Approximate Reduction of Finite Automata for High-Speed Network Intrusion Detection

Milan Češka Vojtěch Havlena Lukáš Holík
Ondřej Lengál Tomáš Vojnar

Brno University of Technology
Czech Republic

18 April 2018 (TACAS'18)

Main Points

- reduction of nondeterministic finite automata (NFAs)

Main Points

- reduction of nondeterministic finite automata (NFAs)
- the reduction does NOT preserve language

Main Points

- reduction of nondeterministic finite automata (NFAs)
- the reduction does NOT preserve language
- BUT guarantees maximum error

Main Points

- reduction of nondeterministic finite automata (NFAs)
- the reduction does NOT preserve language
- BUT guarantees maximum error
- w.r.t. a probabilistic distribution

Main Points

- reduction of nondeterministic finite automata (NFAs)
- the reduction does NOT preserve language
- BUT guarantees maximum error
- w.r.t. a probabilistic distribution
- application in high-speed network intrusion detection

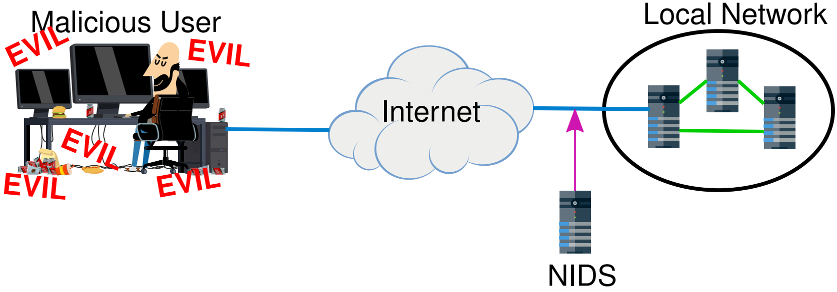
Computer Network Intrusion Detection

- recently a large number of security incidents, e.g.
  - WannaCry
    - ransomware, 1 G$
  - Spectre & Meltdown
    - security vulnerabilities in Intel CPUs

SPECTRE MELTDOWN

- exploits often spread via networks
  - these attacks **can be detected**

Computer Network Intrusion Detection



- NIDS = Network Intrusion Detection System

Computer Network Intrusion Detection

- SNORT
  - popular NIDS
  - **RegExes** to describe attacks

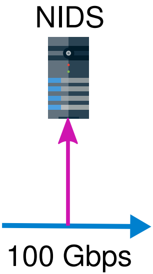
Computer Network Intrusion Detection

- SNORT
  - popular NIDS
  - **RegExes** to describe attacks

```
/^POST HTTP\/1\.[01]\r\n(\V+\r\n)*\r\n[\x00-\xff]*DROP TABLE/
/^HTTP\/1\.[01] 404[\x00-\xff]*(admin|wordpress)/
/^POST HTTP\/1\.[01]\r\n(\V+\r\n)*\r\n[\x00-\xff]*admin:admin/
/^POST HTTP\/1\.[01]\r\n(\V+\r\n)*\r\n[\x00-\xff]*admin:password/
/^POST HTTP\/1\.[01]\r\n(\V+\r\n)*\r\n[\x00-\xff]*YWRtaW46cGFzc3dvcmQ/
/^POST HTTP\/1\.[01]\r\n(\V+\r\n)*\r\n[\x00-\xff]*YWRtaW46YWRtaW4/
/^POST HTTP\/1\.[01]\r\n(\V+\r\n)*\r\n[\x00-\xff]*\/bin\/sh/
```

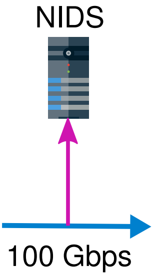
Computer Network Intrusion Detection

- High-speed networks
  - 100 Gbps, 400 Gbps



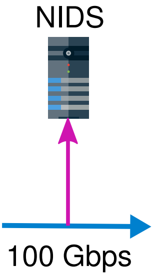
Computer Network Intrusion Detection

- High-speed networks
  - 100 Gbps, 400 Gbps
- **100 Gbps** — max. $\sim$150 Mpkt/s (100 / 84*8)
  - cf. 56 kbps dial-up — max. $\sim$80 pkt/s
  - $\sim$10 GB/s (of data)



