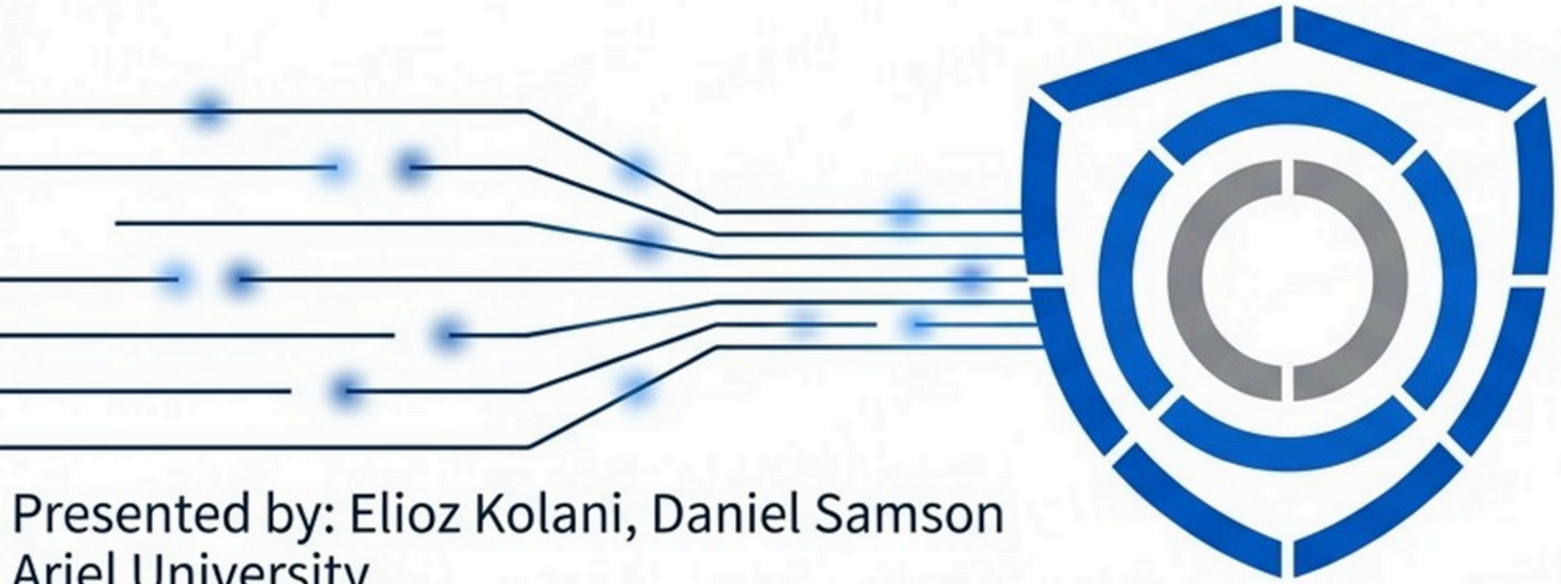
Detecting Malicious URLs

From a Lightweight Classifier to a Scalable, Distributed System



Presented by: Elioz Kolani, Daniel Samson
Ariel University
End-to-End Project in Cyber Attack Detection Course
January 2025

The Challenge: Real-Time Detection in a High-Stakes Environment

The goal is to detect malicious URLs immediately before a user clicks.

Obfuscation

Obfuscation

URLs are short, noisy, and easily hidden. Attackers constantly change patterns to evade static lists.

Latency

Latency

Decisions must happen in milliseconds. If the system is slow, the user experience breaks.

