Limits of Learning-based Signature Generation with Adversaries

Shobha Venkataraman, Carnegie Mellon University
Avrim Blum, Carnegie Mellon University
Dawn Song, University of California, Berkeley

Signatures

- Signature: function that acts as a classifier
  - Input: byte string
  - Output: Is byte string **malicious** or **benign**?

- e.g., signature for Lion worm:

  "\xFF\xBF" && "\x00\x00\FA"

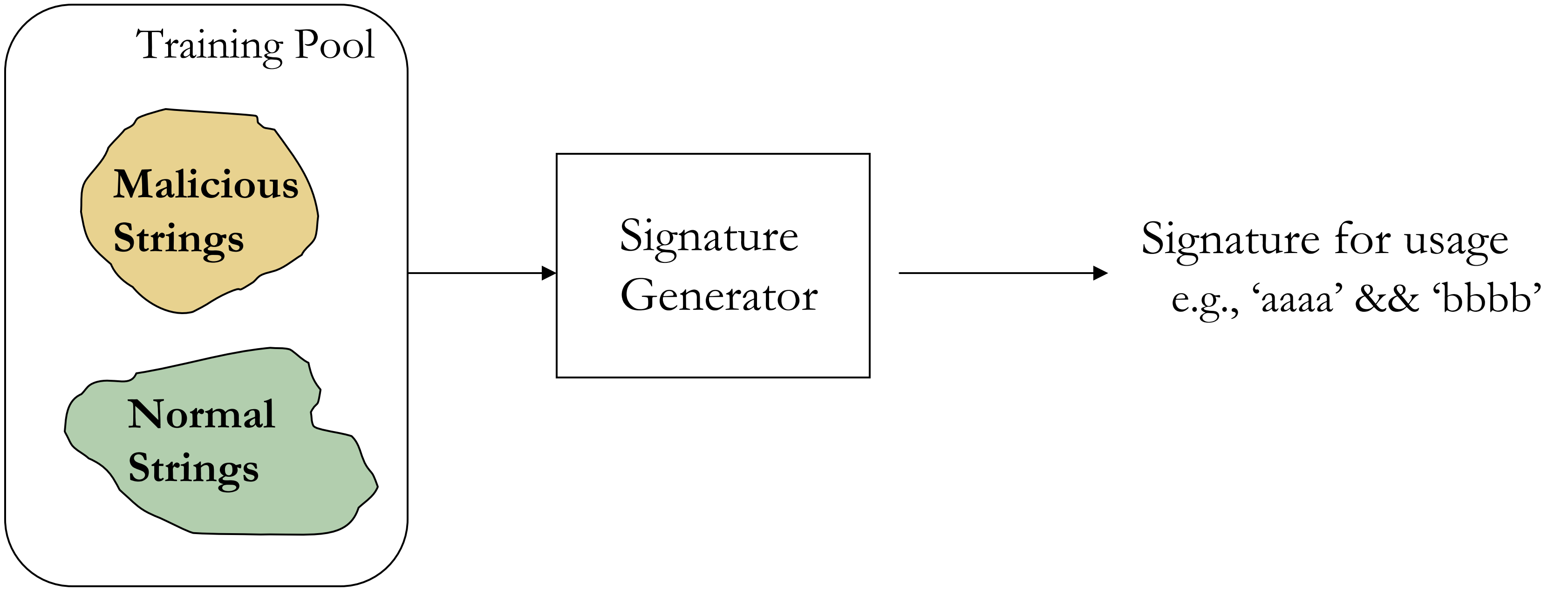
  "aaaa" "bbbb"

  - If both present in byte string, MALICIOUS
  - If either one **not** present, BENIGN

- This talk: focus on signatures that are sets of byte patterns
  - i.e., signature is conjunction of byte patterns
  - Our results for conjunctions imply results for more complex functions, e.g. regexp of byte patterns

Automatic Signature Generation

- Generating signatures automatically is important:
  - Signatures need to be generated quickly
  - Manual analysis slow and error-prone
- Pattern-extraction techniques for generating signatures



History of Pattern-Extraction Techniques

Signature Generation Systems

Evasion Techniques

2003

Earlybird, Autograph, Honeycomb
[SEVS] [KK] [KC]

Polymorphic worms

2005

Polygraph
[NKS]

Hamsa
[LSCCK]

2007

Anagram
[WPS]

…

Malicious Noise Injection
[PDLFS]

