Intrusion Detection and Malware Analysis
Automatic signature generation

Pavel Laskov
Wilhelm Schickard Institute for Computer Science
The quest for attack signatures

- Post-mortem: security research, computer forensics
- Reactive: analysis of anomalies (forensic sinks)
- Proactive: acquisition and analysis of malicious data
A general framework for ASG

- **Clustering**: finding groups of similar malicious events
- **Token extraction**: finding common patterns in malicious data
- **Signature assembly**: assessment of extracted tokens
Invariance as a main principle of ASG

- Invariance is inherent for attacks due to extremely specific nature of exploits.
- Diversity makes signatures more general and accurate.
- Too much diversity makes signatures smaller and leads to false positives.
Token extraction: basic definitions

- A token is a substring found in malicious content that satisfies pre-defined empirical conditions, such as:
  - minimal length
  - minimal support: percentage of malicious events it occurs in
- A pair of tokens is said to be distinct if they are not a substring of one another.
- A token $s$ that is a substring of another token $t$ is ignored unless it satisfies tokenization conditions while being not part of $t$. 
Problem
Given a set \( \{s_1, \ldots, s_n\} \) of malicious payloads, find a set \( \{t_1, \ldots, t_k\} \) of tokens, such that \( |t_i| > L_{\text{min}} \) and each \( t_i \) occurs in at least \( v \% \) of malicious payloads.

Remark
Unlike many other applications of tokenization, tokens for ASG are not defined in terms of delimiters. Such delimiters may not be known in advance.
Generalized suffix tree (GST)

- A suffix tree for more than one string.
- Creation: concatenate two strings with different delimiters and build a single suffix tree
- Internal nodes are augmented with arrays of leaf counts for all strings.
- Example: GST for x = 'abbaa' and y = 'baaaab':

![Diagram of GST for x = 'abbaa' and y = 'baaaab']
Token extraction using GST

- Traverse a GST from top to bottom.
- For each node, output its path from the root if its depth is greater than $L_{\text{min}}$ and the number of non-zero entries in its leaf count is greater than $v n$. Output the percentage of non-zero entries in its leaf count as a token support.
Open problems

- How can we define unique “end-of-string” markers for a full alphabet of byte values?
- How can we avoid generation of non-distinct tokens?
Signature assembly

- **Goal:** remove tokens that frequently occur in normal traffic.
- **Rules for removal:**
  - $v_-(t_i) > v_+(t_i)$
  - $v_-(t_i) > v_{max}$
- **Underlying problem:** set matching.
- **Algorithms:**
  - Knuth-Morris-Pratt: $O(k(n + M))$
  - Aho-Corasick: $O(k + n + M)$
Given the set of token/support pairs \( \{(t_1, \nu_1), \ldots, (t_k, \nu_k)\} \), signature refinement consists of the following steps:

- Normalize individual support values so that they sum to 1:
  \[
  \nu_i = \frac{\nu_i}{\sum_{j=1}^{k} \nu_k}
  \]

- Detection rule: let \( t_{i_1}, \ldots, t_{i_m} \) be a set of tokens identified in a given string. Then the string is marked as malicious, if \( \sum_{i} \nu_{ij} > \theta \).

- The threshold \( \theta \) is determined by calibration on the set of benign data so as not to exceed some minimal false positive rate.
Lessons learned

- Automatic signature generation enables one to quickly extract signatures for samples of malicious and benign traffic.
- Careful choice of algorithms and data structure is important for practical feasibility of ASG.
- ASG enable some very interesting applications to malware analysis, especially detection of malware communication.
D. Gusfield.  
*Algorithms on strings, trees, and sequences.*  

Konrad Rieck, Guido Schwenk, Tobias Limmer, Thorsten Holz, and Pavel Laskov.  
*Botzilla: Detecting the "phoning home" of malicious software.*  
In *Proc. of 25th ACM Symposium on Applied Computing (SAC),*  
March 2010.  
(to appear).