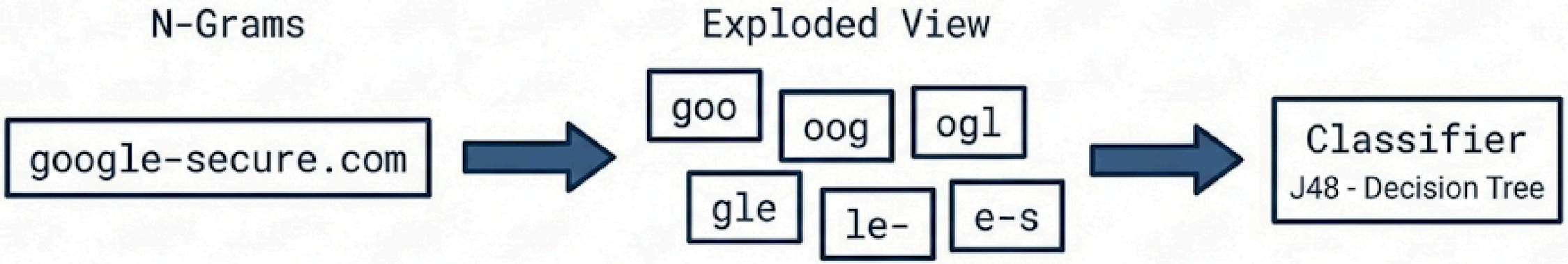
Imbalance

Imbalance

High cost of missed attacks compromises security, but blocking safe sites destroys usability.

First Article: LEXICAL BASED METHOD.

Daeef et al. (2016)



Mechanism

Character N-grams (n=1..4) converted to frequency vectors.

Value

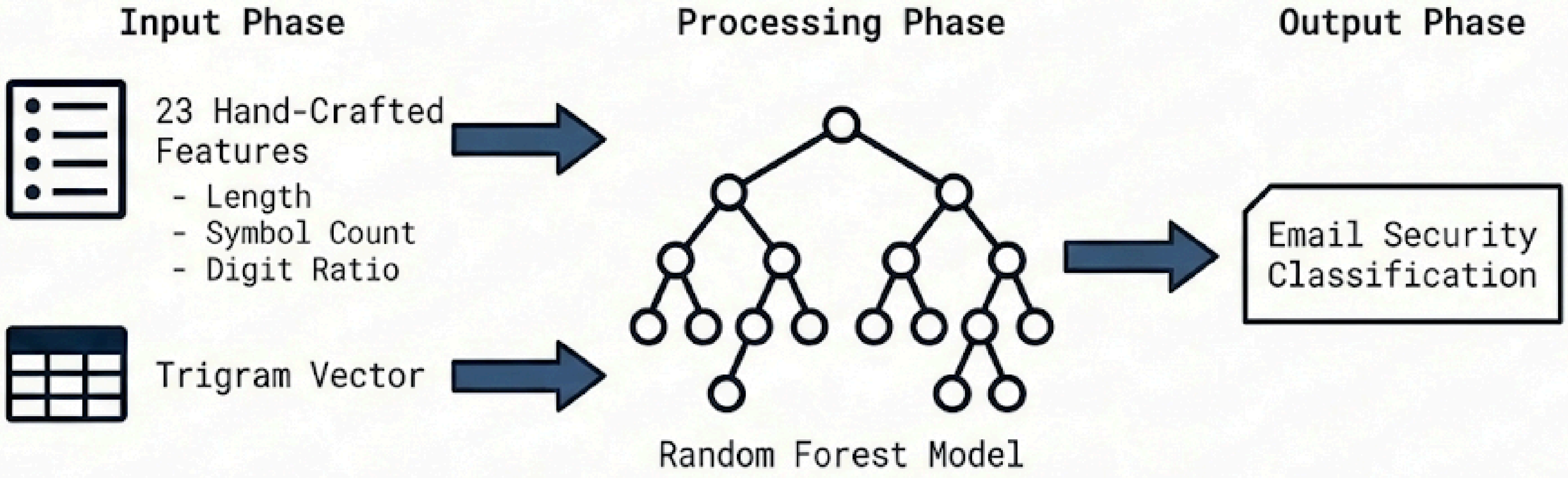
High speed. No page download required.

Limitation

Limited to Phishing contexts. Blind to structural anomalies.

Second Article:LEXICAL FEATURES FOR MALICIOUS URL

Joshi et al. (2019)



Key Takeaway: Hybrid representations (Structure + Text) provide a scalable baseline.

Analyzing the Foundation: Pros & Cons

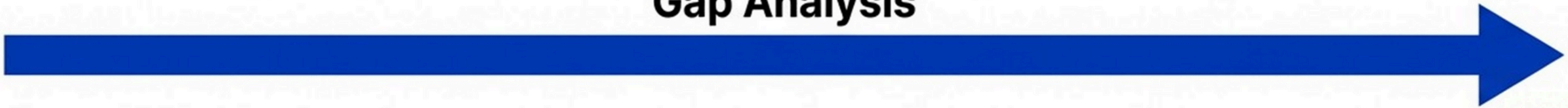
What They Did Well (The Pros)

- **Lexical Focus**: >90% accuracy without dangerous content downloads.
- **Feature Engineering**: Validated utility of N-grams and Hybrid features.
- **Efficiency**: Low latency suitable for real-time applications.