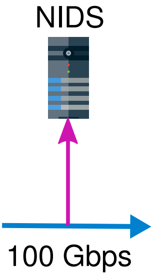
Computer Network Intrusion Detection

- High-speed networks
  - 100 Gbps, 400 Gbps
- **100 Gbps** — max. ∼150 Mpkt/s (100 / 84*8)
  - cf. 56 kbps dial-up — max. ∼80 pkt/s
  - ∼10 GB/s (of data)
- consider 4 GHz CPU
  - 0.4 cycle/B
  - ∼27 cycles/pkt cf. DRAM latency ∼100 cycles



Computer Network Intrusion Detection

- High-speed networks
  - 100 Gbps, 400 Gbps
- **100 Gbps** — max. ~150 Mpkt/s (100 / 84*8)
  - cf. 56 kbps dial-up — max. ~80 pkt/s
  - ~10 GB/s (of data)
- consider 4 GHz CPU
  - 0.4 cycle/B
  - ~27 cycles/pkt cf. DRAM latency ~100 cycles
- **no hope** for SW solutions



HW-accelerated NIDS

HW-accelerated NIDS

HW-accelerated NIDS

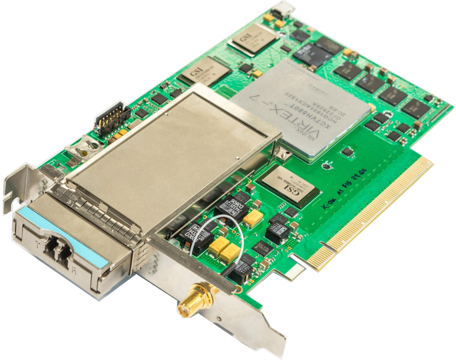
HW-accelerated NIDS

- cooperation with ANT@FIT

HW-accelerated NIDS

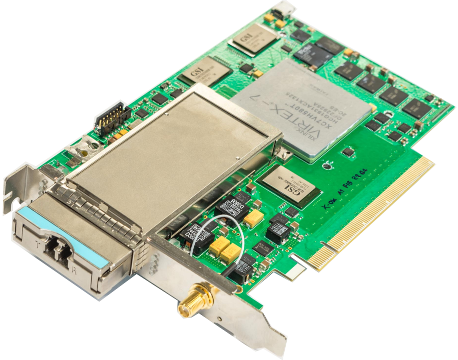
HW-accelerated NIDS

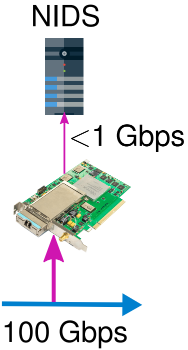
- cooperation with ANT@FIT
- using a COMBO-100G accelerator card
  - FPGA Xilinx Virtex-7 H580T



HW-accelerated NIDS

- cooperation with ANT@FIT
- using a COMBO-100G accelerator card
  - FPGA Xilinx Virtex-7 H580T





- used as a pre-filter

HW-accelerated NIDS

HW-accelerated NIDS

- RegEx matching in HW

HW-accelerated NIDS

- RegEx matching in HW
  - **NFAs**!

HW-accelerated NIDS

HW-accelerated NIDS

- RegEx matching in HW
  - **NFAs**!
  - smaller than DFAs
  - but still too big (even after language-preserving reduction)

HW-accelerated NIDS

HW-accelerated NIDS

- RegEx matching in HW
  - **NFAs**!
  - smaller than DFAs
  - but still too big (even after language-preserving reduction)
  - many units in parallel

HW-accelerated NIDS

HW-accelerated NIDS

- RegEx matching in HW
  - **NFAs**!
  - smaller than DFAs
  - but still too big (even after language-preserving reduction)
  - many units in parallel
  - `http-backdoor.pcre`: 38.4 Gbps

HW-accelerated NIDS

HW-accelerated NIDS

- RegEx matching in HW
  - **NFAs**!
  - smaller than DFAs
  - but still too big (even after language-preserving reduction)
  - many units in parallel
  - `http-backdoor.pcre`: 38.4 Gbps
  - $\rightsquigarrow$ language **non-preserving reduction**

