# Secret Race Strolling - A scalable privacy competition platform

Semester Project at SPRING

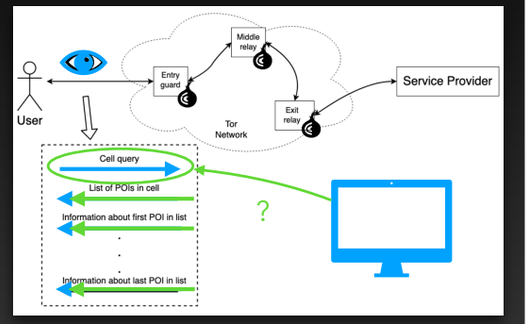
Tom Demont supervised by Bogdan Kulynych - June 2022



Hello, welcome presentation semester project
We called
Before we get started, let us relax and talk about strolling

# Back to Secretstroll

- Queries for point of interest around location
- Client-Server architecture, networking through Tor
- Students should:
  - Collect network trace
  - Extract features and create model
  - Classify from network trace to queried location




Secretstroll system architecture



Indeed, we must talk about secret stroll
Building a toy app
As you can observe in this picture from the handout
In the last part of the project

Will observe they can obtain excellent results. Toy app leaves a clear fingerprint for different cell id queries. They should talk about counter-measures

# Exploit the privacy evaluation framework

- The privacy evaluation framework is there!
  - From theoretical questions to practical implementation
- AICrowd and Kaggle → ML Competition
  - Bring it further: p2p attack/defence competition
- Counter-measure parsing, data sets generation, automated evaluation
- Fully exploit the pedagogical utility of Secrestroll!

AIcrowd

kaggle

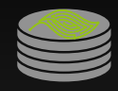

CS-523 student in a gladiator fight for SecretStroll



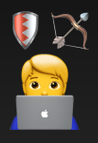
Our project stands here
Students already have built a lot at this point of the project
Inspired by other ML competition platforms
Append some features to existing platforms
Allow students to taste more of the PETs implementation and use the app they built

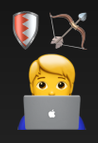
# Secret Race Strolling - designing a competition

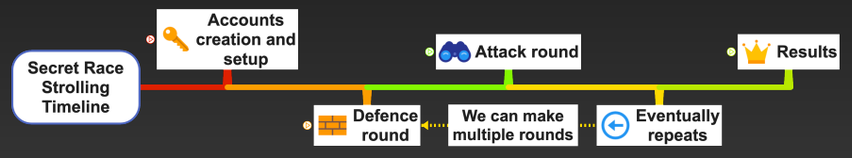
## Create a timeline



- Imagined 2 approaches:
  - Fully interactive
  - Round based
- Considered interactivity, scalability and student workload








The final Secret Race Strolling timeline



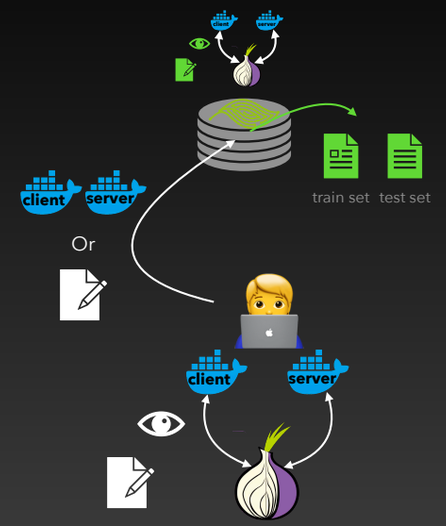
Now sure of it: we want a platform! Put in engineer's shoes, wanted create a timeline how the competition should be.
Imagined 2 approaches:
- fully: student can update their defence and attack others at any point in time during the competition
- round: in a first phase, students update their defence, up to the beginning of the second phase, where they start attacking each other

Decide on the best one, considered many aspects.
Interactivity: more interesting fully, direct response
Scalability: more deterministic server load with segmented
Workload: afraid fully is overwhelming. Rounds fit schedule
Came to the conclusion round approach preferable, described it timeline defined structure project [describe timeline]

# Secret Race Strolling - designing a competition

## Attack and defence format

- Format of evaluated files
- Defence:
  - Key decision: student generated trace vs. server generated trace
  - Scalability and fairness issues
  - Went with student trace
- Attack:
  - AICrowd and Kaggle way: csv classification...
  - ... Classifier probability of each cell id!