Where They Failed (The Cons)

- **Binary Scope**: Limited to Safe vs. Unsafe (`No Multi-class`).
- **No Calibration**: Overconfident models lacking probability scores.
- **Blind Spots**: Vulnerable to URL shortening services.
- **Offline Only**: Not designed as distributed production systems.

Gap Analysis

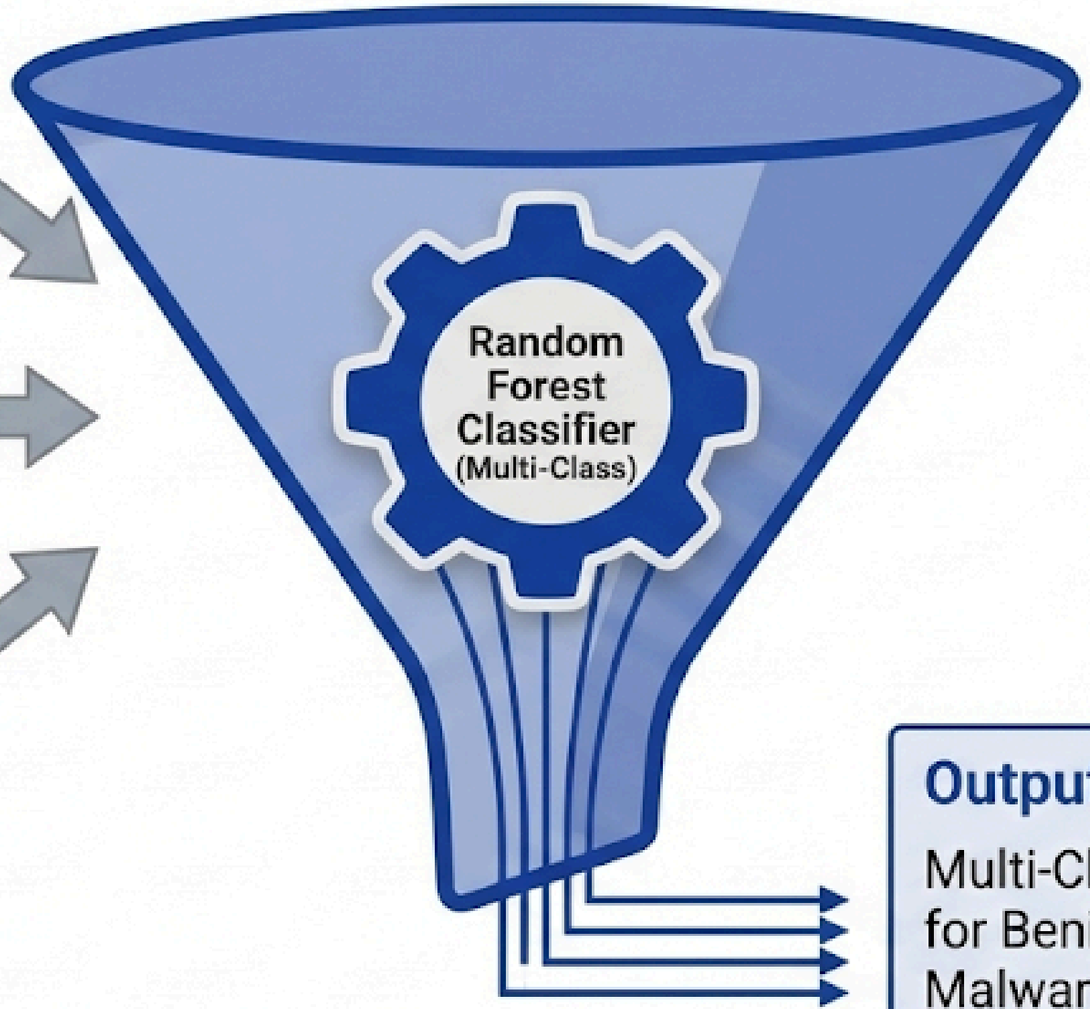


Our Approach to Model Improvement

Lexical & Statistical
(length, entropy, keywords)

Hashed Character Trigrams
(fixed-size buckets)

LM-Inspired Hashing
(host, path, query parts)



Output Classification:

Multi-Class predictions for Benign, Defacement, Malware, Phishing.