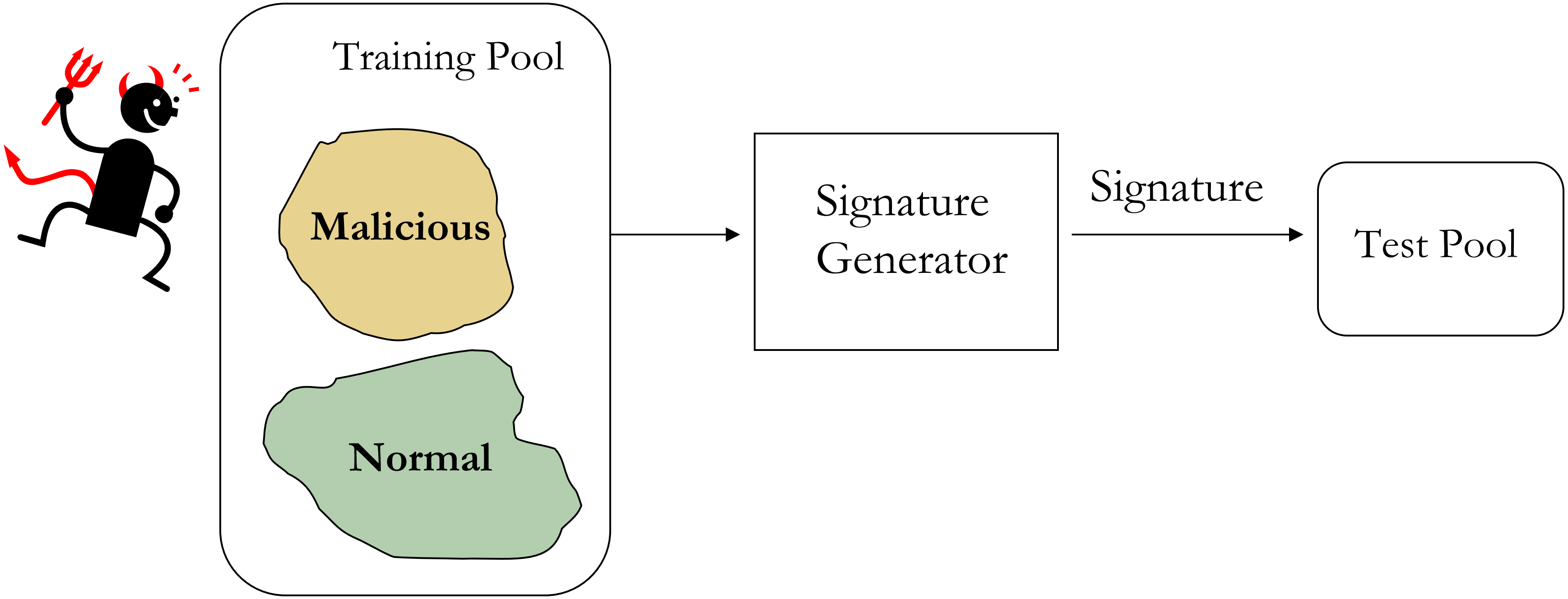
Paragraph [NKS]

Allergy attacks [CM]

…

Our Work: Lower bounds on how quickly ALL such algorithms converge to signature in presence of adversaries

