User Identification based on HTTP and TCP Traffic

Matej Majdiš

In this presentation

1. Topic introduction & Goals
2. Motivation
3. How it works (top level view)
4. Unique identifier structure
5. Similarity search algorithm
6. Technologies used
7. Actual Java implementation
8. Conclusion

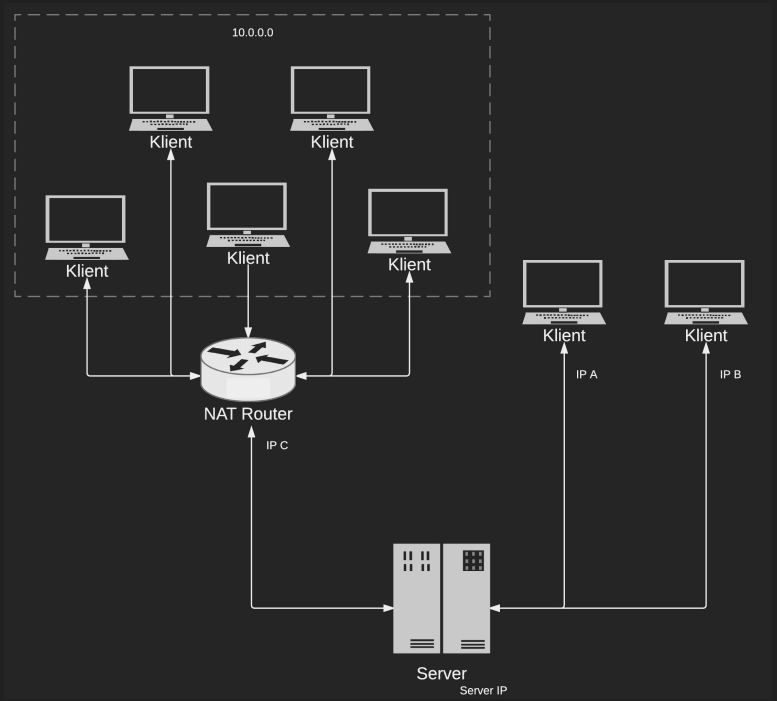
Topic introduction & Goals

- What ?
  - Create generally usable algorithm to
    - Identify single user connected to server
    - User can use multiple connections and multiple devices

- How ?
  - Analyze HTTP and TCP traffic
    - Extracting specific information
  - Create unique footprint (UFoo) for every single request
  - Group similar UFoos to identify single user relation

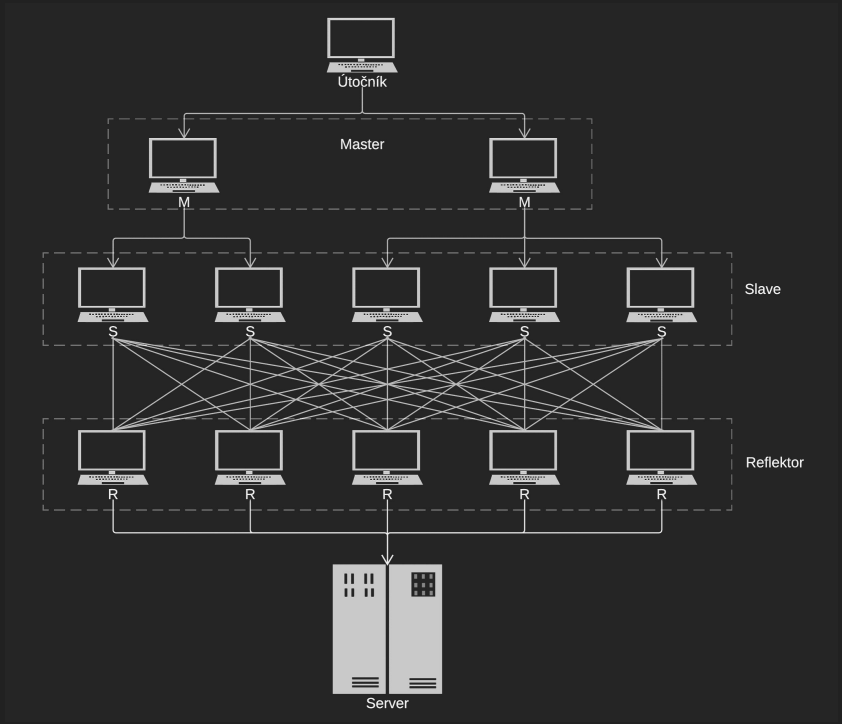
Motivation

- To identify actual single user, which server is communicating with
- Current approach
  - IP based identification (most cases)
- Problems
  - NAT networks
  - Proxy servers
  - Actual IP of user is not accessible