Distance of NFAs

Language non-preserving NFA reduction $\mathcal{A} \to \mathcal{A}_{red}$:

Distance of NFAs

Language non-preserving NFA reduction $\mathcal{A} \to \mathcal{A}_{red}$:

- trivial solutions not satisfactory

Distance of NFAs

Language non-preserving NFA reduction $\mathcal{A} \to \mathcal{A}_{red}$:

- trivial solutions not satisfactory
- need to quantify the error
  - distance of NFAs

Distance of NFAs

Language non-preserving NFA reduction $\mathcal{A} \to \mathcal{A}_{red}$:

- trivial solutions not satisfactory
- need to quantify the error
  - distance of NFAs

Distance of NFAs:

- Jaccard distance, Cesàro-Jaccard distance
- Levenshtein (edit) distance
  - not suitable for languages
- ...

Distance of NFAs

Language non-preserving NFA reduction $\mathcal{A} \to \mathcal{A}_{red}$:

- trivial solutions not satisfactory
- need to quantify the error
  - distance of NFAs

Distance of NFAs:

- Jaccard distance, Cesàro-Jaccard distance
- Levenshtein (edit) distance
  - not suitable for languages
- ...
- **not suitable!**
  - distribution of network packets is not uniform

Distance of NFAs

Probabilistic distance of NFAs:

Distance of NFAs

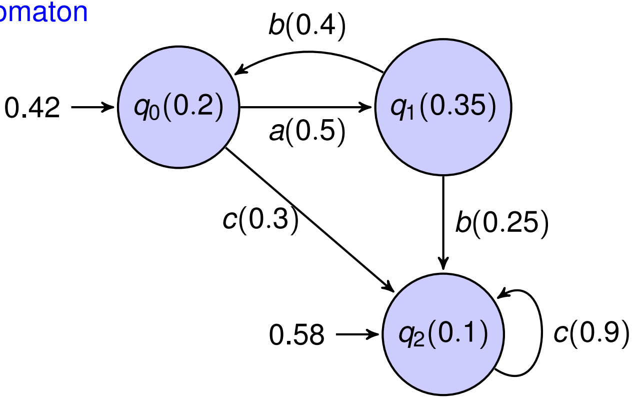
Probabilistic distance of NFAs:

- various packets have different likelihood
  - e.g. $Pr(\texttt{HTTP}) > Pr(\texttt{Gopher})$
  - e.g. $Pr_{\texttt{HTTP}}(\texttt{GET}) > Pr_{\texttt{HTTP}}(\texttt{POST})$

Distance of NFAs

Probabilistic distance of NFAs:

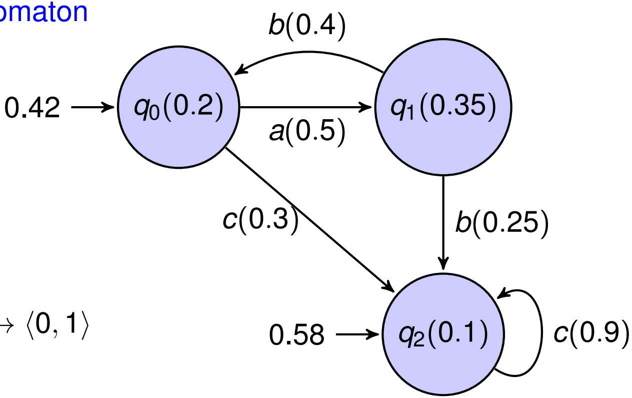
- various packets have different likelihood
  - e.g. $Pr(\texttt{HTTP}) > Pr(\texttt{Gopher})$
  - e.g. $Pr_{\texttt{HTTP}}(\texttt{GET}) > Pr_{\texttt{HTTP}}(\texttt{POST})$
- probabilistic automaton



Distance of NFAs

Probabilistic distance of NFAs:

- various packets have different likelihood
  - e.g. $Pr(\texttt{HTTP}) > Pr(\texttt{Gopher})$
  - e.g. $Pr_{\texttt{HTTP}}(\texttt{GET}) > Pr_{\texttt{HTTP}}(\texttt{POST})$
- probabilistic automaton