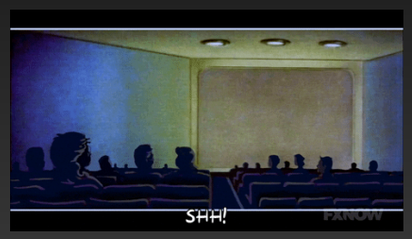



Know what SRS looks like, now. Decide on interface
Defence, problem, explain
    Scalability: trace collection borrows ressources long time, hard if 40 teams
    Fairness: cannot prevent cheating (passing trace w/ dummy app) vs. take care providing deterministic output
    Decided student trace: easier/faster to implement, code could by reused, left room for the other implementation
Attack, we followed
    classification in a csv file
    asked full proba classif for every cell_id. Compute cool metrics

We have a structure, we defined interface with user, what's next ...

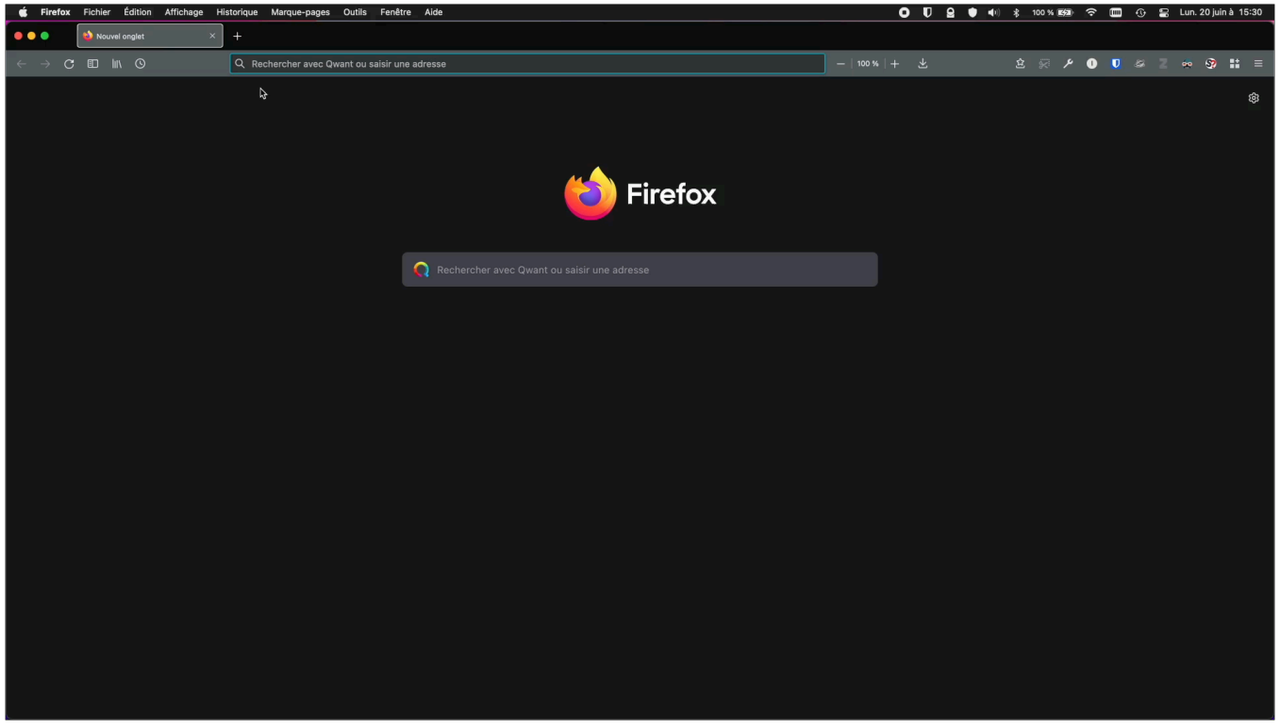
# Implementation considerations

- Scoring & ranking: reflect attack/defence performance in one value
- Technological choices: scalable, user-friendly, developer-friendly
- ... But it's demo time!