Motivation 2

- To effectively prevent DOS and D-DOS types of attacks
- Current approach
  - Application filters
  - IP based rejecting of requests
- Problems
  - Not usable in all cases
  - One attacker using multiple devices



How it works ?

- 5 main steps

1. Handle request before it’s processed by host application
2. Collect it’s HTTP and TCP data
   a. HTTP data from request
   b. TCP data from monitor of network traffic
3. Process the data and create unique footprint for request

How it works ?

- 5 main steps

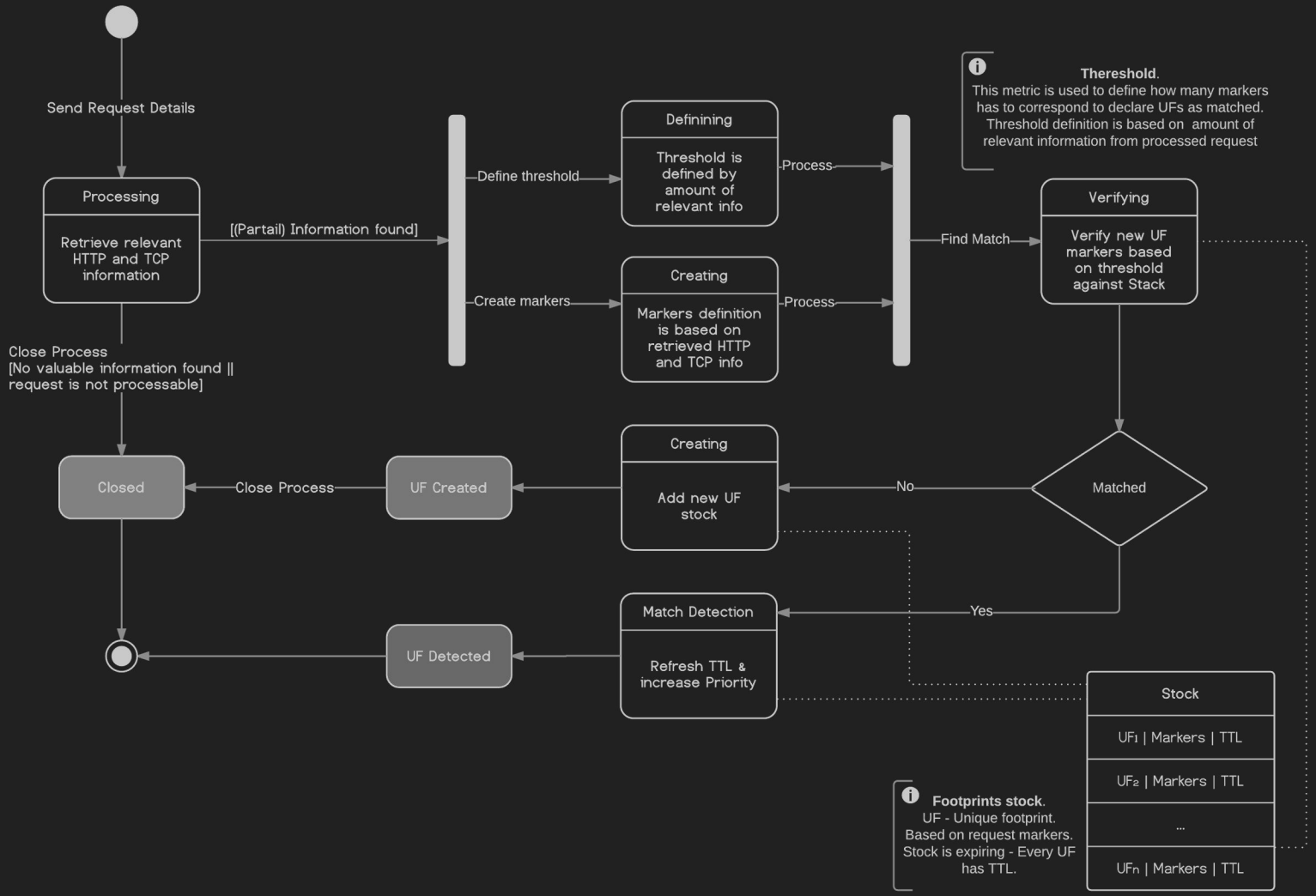
4. Find nearest neighbour of actual footprint in stock

