BitMiner: Bits Mining in Internet Traffic Classification

Zhenlong Yuan†, Yibo Xue‡ and Mihaela van der Schaar¶
†Department of Automation, Tsinghua University, Beijing, China
‡Department of Electrical Engineering, UCLA, Los Angeles, CA, USA
¶Tsinghua National Lab for Information Science and Technology, Beijing, China
†‡Research Institute of Information Technology, Tsinghua University, Beijing, China
yuanzl11@mails.tsinghua.edu.cn, yiboxue@tsinghua.edu.cn, mihaela@ee.ucla.edu

ABSTRACT
Traditionally, signatures used for traffic classification are constructed at the byte-level. However, as more and more data-transfer formats of network protocols and applications are encoded at the bit-level, byte-level signatures are losing their effectiveness in traffic classification. In this poster, we creatively construct bit-level signatures by associating the bit-values with their bit-positions in each traffic flow. Furthermore, we present BitMiner, an automated traffic mining tool that can mine application signatures at the most fine-grained bit-level granularity. Our preliminary test on popular peer-to-peer (P2P) applications, e.g. Skype, Google Hangouts, PPTV, eMule, Xunlei and QQDownload, reveals that although they all have no byte-level signatures, there are significant bit-level signatures hidden in their traffic.

CCS Concepts
• Networks → Network management;

Keywords
Traffic classification, bit-level signatures, bits mining

1. INTRODUCTION
Signature based traffic classification has been playing an important role in a broad range of network operations and security management, such as quality-of-service control and intrusion detection. However, due to the increasing number of network applications and their frequent updates, it is becoming more challenging to keep track of the signatures. To address this challenge, a number of existing solutions have focused on automatically extracting signatures at the byte-level [4, 5], which first divide packet payloads into groups of consecutive bytes and then analyze to get the possible signatures. However, these solutions have two major limitations. Firstly, they are unable to discover signatures at the more fine-grained bit-level granularity. Note that previous work [1, 2] have revealed that bit-level characteristics are of great importance in identifying a few P2P applications. Secondly, they confine signatures to groups of consecutive bytes and thus are hard to discover the signatures that consist of inconsecutive bytes (e.g. 1 byte) in packet payloads. In this poster, we propose the novel bit-level signatures, and present an automated traffic mining tool (BitMiner) that can mine signatures at the most fine-grained bit-level granularity.

2. BITMINER
In this poster, we have two observations. The first is that an application signature should be robust enough to support per-flow identification due to the prevalence of asymmetric routing. For this reason, a favorable application signature should be one of the most frequent patterns in captured traffic after running an application for plenty of times. Therefore, our goal can turn into mining the most frequent patterns' in the application traffic. The second is that the bit-value of a bit-position in a flow often determines the bit-values of other bit-positions in this flow. Therefore, we are motivated to associate all the bit-values with their bit-positions in a flow for frequent pattern mining.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

SIGCOMM ’15 August 17-21, 2015, London, United Kingdom © 2015 Copyright held by the owner/author(s).
DOI: http://dx.doi.org/10.1145/2785956.2789997

From here, we start using some terms in Data Mining.
packet payload because the MTU of an IP packet over Ethernet networks is 1500-byte where 1 byte has 8 bit-orders.

BitMiner consists of two parts: Bit-table and Miner-tree. Figure 2 shows an example of how BitMiner works. Bit-table is a hash table used for hashing and storing all the items read from a transaction database. In this process, Bit-table will read the transaction database twice. For the first time, Bit-table will count the support of every item. For the second time, Bit-table will remove the items whose support is below the initially set support threshold and sort the remaining items in every transaction by their supports (maximum to minimum). After that, all the sorted transactions will be entered into Miner-tree as a new transaction database.

Miner-tree is a prefix tree of the new transactions, which takes idea from the FP-tree [3] but is different. Note that there are probably multiple tasks running within an application and thus the signature could be a regular expression. Considering a transaction (flow) can only belong to one of the tasks, all the transactions are divided into multiple clusters to represent different tasks. Since the items in each transaction have been sorted by their supports, it is extremely fast to construct the Miner-tree.

After constructing the Miner-tree, there will be a pruning process controlled by two thresholds: minimum support and minimum confidence. Particularly, the support (defined as the proportion of transactions in a node from the whole transaction database) will be checked for every single node. Moreover, the confidence (defined as the proportion of transactions in all the child-nodes of a node from the node itself) will be checked for every parent node. In this way, it can be determined whether a branch should be removed or a parent node should stop splitting. Finally, the branches of the pruned Miner-tree are the target signature.

3. EVALUATION

BitMiner has been tested on the UDP traffic of six popular applications. As shown in Table 1, every signature is generated by BitMiner within a few seconds. The “(p)” represents a pattern (p) matching within one packet’s payload, the “∧(p)” represents this matched packet is the first packet of a flow, the “(p)$” represents this matched packet is the last packet of a flow, the “(p)+” represents this matched packet appears one or more times in succession within a flow, the “(p)^” represents this matched packet appears zero or more times in succession within a flow, the “002_0x02” represents the third byte value of a packet’s payload is 0x02, the “002_4_1” represents the fifth bit value of the third byte is 1, the “pkp” represents two patterns matching with one packet’s payload simultaneously and the “(p)|(p)” represents either one matched packet appears within a flow. For instance, the third byte values of the first one or more packets of a Skype flow are always 0x02 while five bit values of the third bytes of all the other packets are fixed. Specially, we also examine the other bits adjacent to the mined ones, such as the ‘second, third and fourth’ bits of the third bytes of Skype flows and the ‘fourth, fifth, sixth, seventh and eighth’ bits of the fourth bytes of Thunder flows. The results show that those bit-values are completely random (i.e. uniformly distributed). Also as shown in Table 1, the support represents the proportion of flows matched with the mined signature, which is equivalent to the recall in traffic classification. In addition, a longer signature generally means a better precision. For example, if we check the first 10 packets of a Thunder flow, the signature used for matching is totally 40 bits long, which may be robust enough to get a high precision in real-world situations.

4. ACKNOWLEDGEMENT

This work was supported by the National Key Technology R&D Program of China under Grant No.2012BAH46B04.

5. REFERENCES