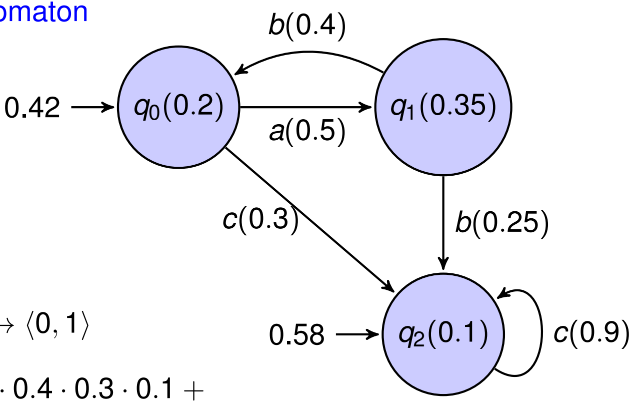


Represents $Pr_P : \Sigma^* \to \langle 0, 1 \rangle$

Distance of NFAs

Probabilistic distance of NFAs:

- various packets have different likelihood
  - e.g. $Pr(\texttt{HTTP}) > Pr(\texttt{Gopher})$
  - e.g. $Pr_{\texttt{HTTP}}(\texttt{GET}) > Pr_{\texttt{HTTP}}(\texttt{POST})$
- probabilistic automaton



Represents $Pr_P : \Sigma^* \to \langle 0, 1 \rangle$

$$Pr_P(abc) = 0.42 \cdot 0.5 \cdot 0.4 \cdot 0.3 \cdot 0.1 + 0.42 \cdot 0.5 \cdot 0.25 \cdot 0.9 \cdot 0.1$$

Distance of NFAs

Probabilistic distance of NFAs:

$$dist_P(\mathcal{A}, \mathcal{A}_{red}) = Pr_P(\mathcal{L}(\mathcal{A}) \underbrace{\bowtie}_{\text{symmetric difference}} \mathcal{L}(\mathcal{A}_{red}))$$

$$= Pr_P(\mathcal{L}(\mathcal{A})) + Pr_P(\mathcal{L}(\mathcal{A}_{red})) - 2Pr_P(\mathcal{A} \cap \mathcal{A}_{red})$$

Distance of NFAs

Probabilistic distance of NFAs:

$$dist_P(\mathcal{A}, \mathcal{A}_{red}) = Pr_P(\mathcal{L}(\mathcal{A}) \underbrace{\bowtie}_{\text{symmetric difference}} \mathcal{L}(\mathcal{A}_{red}))$$

$$= Pr_P(\mathcal{L}(\mathcal{A})) + Pr_P(\mathcal{L}(\mathcal{A}_{red})) - 2Pr_P(\mathcal{A} \cap \mathcal{A}_{red})$$

Theorem

*Computing $Pr_P(\mathcal{L}(\mathcal{A}))$ is **PSPACE**-complete.*
*If $\mathcal{A}$ is unambiguous, it is in **PTIME**.*

Distance of NFAs

Probabilistic distance of NFAs:

$$dist_P(\mathcal{A}, \mathcal{A}_{red}) = Pr_P(\mathcal{L}(\mathcal{A}) \underbrace{\bowtie}_{\text{symmetric difference}} \mathcal{L}(\mathcal{A}_{red}))$$

$$= Pr_P(\mathcal{L}(\mathcal{A})) + Pr_P(\mathcal{L}(\mathcal{A}_{red})) - 2Pr_P(\mathcal{A} \cap \mathcal{A}_{red})$$

Theorem

*Computing $Pr_P(\mathcal{L}(\mathcal{A}))$ is **PSPACE**-complete.*
*If $\mathcal{A}$ is unambiguous, it is in **PTIME**.*

Proof. **PSPACE**-hardness: reduction from NFA universality (**PSPACE**):

- let $\forall w \in \Sigma^* : Pr_P(w) > 0$
- check $Pr_P(\mathcal{L}(\mathcal{A})) = 1$

Distance of NFAs

Probabilistic distance of NFAs:

$$dist_P(\mathcal{A}, \mathcal{A}_{red}) = Pr_P(\mathcal{L}(\mathcal{A}) \underbrace{\bowtie}_{\text{symmetric difference}} \mathcal{L}(\mathcal{A}_{red}))$$

$$= Pr_P(\mathcal{L}(\mathcal{A})) + Pr_P(\mathcal{L}(\mathcal{A}_{red})) - 2Pr_P(\mathcal{A} \cap \mathcal{A}_{red})$$