Learning-based Signature Generation



Signature generator's goal:
Learn as quickly as possible

Adversary's goal:
Force as many errors as possible

Our Contributions

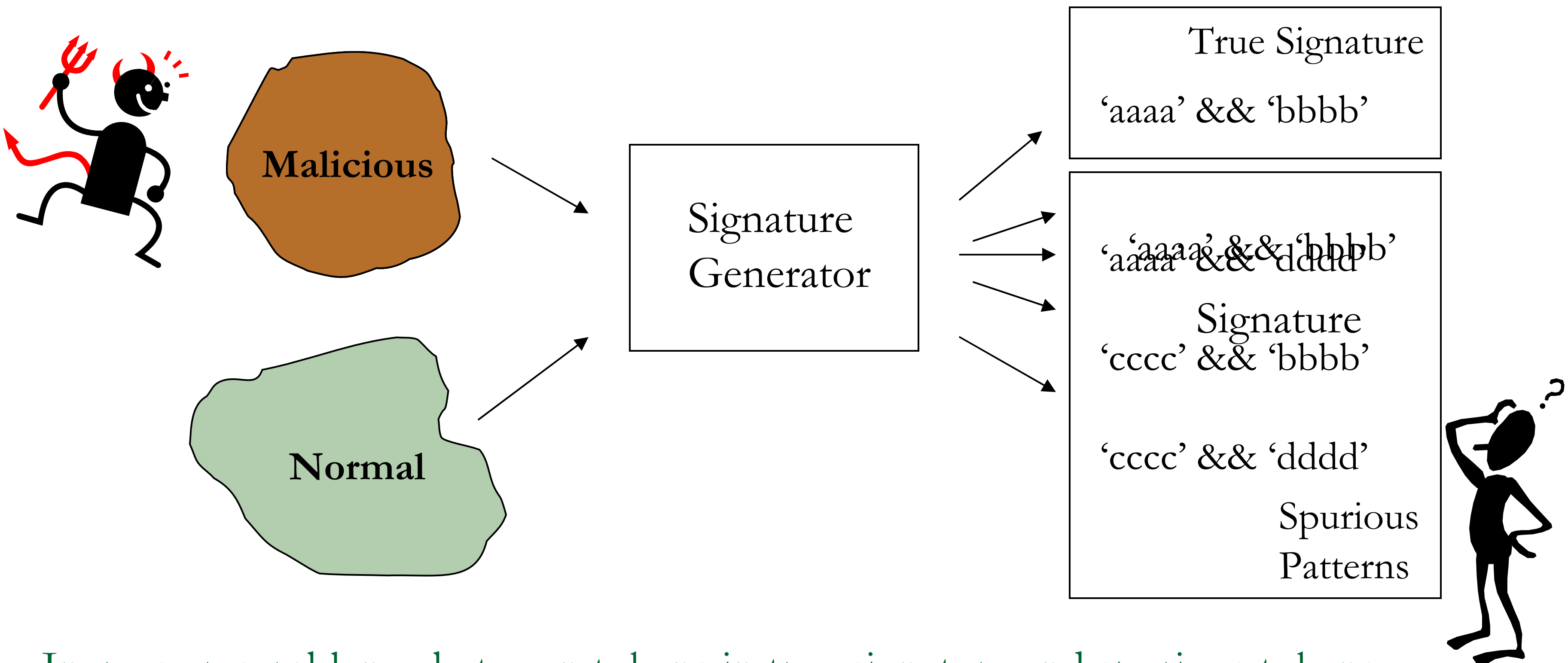
Formalize a framework for analyzing performance of pattern-extraction algorithms under adversarial evasion

- Show fundamental limits on accuracy of pattern-extraction algorithms with adversarial evasion
  - Generalize earlier work (e.g.,[FDLFS],[NKS,[CM]]) focused on individual systems
- Analyze when fundamental limits are weakened
  - Kind of exploits for which pattern-extraction algorithms may work
- Applies to other learning-based algorithms using similar adversarial information (e.g., COVERS[LS])

Outline

- Introduction
- **Formalizing Adversarial Evasion**
- Learning Framework
- Results
- Conclusions

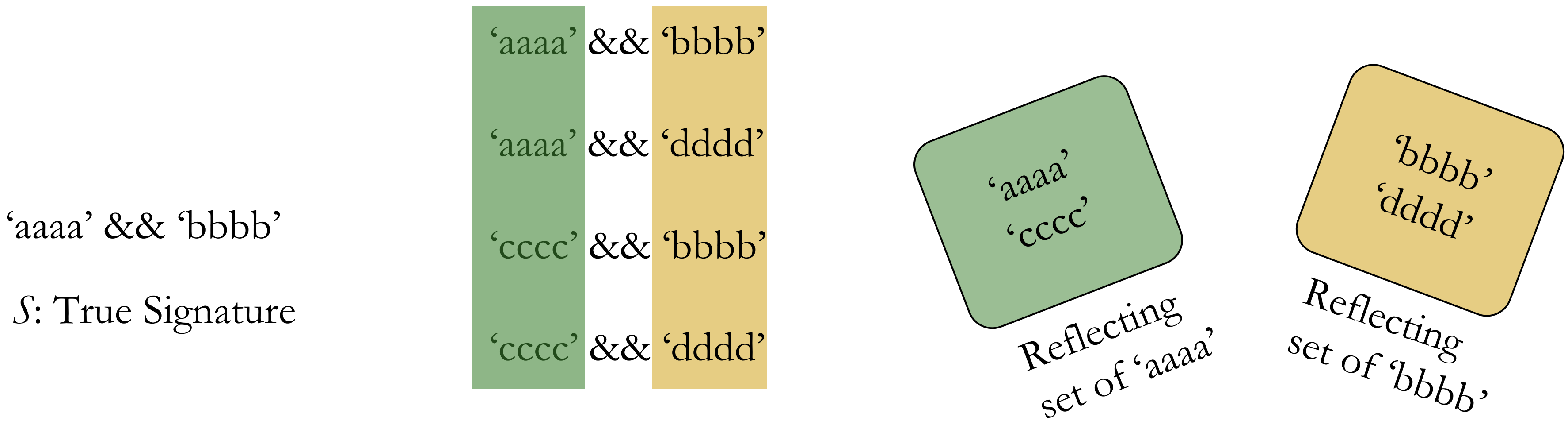
Strategy for Adversarial Evasion



Increase resemblance between tokens in true signature and spurious tokens

- e.g. can add infrequent tokens (i.e, red herrings [NKS]), change token distributions (i.e., pool poisoning [NKS]), mislabel samples (i.e, noise-injection [PDLFS])
- Could generate high false positives or high false negatives