5. Return distance and perform reaction

    a. Create new UFoo entry in stock (distance above threshold)

    b. Match UFoos and increase its frequency (distance below threshold)

    c. Possibly notify host application (UFoo frequency is above the limit


Send Request Details
Processing
Retrieve relevant HTTP and TCP information
[(Partail) Information found]
Close Process
[No valuable information found || request is not processable]
Define threshold
Create markers
Definining
Threshold is defined by amount of relevant info
Creating
Markers definition is based on retrieved HTTP and TCP info
Process
Process
Find Match
Thereshold.
This metric is used to define how many markers has to correspond to declare UFs as matched. Threshold definition is based on amount of relevant information from processed request
Verifying
Verify new UF markers based on threshold against Stack
Matched
No
Yes
Creating
Add new UF stock
UF Created
Close Process
Closed
Match Detection
Refresh TTL & increase Priority
UF Detected
Stock
UF1 | Markers | TTL
UF2 | Markers | TTL
...
UFn | Markers | TTL
Footprints stock.
UF - Unique footprint.
Based on request markers.
Stock is expiring - Every UF has TTL.

UFoo Structure

- Two parts
  - Static data - used for similarity search - stored as String
  - Relation data - used for additional computing operations between 2 requests in relation
- Relation data are composed of
  - Relation headers
    - Forwarded
    - X-csrf-token

    } HTTP Headers
  - Timestamp - TCP Timestamp
  - Country - Geo IP Country based on MaxMind library

UFoo Structure

- Static data are composed of
  - Static headers TCP Data
    - accept
    - authorization
    - cache-control
    - cookie
    - content-length
    - content-type
    - user-agent

    HTTP Headers
  - Unknown headers - other HTTP headers which were not specified
  - IP address
  - Country code
  - City
  - Encoding
  - Locales
  - Servlet Path

    Request Data
  - Tcp Window
  - Tcp Length

    TCP Data

Similarity search algorithm

- Input
  - Current UFoo and UFoo from stock
  - Containing all gathered data
- Phase 1
  - Direct distance is computed on static data part
    - Based on Jaccard index
      - Jaccard index for every subset of static data e.g. Headers
      - Simple attributes e.g Servlet Path are compared as Strings
    - Every attribute and sub-attribute has it’s weight

Similarity search algorithm

- Phase 2
  - Resulting distance is modified by relation data connections
    - Timestamp difference
    - Safe and less safe zones
- Output
  - Algorithm returns nearest neighbour and it’s distance

Implementation - Technologies

- Java SE + AOP
  - Framework independent implementation
- Aspects for injecting request methods
  - Single aspect for every application type
- MaxMind’s Geo IP detection
  - Used to determine Country and City based on IP address
- Example Client and Server implemented

Implementation

- Main logic is contained in 4 casses
  - `Injector` - collecting HTTP requests
  - `Serializer` - serialization of data into UFoo
  - `UFooProcessor` - analyzes UFoo data and similarity results
  - `FootprintSimilarityService` - nearest neighbour search logic itself
- In the moment implementation part is finished
- Calibration of weights and threshold needs to be done

Conclusion

- Algorithm design
- Main implementation part

} Finished

- Calibration of parameters
- Test on real data

} Needs to be done

Thank you :)

Q & A