Theorem

Computing $Pr_P(\mathcal{L}(\mathcal{A}))$ is ***PSPACE****-complete.*
If $\mathcal{A}$ is unambiguous, it is in ***PTIME****.*

Proof. **PSPACE**-hardness: reduction from NFA universality (**PSPACE**):

- let $\forall w \in \Sigma^* : Pr_P(w) > 0$
- check $Pr_P(\mathcal{L}(\mathcal{A})) = 1$

Upper bounds:

- **PTIME**: product of $\mathcal{A}$ and $P \rightsquigarrow$ system of linear equations
- **PSPACE**: on-the-fly determinize $\mathcal{A}$ $\times$ run $\uparrow$ (std. composition) □

Pr-driven NFA Reduction

Probability-driven NFA Reduction

- 2 optimization problems:

Pr-driven NFA Reduction

Probability-driven NFA Reduction

- 2 optimization problems:
  - size-driven: (n) $\mathcal{A} \rightsquigarrow \mathcal{A}_{red}$ s.t. $|\mathcal{A}_{red}| \leq n$ and $dist_P(\mathcal{A}, \mathcal{A}_{red})$ minimal

Pr-driven NFA Reduction

Probability-driven NFA Reduction

- 2 optimization problems:
  - size-driven: (n) $\mathcal{A} \rightsquigarrow \mathcal{A}_{red}$ s.t. $|\mathcal{A}_{red}| \leq n$ and $dist_P(\mathcal{A}, \mathcal{A}_{red})$ minimal
  - error-driven: (ϵ) $\mathcal{A} \rightsquigarrow \mathcal{A}_{red}$ s.t. $dist_P(\mathcal{A}, \mathcal{A}_{red}) \leq \epsilon$ and $|\mathcal{A}_{red}|$ minimal

Pr-driven NFA Reduction

Probability-driven NFA Reduction

- 2 optimization problems:
  - size-driven: (n) $\mathcal{A} \rightsquigarrow \mathcal{A}_{red}$ s.t. $|\mathcal{A}_{red}| \leq n$ and $dist_P(\mathcal{A}, \mathcal{A}_{red})$ minimal
  - error-driven: (ϵ) $\mathcal{A} \rightsquigarrow \mathcal{A}_{red}$ s.t. $dist_P(\mathcal{A}, \mathcal{A}_{red}) \leq \epsilon$ and $|\mathcal{A}_{red}|$ minimal

Theorem

Determining existence of $\mathcal{A}_{red}$ s.t. $dist_P(\mathcal{A}, \mathcal{A}_{red}) \leq \epsilon$ and $|\mathcal{A}_{red}| \leq n$ is ***PSPACE****-complete.*

- not easier than finding minimal NFA
- an enumerative algorithm $\rightsquigarrow$ not practical
- prob. (bi-)simulations don't work

Pr-driven NFA Reduction

Practical reductions:

- based on removing states and transitions

Pr-driven NFA Reduction

Practical reductions:

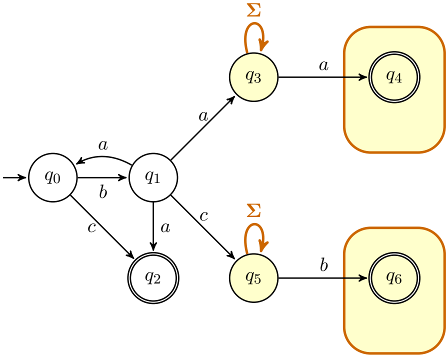
- based on removing states and transitions
- 2 approaches:

Pr-driven NFA Reduction

Practical reductions:

- based on removing states and transitions
- 2 approaches:

self-loop reduction

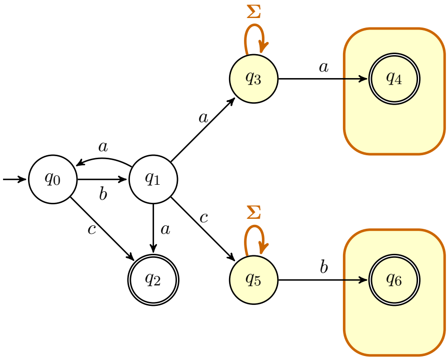


Pr-driven NFA Reduction

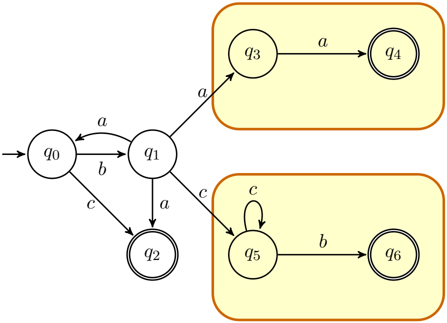
Practical reductions:

- based on removing states and transitions
- 2 approaches:

self-loop reduction



pruning reduction



Self-Loop Reduction

Self-Loop Reduction

- introduces self-loops $\Rightarrow$ over-approximating

Self-Loop Reduction

Self-Loop Reduction

- introduces self-loops $\Rightarrow$ over-approximating

Theorem

Given **n** *and* ϵ*, determining whether there exists* $\mathcal{A}_{red}$ *with* **n** *states and error* $\leq \epsilon$ *obtained from* $\mathcal{A}$ *by adding self-loops is* ***PSPACE****-complete.*

Self-Loop Reduction

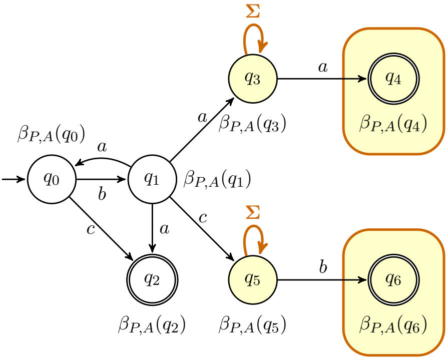
Self-Loop Reduction

- introduces self-loops ⇒ over-approximating

Theorem

Given **n** *and* ϵ*, determining whether there exists* $\mathcal{A}_{red}$ *with* **n** *states and error* $\leq \epsilon$ *obtained from* $\mathcal{A}$ *by adding self-loops is* ***PSPACE****-complete.*

- practical
- greedy algorithm to select states to add self-loops
- redundant states removed
- labelling — approximates the error



Pruning Reduction

Pruning Reduction:

- removes states ⇒ under-approximating

Pruning Reduction

Pruning Reduction:

- removes states $\Rightarrow$ under-approximating

Theorem

Given **n** *and* ϵ, *determining whether there exists* $\mathcal{A}_{red}$ *with* **n** *states and error* $\leq \epsilon$ *obtained from* $\mathcal{A}$ *by removing states is* ***PSPACE****-complete.*

Pruning Reduction

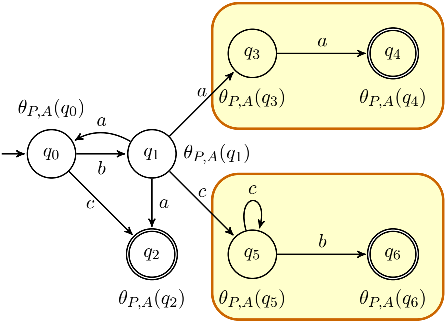
Pruning Reduction:

- removes states $\Rightarrow$ under-approximating

Theorem

Given **n** *and* ϵ, *determining whether there exists* $\mathcal{A}_{red}$ *with* **n** *states and error* $\leq \epsilon$ *obtained from* $\mathcal{A}$ *by removing states is* ***PSPACE***-*complete.*

- practical
- greedy algorithm to select states to remove
- labelling — approximates the error



Results

Results

- case studies from SNORT
  - targeting attacks over HTTP
  - self-loop reduction

Results

- case studies from SNORT
  - targeting attacks over HTTP
  - self-loop reduction
- model of network traffic — probabilistic automaton P_{HTTP}
  - structure constructed manually
  - probabilities learnt using real traffic (∼243 kpkt from ∼30 GiB)