Definition: Reflecting Set



T: Set of Potential Signatures

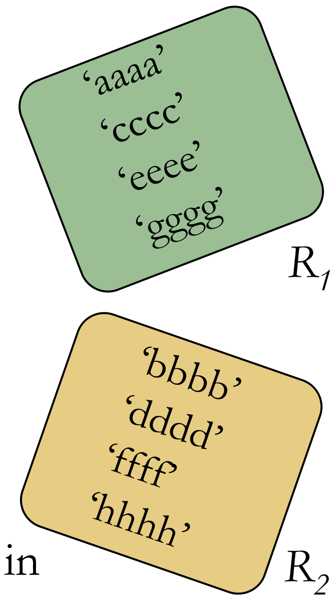
Reflecting Sets: Sets of Resembling Tokens

- **Critical token**: token in true signature *S*. e.g., 'aaaa', 'bbbb'
- **Reflecting set** of a critical token *i* for a signature generator:
  All tokens as likely to be in *S* as critical token *i*, for current signature-generator
  e.g., Reflecting set for 'aaaa': 'aaaa', 'cccc'

Reflecting Sets and Algorithms

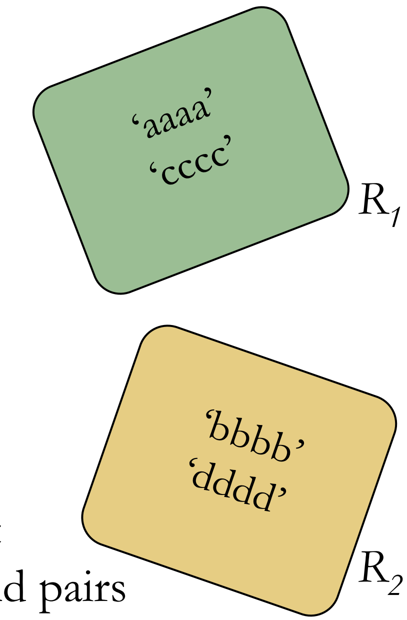
Specific to the family of algorithms under consideration

Signature Generator 1



e.g., coarse-grained
All tokens infrequent in normal traffic, say, first-order statistics