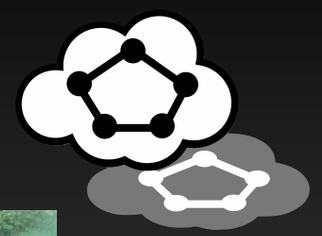

Do not hesitate to ask questions about these in the question time
However it was important to us to include some quick demo time!
...




Hope enjoyed this quick demo! Let me go to the last part ...

# Future work

- Server side trace generation: Shadow
  - Simulate tor network: scalability + fairness
- We tried it
  - Not stable docker
  - Ran simple non-tor network
  - Faced config generation tools

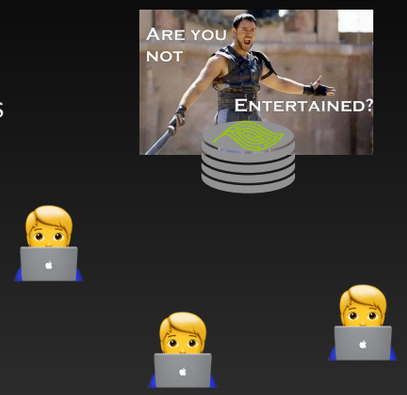
Shadow logo



Further steps: solve server-side generation issues
Found a tool: ...
Simulate avoids real tor overhead + deterministic outputs
In the end of the project, we tried:
docker image not yet fully working (lost lot of time)
ran on native amd64 simple client-server
more complex networks: needs heavy tools, didn't have time

# Conclusion

- Designed a platform fitting CS-523 requirements
  - Evaluated different approaches on timeline, defence/attack format, ranking metrics and technological choices
- Implemented the platform
  - Working prototype
  - Friendly for appending future work



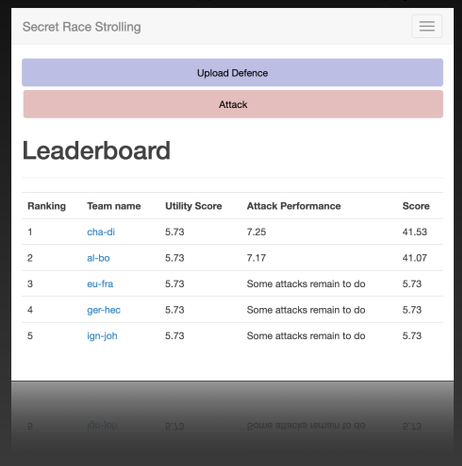

CS-523 students, waiting for more Secretstroll

# Secret Race Strolling - designing a competition

## Scoring system

- Give a score reflecting attack/defence performance → summary in one number for ranking
- Utility metric: over all recorded queried - $med_{in-volume} * med_{out-volume} * med_{time}$
- Attack performance metric: $roc\_auc\_score$
- Final score: ratio of both

Screenshot from the application on leaderboard page



# Implementing this platform

- Flask: popular lightweight web framework
- Handled background async tasks with Celery
- Object Relational Model - SQLAlechmy: database agnostic

SQLAlchemy logo


Celery logo

Flask logo

# Mental trace upload load

- Not breaking the server when teams upload
  - Avoid having RAM issues (CSV loaded in memory)
  - Avoid network congestion

Expected size of student uploaded/downloaded files

| | Defence upload | Attack upload |
|---|---|---|
| **Unit CSV size** | 48.1MB | 879KB |
| **Unit compressed CSV size** | 14.2MB | 61KB |
| **40 Teams CSV size** | 1.92GB | 35.2MB |
| **40 Teams compressed CSV size** | 568MB | 2.4MB |

From the raw Secretstroll implementation. Defence uploads could be bigger