Results

- case studies from SNORT
  - targeting attacks over HTTP
  - self-loop reduction
- model of network traffic — probabilistic automaton P_{HTTP}
  - structure constructed manually
  - probabilities learnt using real traffic (∼243 kpkt from ∼30 GiB)
- RABIT (R. Mayr) used for exact NFA reduction
  - simulation-based

Results

- case studies from SNORT
  - targeting attacks over HTTP
  - self-loop reduction
- model of network traffic — probabilistic automaton P_{HTTP}
  - structure constructed manually
  - probabilities learnt using real traffic (∼243 kpkt from ∼30 GiB)
- RABIT (R. Mayr) used for exact NFA reduction
  - simulation-based
- synthesis for Xilinx Virtex-7
  - reporting #LUTs (look-up tables)

Results

- case studies from SNORT
  - targeting attacks over HTTP
  - self-loop reduction
- model of network traffic — probabilistic automaton P_{HTTP}
  - structure constructed manually
  - probabilities learnt using real traffic (∼243 kpkt from ∼30 GiB)
- RABIT (R. Mayr) used for exact NFA reduction
  - simulation-based
- synthesis for Xilinx Virtex-7
  - reporting #LUTs (look-up tables)
- tool APPREAL
  - APProximate REduction of Automata and Languages
  - `https://github.com/vhavlena/appreal`

Results — case study 1

http-malicious.pcre

```
/^POST HTTP\/1\.[01]\r\n(\V+\r\n)*\r\n[\x00-\xff]*DROP TABLE/
/^HTTP\/1\.[01] 404[\x00-\xff]*(admin|wordpress)/
/^POST HTTP\/1\.[01]\r\n(\V+\r\n)*\r\n[\x00-\xff]*admin:admin/
/^POST HTTP\/1\.[01]\r\n(\V+\r\n)*\r\n[\x00-\xff]*admin:password/
/^POST HTTP\/1\.[01]\r\n(\V+\r\n)*\r\n[\x00-\xff]*YWRtaW46cGFzc3dvcmQ/
/^POST HTTP\/1\.[01]\r\n(\V+\r\n)*\r\n[\x00-\xff]*YWRtaW46YWRtaW4/
/^POST HTTP\/1\.[01]\r\n(\V+\r\n)*\r\n[\x00-\xff]*\/bin\/sh/
```

Before *Pr* reduction

- $|A_{\mathtt{mal}}| = 249$ states
- $|A^{\textsc{Red}}_{\mathtt{mal}}| = 98$ states
- $time(label) = 39$ s
- $time(\textsc{App}) < 1$ s
- $LUT(A^{\textsc{Red}}_{\mathtt{mal}}) = 382$

k	$\lvert A^{\textsc{App}}_{\mathtt{mal}}\rvert$	$\lvert A'_{\mathtt{mal}}\rvert$	Error label	Error P_{HTTP}	Error traffic	LUTs
0.1	9	9	0.0704	0.0704	0.0685	—
0.2	19	19	0.0677	0.0677	0.0648	—
0.3	29	26	0.0279	0.0278	0.0598	154
0.4	39	36	0.0032	0.0032	0.0008	—
0.5	49	44	2.8e-05	2.8e-05	4.1e-06	—
0.6	58	49	8.7e-08	8.7e-08	0.0	224
0.8	78	75	2.4e-17	2.4e-17	0.0	297

Results — case study 2

http-attacks.pcre

```
/calendar(|[-_]admin)\.pl[\x00-\xff]*/Ui
/db4web_c(\.exe)?\/.*(\.\.[\#\/]|[a-z]\:)[\x00-\xff]*/smiU
/evtdump\x3f.*?\x2525[^\x20]*?\x20HTTP[\x00-\xff]*/i
/instancename=[^&\x3b\r\n]{10}[\x00-\xff]*/smi
/itemid=\d*[^\d\&\;\r\n][\x00-\xff]*/i
/^GET\s+[^\x20]*\x2Ewm[zd][\x00-\xff]*/smi
/mstshash\s*\x3d\s*Administr[\x00-\xff]*/smi
/SILC\x2d\d\x2e\d[\x00-\xff]*/smi
```