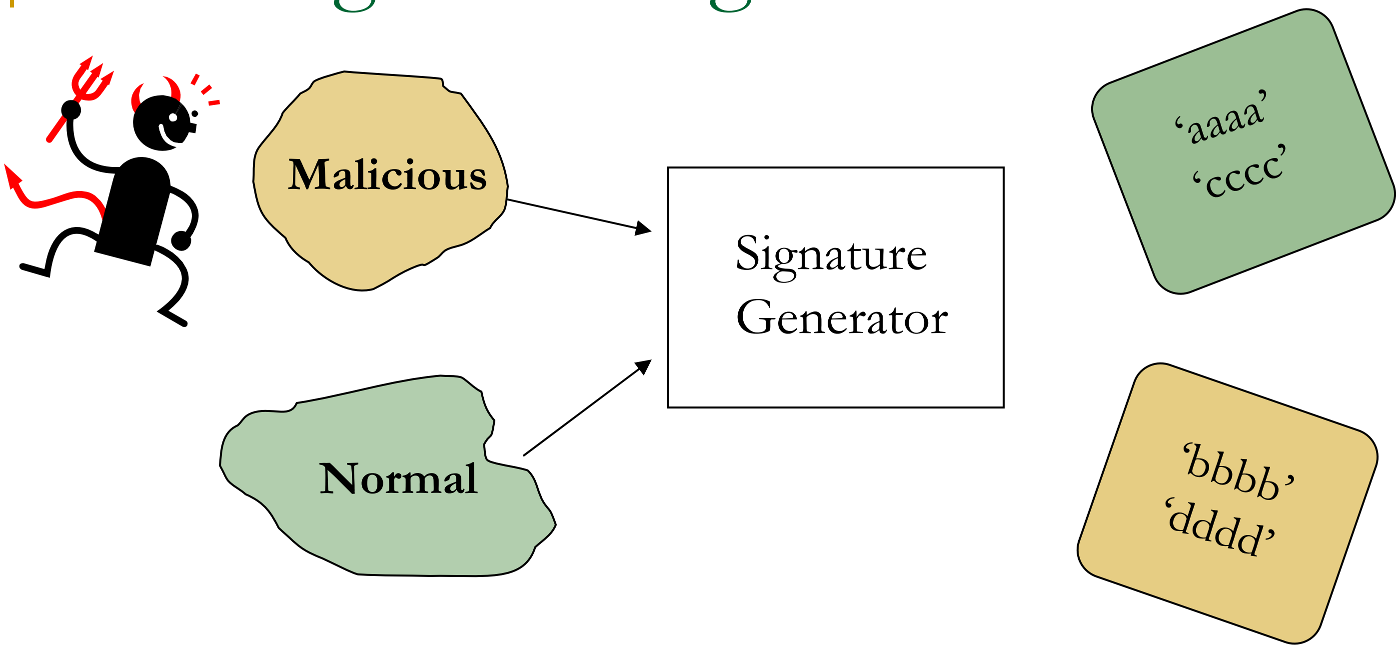
Signature Generator 2



e.g. fine-grained
All tokens such that individual tokens and pairs of tokens infrequent

By definition of reflecting set, *to signature-generation algorithm*, true signature appears to be drawn at random from R_1 x R_2

Learning-based Signature Generation

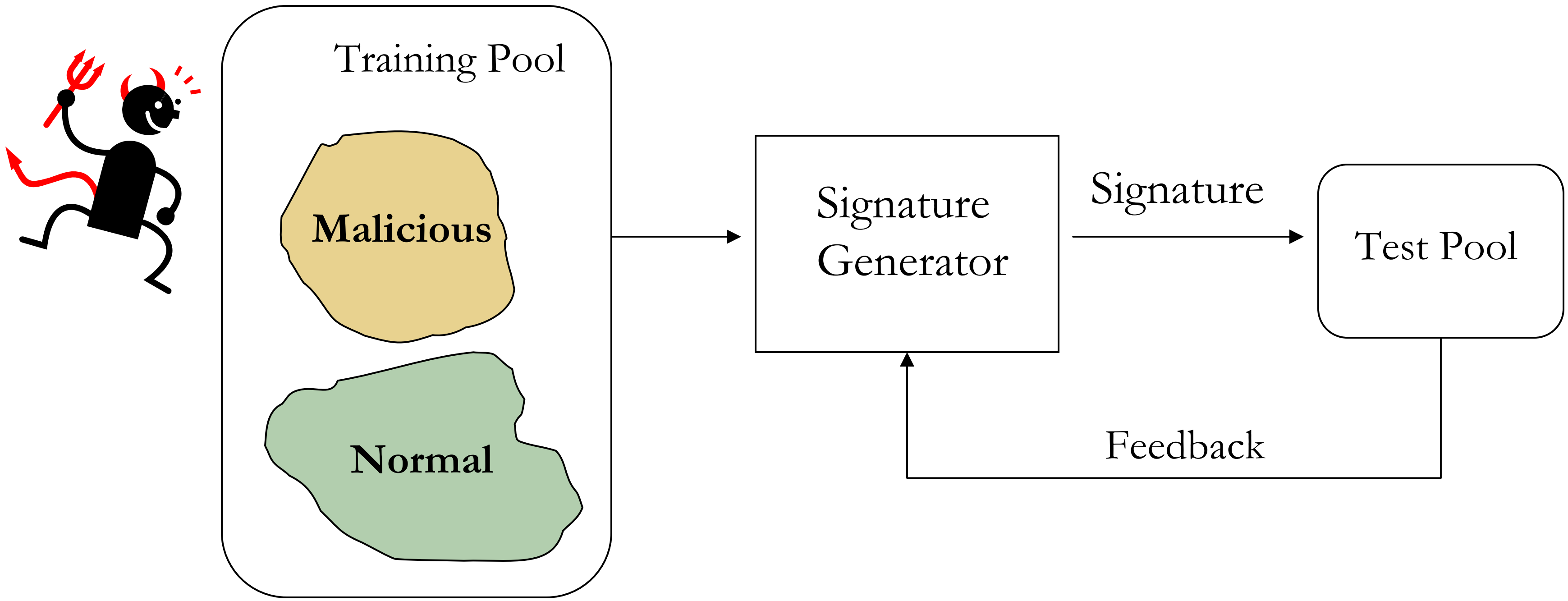


- Problem: Learning a signature when a malicious adversary constructs reflecting sets for each critical token
- Lower bounds depend on size of reflecting set:
  - power of adversary,
  - nature of exploit,
  - algorithms used for signature generation

Outline

- Introduction
- Formalizing Adversarial Evasion
- **Learning Framework**
- Results
- Conclusions

