**Fast and Scalable Signature Matching with Extended Finite Automata**

Randy Smith, Cristian Estan, Somesh Jha
University of Wisconsin—Madison

**Motivation**
- Network Intrusion Detection Systems (NIDS) protect networked computing resources from unwanted or malicious traffic.
- NIDS are between a rock and a hard place:
  - Increasing traffic volume
  - Increasing signature complexity
  - Increasing number of signatures
  - Few cycles available for detection
- Current matching techniques require too much time or too much memory (or both)
- Goal: Develop signature matching mechanisms with smaller memory footprints and time requirements.

**Experiments**
- Measure XFA space savings and performance penalty applied to commercial signatures:
  - Extract 1600+ regular expressions from FTP, SMTP, and HTTP protocol signatures, drawn from Sourcefire (Snort) and Cisco IPS rule sets.
  - Compare XFAs to other mechanisms: DFAs, D^2FAs, and DFA Set Splitting (mDFAs).
- Methodology:
  - For each inspection mechanism, combine signatures from the same source and protocol.
  - Use combined signatures to inspect traces of actual traffic. Measure running time and memory usage.

**Results**
- XFA representation yields fast inspection with small memory usage.
- XFAs smaller and faster than best alternatives for most complex rule sets.
- Space-time tradeoff: perform slightly more work per byte (to execute instructions) in exchange for a drastic reduction in space.

**Approach**
- State-space explosion occurs because DFAs are ambiguous, with many distinct paths to a state.
- Use alternate representation that employs bits, counters, and other data structures to track inspection progress and reduce ambiguity.
- Bit and counter values reside in an auxiliary "scratch memory" and are updated by small programs during inspection.
- Apply compiler construction techniques to optimize the resulting XFAs.

**RegExps**
- \( /na[^\n]{200}/ \)
- \( /.*bc/ \)

**DFA**

**XFA**

**Fast and Scalable Signature Matching with Extended Finite Automata**

Randy Smith, Cristian Estan, Somesh Jha
University of Wisconsin—Madison

**Motivation**
- Network Intrusion Detection Systems (NIDS) protect networked computing resources from unwanted or malicious traffic.
- NIDS are between a rock and a hard place:
  - Increasing traffic volume
  - Increasing signature complexity
  - Increasing number of signatures
  - Few cycles available for detection
- Current matching techniques require too much time or too much memory (or both)
- Goal: Develop signature matching mechanisms with smaller memory footprints and time requirements.

**Experiments**
- Measure XFA space savings and performance penalty applied to commercial signatures:
  - Extract 1600+ regular expressions from FTP, SMTP, and HTTP protocol signatures, drawn from Sourcefire (Snort) and Cisco IPS rule sets.
  - Compare XFAs to other mechanisms: DFAs, D^2FAs, and DFA Set Splitting (mDFAs).
- Methodology:
  - For each inspection mechanism, combine signatures from the same source and protocol.
  - Use combined signatures to inspect traces of actual traffic. Measure running time and memory usage.

**Results**
- XFA representation yields fast inspection with small memory usage.
- XFAs smaller and faster than best alternatives for most complex rule sets.
- Space-time tradeoff: perform slightly more work per byte (to execute instructions) in exchange for a drastic reduction in space.

**Approach**
- State-space explosion occurs because DFAs are ambiguous, with many distinct paths to a state.
- Use alternate representation that employs bits, counters, and other data structures to track inspection progress and reduce ambiguity.
- Bit and counter values reside in an auxiliary "scratch memory" and are updated by small programs during inspection.
- Apply compiler construction techniques to optimize the resulting XFAs.

**RegExps**
- \( /na[^\n]{200}/ \)
- \( /.*bc/ \)

**DFA**

**XFA**