits=2, scientific=FALSE)) %>%
  rename('proportion' = made.vs.unsealed)
```

```
## # A tibble: 4,790 x 5
## # Groups:   from [4,790]
##                                           from   bid unsealBid
##                                          <chr> <dbl>     <dbl>
##  1 0x000aef77eced4faa38bb60558c63296cf42d7817     6         0
##  2 0x0011542688a27f9f5bd10a803659290a1480f6ca    11         0
##  3 0x00129669b171f73773d712ede3fb9afce2aff35b     7         0
##  4 0x003b3efe25cd21d99da84b66f1dff067f3632332    10         0
##  5 0x003f12ba2e37d864732ce8b000270b05fdb2a893    19         0
##  6 0x005e530ec4524d5068cbba21560b22860546cabe     8         0
##  7 0x00637e13e40ffe973bd286168751a9261a76788d     8         0
##  8 0x0074ca1979889653aaefd4b112b93b029e682288    11         0
##  9 0x007c6e5425e9af288a1760c04efe64beae153942    17         0
## 10 0x008d46a1df65b0b2117f85a4e07cf82740a82e01     9         0
## # ... with 4,780 more rows, and 2 more variables: finalizeAuction <dbl>,
## #   proportion <chr>
```

Interpretation:

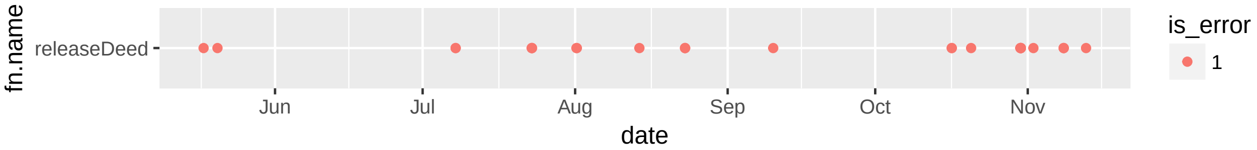
III. Other function calls and their error rates

i. `releaseDeed()`

Release deed was called 29 times, all of which were errors. We have an explanation: deeds can't be released until a year after their initial possession date.

Timeline of calls:

```r
base.exploded %>%
  filter(traceid == 0,
         fn.name == 'releaseDeed') %>%
  mutate(is_error = as.character(is_error)) %>%
  ggplot(aes(x=date, y=fn.name, color=is_error)) +
  geom_point()
```

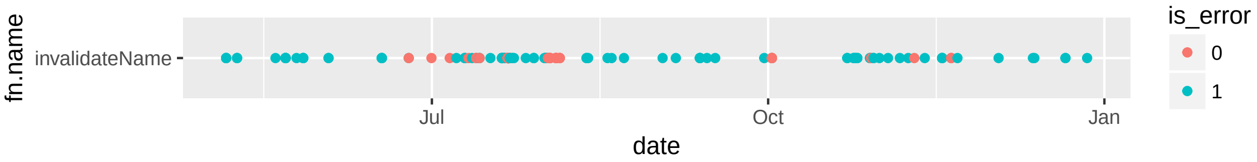


ii. `invalidateName()`

`invalidateName()` was called 387 times, 219 were successful and 168 were errors.

Timeline of calls:

```r
base.exploded %>%
  filter(traceid == 0,
         fn.name == 'invalidateName') %>%
  mutate(is_error = as.character(is_error)) %>%
  ggplot(aes(x=date, y=fn.name, color=is_error)) +
  geom_point()
```



About QuickBlocks

QuickBlocks is a collection of software libraries, applications, and command-line tools designed to give you quick access to the data provided by an Ethereum node.

In a manner very similar to the way web3.js works, QuickBlocks sits between a locally running Ethereum node (or any node, local or remote, for that matter), and delivers the Ethereum data to your application. There are two significant advantages to using QuickBlocks over web3.js First, QuickBlocks caches the data locally which means that time-to-data is severely decreased. Secondly, if you provide QuickBlocks with the ABI to your smart contract, it can deliver what we call articulated data. By this we mean that instead of returning data in the language of the Ethereum node (blocks and transactions and receipts and logs), we return data in the language of your smart contract (transfers and votes and proposals and expenditures). This eases the burden on your dApp developers.

Whereas the web3.js library delivers nearly identical data as is retrieved from the RPC interface, making it difficult for any but the most well versed in the data to easily use it, QuickBlocks stands between the node and your application improving the data significantly in two ways: (1) it's much faster, and (2) it's translated into the language of the smart contract. See quickblocks.io for more detail.

View our open-source library at https://github.com/Great-Hill-Corporation/quickblocks.