Framework: Online Learning Model



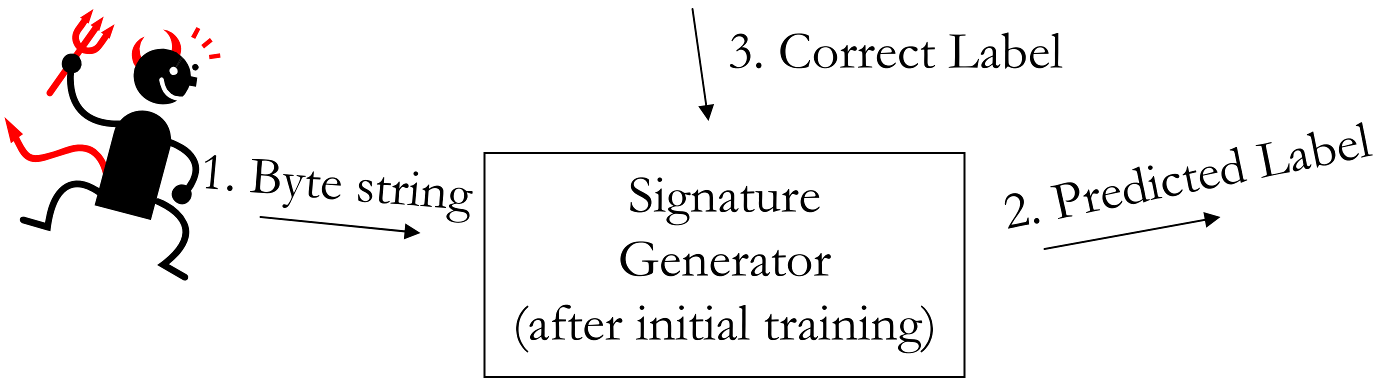
Signature generator's goal:
Learn as quickly as possible
- Optimal to update with new information in test pool

Adversary's goal:
Force as many errors as possible
- Optimal to present only one new sample before each update

Equivalent to the **mistake-bound model of online learning** [LW]

Learning Framework: Problem

Mistake-bound model of learning



- Notation:
  - n: number of critical tokens
  - r: size of reflecting set for each critical token
- Assumption: true signature is a **conjunction** of tokens
  - Set of all potential signatures: r^n
- Goal: find true signature from r^n potential signatures

  minimize mistakes in prediction while learning true signature

Learning Framework: Assumptions

- **Signature Generation Algorithms Used**
  - Algorithm can learn *any* function for signature

    Not necessary to learn only conjunctions

- **Adversary Knowledge**
  - Algorithms/systems/features used to generate signature
  - Does not necessarily know how system/algorithm is tuned

- **No Mislabeled Samples**
  - No mislabeling, either due to noise or malicious injection

    e.g., use host-monitoring techniques[NS] to achieve this

Outline

- Introduction
- Formalizing Adversarial Evasion
- Learning Framework
- **Results:**
  - **General Adversarial Model**
  - Can General Bounds be Improved?
- Conclusions

Deterministic Algorithms

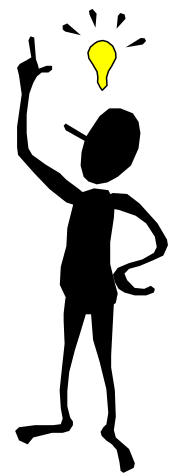
Theorem: For any **deterministic** algorithm, there exists a sequence of samples such that the algorithm is forced to make at least ***n log r*** mistakes.

Additionally, there exists an algorithm (Winnow) that can achieve a mistake-bound of $n(\log r + \log n)$

Practical Implication:

For arbitrary exploits, any pattern-extraction algorithm can be forced into making a number of mistakes:

- even if extremely sophisticated pattern-extraction algorithms are used
- even if all labels are accurate, e.g., if TaintCheck [NS] is used



Randomized Algorithms

Theorem: For any **randomized** algorithm, there exists a sequence of samples such that the algorithm is forced to make at least ***½ n log r*** mistakes in expectation.

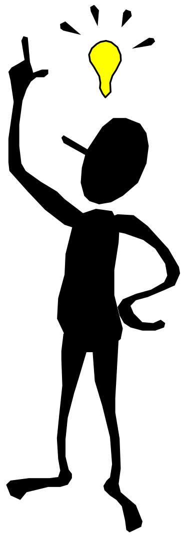
Practical Implication:

For arbitrary exploits, any pattern-extraction algorithm can be forced into making a number of mistakes:

- even if extremely sophisticated pattern-extraction algorithms are used
- even if all labels are accurate (e.g., if TaintCheck [NS] is used)
- **even if the algorithm is randomized**

One-Sided Error: False Positives

Theorem: Let $t < n$. Any algorithm forced to have **fewer than *t* false positives** can be forced to make at least ***(n – t) (r – 1)*** mistakes on malicious samples.



