Approximate search in large security-related data sets

Slobodan Petrović, NTNU in Gjøvik

Norwegian University of Science and Technology

Contents

- Intrusion detection and prevention
- Handling large data volumes
- Approximate search – dataset reduction
- Constrained approximate search

Intrusion detection and prevention

- Large networks everywhere, including the maritime environment (on a single ship)
  - *Internet of things* – networked shafts, wheels, gears, transported goods, etc. – sensors everywhere (so-called surprise prevention mechanisms)
  - Little computing power on the sensors (often nanosensors)
    - Difficult to implement complex security mechanisms on sensors
    - What remains (since the sensors use their computing power to communicate with the rest of the network) – *detect attacks in networks*

Intrusion detection and prevention

- Unmanned ships
  - Remote controlled, problems
    - Jamming, also GPS jamming
    - Penetration into networks
    - Taking control over ship(s)
  - Connection of vessels through a network
    - Can be attacked as any other network
    - And defended as well

Intrusion detection and prevention

- Detecting attacks (on hosts and networks)
  - Huge amount of traffic and large log files – manual analysis impossible
  - Automatic analysis – Intrusion detection systems (IDS)
  - Blocking malicious activity – Intrusion prevention systems (IPS)
  - IDS/IPS classification
    - Misuse detection – known attack patterns, alarm if match
    - Anomaly detection – profiles of normality – alarm if non-match

Handling large data volumes

- *Big data*
  - Large database of known attack patterns
  - Even larger database of profiles of normal behavior
    - Defining normality more difficult than defining malicious activity
  - Every data unit (typically a packet or a session) to be checked against all the profiles in the database in the worst case
    - Slow – sophisticated search algorithms needed, implemented in advanced hardware
    - Even the best known search algorithms are not good enough

Handling large data volumes

- To process so large data sets in *real time*
  - Invent better search algorithms
    - Difficult, we have already achieved sub-linear time complexity
    - Bit-parallel techniques, parallel processing in general, implementation on fast hardware
  - Reduce the data set
    - More possibilities to do that – current data sets are often redundant
    - Reducing redundancy – fingerprinting, more intelligent data analysis

Approximate search

- An intelligent way of reducing the dataset size
  - Redundancy in the dataset to exploit
    - Many attacks have a common predecessor – similar attacks – small difference in signatures
    - In spite of that fact, current IDS store a signature for each variation of the same attack
      - A consequence of using *exact search*
      - A single bit of attack traffic changed – new signature needed, otherwise missing the event (*false negative*)

Approximate search

- Solution – use *approximate search*
  - Allow tolerance at matching database entries with traffic patterns
    - Up to *k* errors allowed
- This effectively reduces the size of the signature database
  - No need to store a separate signature for every new variation of the same attack
  - Consequence – save time needed to browse through the whole signature database

Approximate search

- But, approximate search algorithm must be very efficient
  - Comparable with efficiency of exact search
  - Currently, all known misuse-based IDS (at least open-source, like Snort, Suricata) use the Aho-Corasick multi-pattern exact search algorithm
    - Worst-case time complexity equal to average-case time complexity
      - Resistant to *algorithmic attacks* – traffic that makes the algorithm perform poorly
    - Too slow for big data

Approximate search

- So-called *bit-parallel* algorithms
  - Exploit inherent parallelism of computer words (64 bits, 32 bits)
  - If the search pattern length is smaller than the length of the computer word – significant gain in efficiency
  - The most important result in search technique development in the last 25 years (Baeza-Yates and Gonnet, 1992)
  - Most important for IDS – *approximate search* also possible following the same lines (Wu and Manber, 1992)

Approximate search

- Problems with approximate search
  - It is not enough to introduce tolerance on number of changed symbols (by deleting, inserting, substituting)
  - Distribution of changes also matters, as well as the numbers of individual change operations
  - Consequence – increased number of *false positives*
  - To cope with this, it is possible to introduce *constraints* in the search algorithms, also in bit-parallel approximate search algorithms

Constrained approximate search

- To introduce constraints in an appropriate way
  - A priori knowledge about the change procedure used by the attacker is necessary
    - Possible, since the attackers often use tools to deliberately change the attack patterns and create new attacks based on the old ones in order to pass unnoticed by IDS
    - The tool parameters are often maximum numbers of particular change operations (deletions, insertions, substitutions of symbols) or maximum lengths of runs of deletions/insertions

Constrained approximate search

- If the constraints in search are properly introduced
  - The false positive rate is kept under control
- We also need efficient (bit-parallel) constrained approximate search algorithms
  - Constraints introduced by means of different bit-masks
  - The bit-masks are applied efficiently
    - A single AND operation with the mask for each computer word used in the search algorithm

Constrained approximate search

- Approximate bit-parallel search algorithm with constraints (1)
  - Simulates so-called Non-deterministic Finite Automaton (NFA) assigned to the search pattern
  - NFA has matrix form
    - With 0 errors in search – 0-th row of the matrix
    - With 1 error – 1st row
    - …
    - With *k* errors – *k*-th row

Constrained approximate search

- Approximate bit-parallel search algorithm with constraints (2)
  - Each row in the matrix consists of automata capable of matching a single character (the current input character from the search string) – *m* such matchers per row, where *m* is the length of the search pattern
  - Constraints introduced by special bit-masks
    - For each row, the bit-mask is obtained from the previous row
    - Counters are assigned each matcher – count modulo the constraint

Constrained approximate search