Model Creation Process

Evolution & Improvement

WE KEPT:

- Lexical Features System
- Trigram Hashing Method
- Random Forest (Stable)

WE LEFT BEHIND:

- Old Algorithms (e.g., J48)
- Local Work (Single Script)

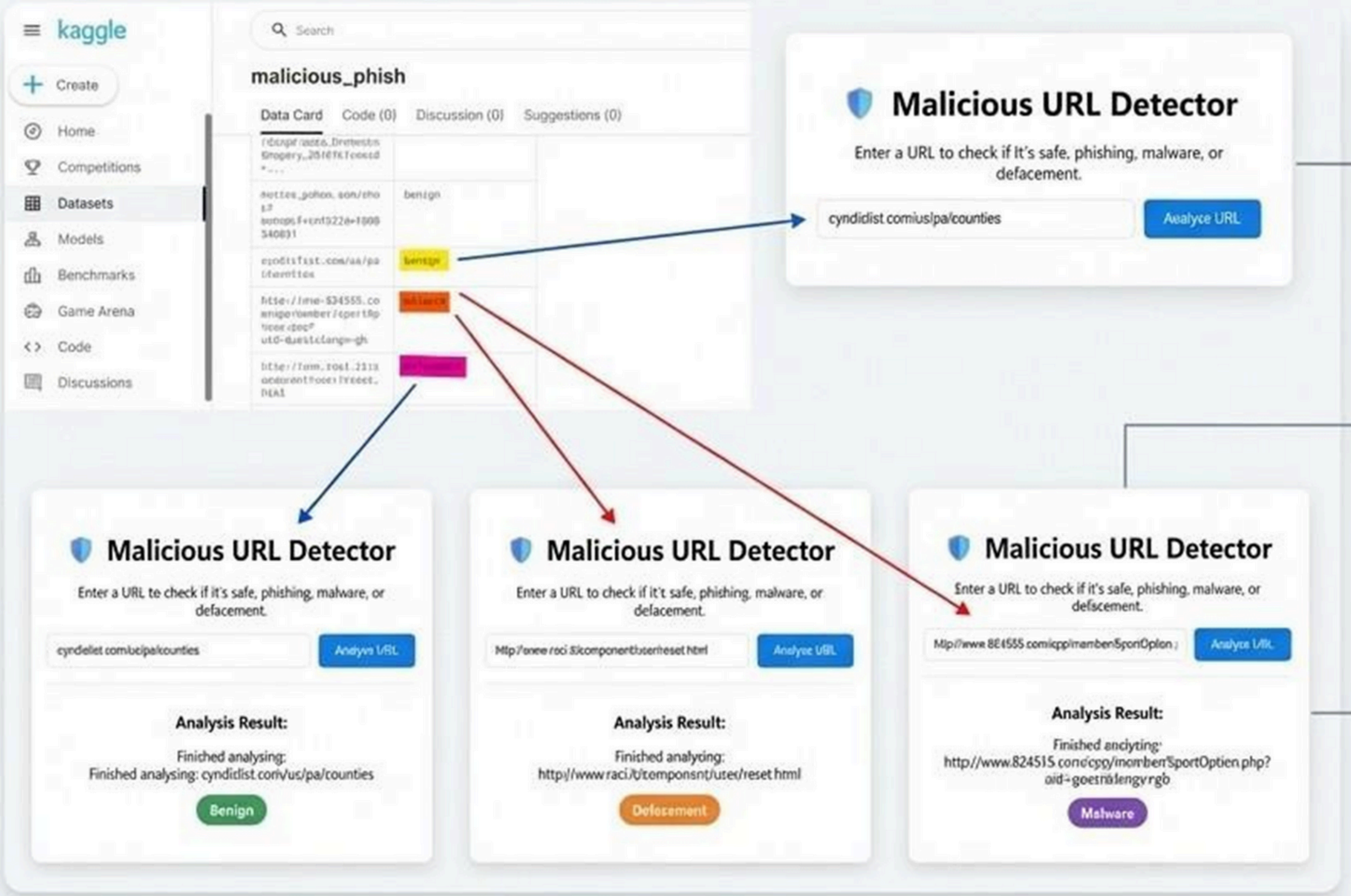
TO: Distributed Docker System

```
=== Evaluation (Confusion Matrix Results) ===
              precision    recall  f1-score   support

   benign 0       0.9591    0.9812    0.9700     85621
defacement 1      0.9761    0.9686    0.9723     19292
   malware 2      0.9953    0.9482    0.9712      6504
  phishing 3      0.8959    0.8238    0.8583     18822

    accuracy                         0.9549    130239
   macro avg      0.9566    0.9304    0.9430    130239
weighted avg      0.9543    0.9549    0.9543    130239
```