Before *Pr* reduction

- $|A_{\mathtt{att}}| = 142$ states
- $|A^{\mathrm{RED}}_{\mathtt{att}}| = 112$ states
- $time(label) = 28$ min
- $time(\mathrm{APP}) \approx 1$ s

k	$\lvert A^{\mathrm{APP}}_{\mathtt{att}}\rvert$	$\lvert A'_{\mathtt{att}}\rvert$	Error label	Error P_{HTTP}	Error traffic
0.2	22	14	1.0	0.8341	0.2313
0.3	33	24	0.081	0.0770	0.0067
0.4	44	37	0.0005	0.0005	0.0010
0.5	56	49	3.3e-06	3.3e-06	0.0010
0.6	67	61	1.2e-09	1.2e-09	8.7e-05
0.7	78	72	4.8e-12	4.8e-12	1.2e-05
0.9	100	93	3.7e-16	1.1e-15	0.0

Results — case study 3

http-backdoor.pcre

```
/000File\s+is\s+executed\x2E\x2E\x2E/smi
/^000Ok\s+echter\s+server\s+\?/smi
/^001\xACOptix\s+Pro\s+v\d+\x2E\d+\s+Connected\s+Successfully\x21/smi
/^100013Agentsvr\x5E\x5EMerlin/smi
/^666\d+\xFF\d+\xFF\d+\xFF\d+\xFF\d+\xFF\d+\xFF\d+\xFF/smi
/^A-311 Death welcome/smi
/^answer\x00{6}NetControl\x2EServer\s+\d+\x2E\d+\s+\x22The\s+UNSEEN\x22\s+
                    [...  42 more lines ...]
```

Before *Pr* reduction

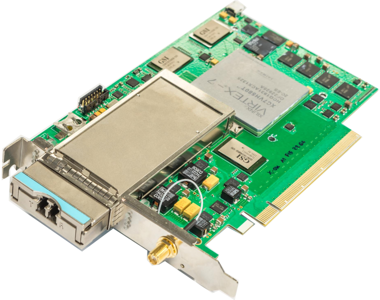
- $|A_{\mathrm{bd}}| = 1{,}352$ states
- $|A^{\mathrm{RED}}_{\mathrm{bd}}| = ??$ states
- $time(label) = 20$ min
- $time(\mathrm{APP}) \approx 1.5$ min
- $LUT(A^{\mathrm{RED}}_{\mathrm{mal}}) = 2{,}266$

k	$\lvert A^{\mathrm{APP}}_{\mathrm{bd}}\rvert$	$\lvert A'_{\mathrm{bd}}\rvert$	Error label	Error traffic	LUTs
0.1	135	8	1.0	0.997	202
0.2	270	111	0.0012	0.0631	579
0.3	405	233	3.4e-08	0.0003	894
0.4	540	351	1.0e-12	0.0003	1063
0.5	676	473	1.2e-17	0.0	1249
0.7	946	739	8.3e-30	0.0	1735
0.9	1216	983	1.3e-52	0.0	2033

Results — case study 4

Real impact on COMBO-100G (Xilinx Virtex-7 H580T)

- `http-malicious.pcre`
  - $LUT(A^{RED}_{\texttt{mal}}) = 382$
- `http-backdoor.pcre`
  - $LUT(A^{RED}_{\texttt{bd}}) = 2{,}266$
- available LUTs = 15,000



Speed	LUTs	$A^{RED}_{\texttt{mal}}$ speed	$A'_{\texttt{mal}}$ error	$A^{RED}_{\texttt{bd}}$ speed	$A'_{\texttt{bd}}$ error
100 Gbps	937	100 Gbps	0	38.4 Gbps	3.4e-18
400 Gbps	238	250 Gbps	8.7e-8	38.4 Gbps	1

Future Work

Future work:

- learning of prob. automaton
- different automaton models (e.g. delayed input DFA)
- better cost function

Summary

- reduction of nondeterministic finite automata (NFAs)
- the reduction does NOT preserve language
- BUT guarantees maximum error
- w.r.t. a probabilistic distribution
- application in high-speed network intrusion detection
- obtained significant speed improvement w/ small error

Summary

- reduction of nondeterministic finite automata (NFAs)
- the reduction does NOT preserve language
- BUT guarantees maximum error
- w.r.t. a probabilistic distribution
- application in high-speed network intrusion detection
- obtained significant speed improvement w/ small error

THANK YOU!