Conditional Anonymization Based on Field Values in FLAIM

Aditya Khanna, Indian Institute of Technology, Delhi
ADVISOR: Adam Slagell

Background

FLAIM is a modular, open-source anonymization framework created for system/network/security administrators. The security engineers need to share logs to understand attacks from a larger perspective. Networking researchers, educators, and log analysis tools’ developers need large data sets for research, training, and testing purposes. Anonymization is needed as logs are sensitive and can reveal security weaknesses to outsiders.

FLAIM’s salient features are:
- A repository of up-to-date and cutting-edge anonymization algorithms.
- Supports multiple logs.
- Anonymization based on log type and log fields.
- Has an extensible, modular architecture.

Research Plan

• Understanding the motivation behind the software, its use, and its architecture.
• Learning XML schemas, XSLT, RELAX NG, and Schematron languages in detail.
• Modifying the user policy and the schemas that validate the user policy as required.
• Understanding the policy manager and the anonymization engine and chalking out a path on which changes would be done.
• Implementing conditionals according to the plan by modifying FLAIM core files.
• A Policy Manager that provides flexible anonymization schemes for each field and for different set of values in a field.
• A new User Policy Language that also supports the old version of the language.
• A Module Schema Language that gives the module developer the choice to include conditionals suitable for the specific log.
• An Anonymization Engine that processes both types of fields, the ones using an algorithm uniformly and the others using conditionals.
• A Module that can send to the FLAIM core what a conditional means; otherwise the default definitions in base classes are used.

Goals

• Motivation:
  - There are cases where we want to anonymize based on a field’s value, like anonymizing internal hosts differently from external hosts.
• Goals:
  - To provide more flexible anonymization schemes by implementing conditionals within a field in the user policy so that the users can anonymize different values of a field differently as required.
  - To give users the power to anonymize related data more efficiently assuming the user understands the relationship between correlated fields.

Fundamental Questions/Challenges

• What changes do the anonymization engine and the anonymization process need?
• How to give the module developer a choice to define the conditionals suitable to the log type?
• How to make a new version of the user policy language such that it supports the old policy language too?
• What all needs to change in the policy manager and validating schemas to implement conditionals?

Related Work/Interaction with Other Projects

Some other anonymization tools:
- Greedy Partitioning Algorithm
- Greedy Top Down Specialization method (TDS)
- K-Anonymization Algorithm
- CANINE
- IHSN
- SCRUB-PA
- LOBSTER
- Crypto-PAN
- Info Gain Mondrian
- Median Mondrian
- LSD Mondrian