Practical Implication:

Algorithms that are allowed to have few false positives make significantly many more mistakes than the general algorithms

e.g., at $t = 0$, bounded false positives: $n(r - 1)$

general case: $n \log r$

One-Sided Error: False Negatives

Theorem: Let $t < n$. Any algorithm forced to have **fewer than *t* false negatives** can be forced to make at least $r^{n/(t+1)} - 1$ mistakes on non-malicious samples.

Practical Implication:

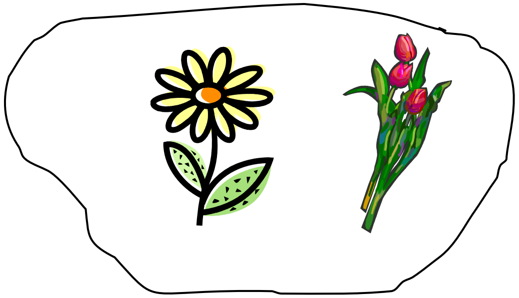
Algorithms allowed to have bounded false negatives have ***far*** worse bounds than general algorithms

e.g., at $t = 0$, bounded false negatives: $r^n - 1$

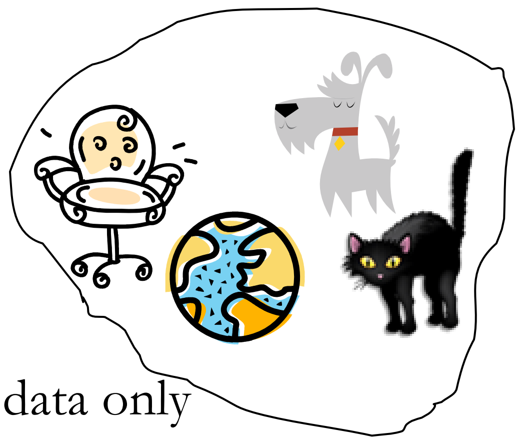
general algorithms: $n \log r$

Different Bounds for False Positives & Negatives!

e.g. Learning: What is a flower?



Positive data only



Negative data only

- Bounded false positives: $\Omega((r(n-t))$
  - learning from positive data only
    - No mistakes allowed on negatives
    - Adversary forces mistakes with positives
- Bounded false negatives: $\Omega(r^{n/t+1})$
  - learning from negative data only
    - No mistakes allowed on positives
    - Adversary forces mistakes with negatives
- Much more "information" about signature in a malicious sample

Outline

- Introduction
- Formalizing Adversarial Evasion
- Learning Framework
- **Results:**
  - General Adversarial Model
  - **Can General Bounds be Improved?**
- Conclusions

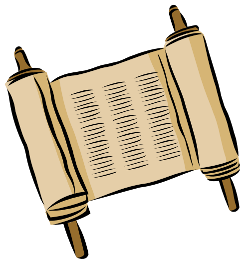
Can General Bounds be Improved?

- Consider Relaxed Problem:
  - Requirement: Classify correctly only
    - Malicious packets
    - Non-malicious packets regularly present in normal traffic
  - Classification does NOT have to match true signature on rest

- Characterize "gap" between malicious & normal traffic
  - **Overlap-ratio** *d*: Of tokens in true signature, fraction that appear together in normal traffic.

    e.g., signature has 10 tokens, but only 5 appear together in normal traffic: $d = 0.5$
  - Bounds are a function of overlap-ratio

Lower bounds with Gaps in Traffic



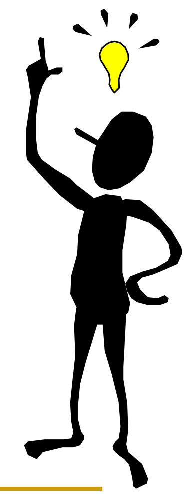
Theorem: Let $d < 1$. For a class of functions called linear separators, any deterministic algorithms can be forced to make $\boldsymbol{log_{1/d} r}$ mistakes, and any randomized algorithm can be forced to make in expectation, $\boldsymbol{\frac{1}{4} \, log_{1/d}\, r}$ mistakes.

As d approaches $\frac{n-1}{n}$, $log_{1/d} r$ approaches $n \, log \, r$!