Proof: The System Works End-to-End

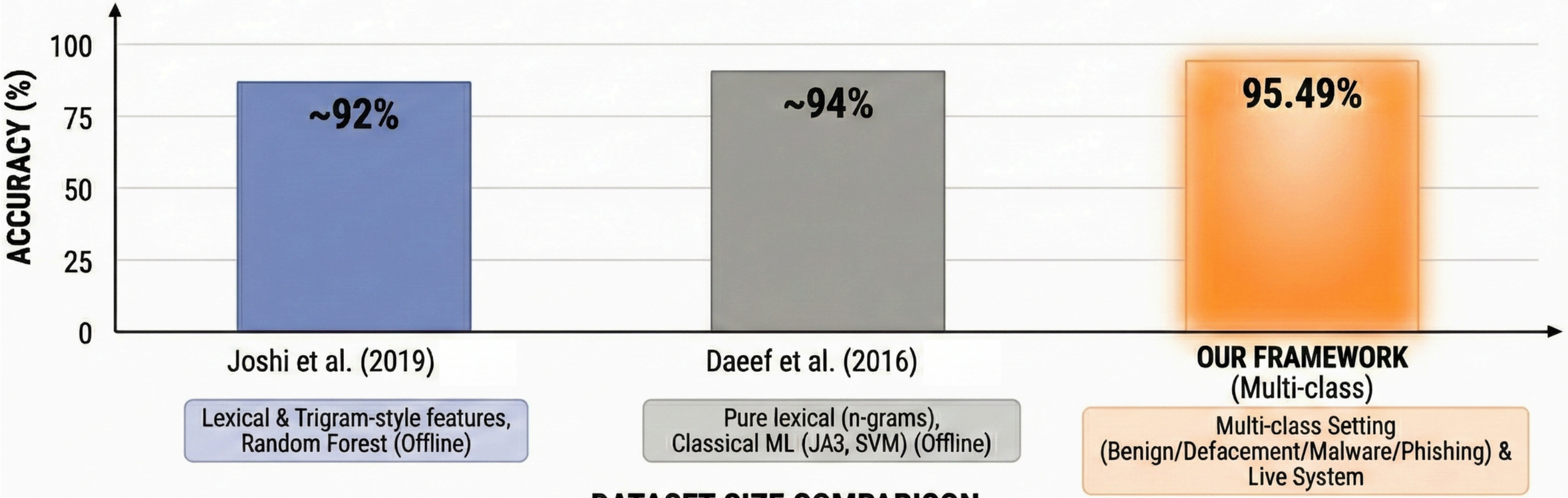


Three distinct URLs submitted.

Asynchronous analysis pipeline.

Correct classifications returned: Benign, Defacement, Malware.

COMPARISON WITH BASELINE STUDIES



DATASET SIZE COMPARISON

APPROACH	DATASET SIZE
Daeef et al. (2016) [9]	~71,000 URLs
OUR FRAMEWORK	**~650,000 URLs**
Joshi et al. (2019) [10]	~5,000,000 URLs

Key Takeaways

Speed:
URL-only analysis allows for privacy-preserving, millisecond detection.

Synergy:
Combining Lexical N-grams with Statistical features creates the most stable model.

Granularity:
Multi-class detection (4-way) offers actionable security intelligence.

Scalability:
Distributed architecture (Redis/Celery) is essential for real-world traffic.

Q & A

Discussion & Feedback

https://www.url

Thank You.