Practical Implication:

Pattern-extraction algorithms may work for exploits if:

- signatures overlap very little with normal traffic
- algorithm is given few (or no) mislabeled samples



Related Work

- Learning-based signature-generation algorithms:

  Honeycomb[KC03], Earlybird [SEVS04], Autograph[KK04], Polygraph[NKS05], COVERS[LS06], Hamsa[LSCCK06], Anagram[WPS06]

- Evasions:

  [PDLFS06], [NKS06],[CM07],[GBV07]

- Adversarial Learning:
  - Closely Related: [Angluin88],[Littlestone88]
  - Others: [A97][ML93],[LM05],[BEK97] ,[DDMSV04]

Conclusions

Formalize a framework for analyzing performance of pattern-extraction algorithms under adversarial evasion

- Show fundamental limits on accuracy of pattern-extraction algorithms with adversarial evasion
  - Generalize earlier work focusing on individual systems
- Analyze when fundamental limits are weakened
  - Kind of exploits for which pattern-extraction algorithms may work

Thank you!

Comparison with Existing Techniques

Form of True Signature: Conjunction

- Simplifying assumption: true signature is a conjunction
  - E.g.
- Motivation:
  - Earlier experimental work shows conjunctions to be useful signatures on traffic traces
  - Lower bounds for conjunctions => lower bounds for more complex functions (e.g., regexp

Why do our bounds eventually converge to the right answer?

- Strong model for learning
  - Every mistake gains information: draw hypercube
  - Adversary not allowed to change
  - Algorithm is allowed to change
  - => Finite number of mistakes before convergence

- Change any of these, never converge
  - Maybe use algorithms designed for adversarial environments (with this kind of adversarial bounds)

Lower Bounds with Gaps in Traffic

- Measuring the Gap in Traffic:

  Overlap-ratio d: Of tokens in the true signature, fraction that appear together in normal traffic.

  e.g., true signature has 10 tokens, but only 5 appear together in normal traffic: $d = 0.5$

- Lower bounds are representation-dependent, when $d < 1$.
  - Algorithms learning linear separators: $\log_{1/d} k$

    (Linear weighted function of attributes)

- Pattern-extraction algorithms may work for exploits whose signatures overlap very little with normal traffic, with host-monitoring techniques
  - Representation-dependent lower bounds that are much weaker

Lower Bounds with Gaps in Traffic

- Lower bounds are representation-dependent, when $d < 1$.
  - Algorithms learning linear separators: $log_{1/d} k$

    (Linear weighted function of attributes)

- Pattern-extraction algorithms may work for exploits whose signatures overlap very little with normal traffic, with host-monitoring techniques
  - Representation-dependent lower bounds that are much weaker

Practical Implications

- For arbitrary exploits, any pattern-extraction algorithm can be forced into making a large number of mistakes, with common assumptions:
  - even if the algorithm is randomized
  - even if host-monitoring techniques are used, to avoid noise in labels
  - even if arbitrarily complex representations of signatures are allowed
- Existing research demonstrates feasibility of attacks on real systems; our results generalize to all systems that use similar properties of traffic.
- Algorithms that tolerate only one-sided error are significantly easier to manipulate by the adversary.
- Pattern-extraction algorithms may work for exploits whose signatures overlap very little with normal traffic, with host-monitoring techniques
  - Weaker lower bounds
  - Bounds depend on complexity of signature used by learning algorithm

Formal Definition of Reflecting Set?

When might signature-generation work?

- When the attacker cannot find reflecting set
  - “gaps” in traffic mean that

Summary

- Table
- Discussion: Notice they eventually converge

Finding Reflecting Sets

- Exist for current generations of pattern-extraction systems
  - Learning from adversarially-generated features that can be manipulated
  - All attributes in reflecting set [do not need to have identical statistics] Sufficient to bias away from true signature.

- Likely to exist for algorithms using traffic statistics of normal and malicious traffic
  - Heavy-tailed nature of traffic patterns (e.g., polymorphic blending attacks illustrate similar behaviour)

Learning Framework: Problem (II)

- Assumption: True signature is a Conjunction of tokens
  - Lower bounds for conjunctions imply lower bounds for more complex functions
  - Common systems have signatures as conjunctions
  - Set of all potential signatures: n^k
- Goal: learn true signature from n^k possible signatures
  - Identify n tokens that constitute true signature
  - **Lower bounds** on the mistakes that can be forced by an adversary

Can General Bounds be Improved?

- Do not always need to classify *all* packets correctly
  - Only need to classify correctly:
    - Malicious packets
    - Non-malicious packets regularly present in normal traffic
  - Classification does not have to match target signature on others

Exploit Gaps in traffic

  - Measure how close malicious traffic is to normal traffic
    - Measure should not be subject to adversarial manipulation
  - Bounds are a function of this measure

Generating Signatures Automatically

- Generating signatures automatically is important:
  - Signatures need to be generated quickly
  - Manual analysis slow and error-prone
- Pattern-extraction techniques for signature-generation