BinaryInferno:
Evaluating Automatic Reverse Engineering of Binary Message Formats

Jared Chandler
Tufts University

Adam Wick
Fastly

Kathleen Fisher
DARPA

This material is based upon work partly supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-19-C-0073. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Approved for Public Release, Distribution Unlimited.

Overview

- Use case: The Task
- What is BinaryInferno?
- Representative Data
- Collection / Processing
- Tools for comparison & Issues
- Recap Lessons Learned

Use Case: Task

Analyst has a sample of packets and wants to understand the protocol.

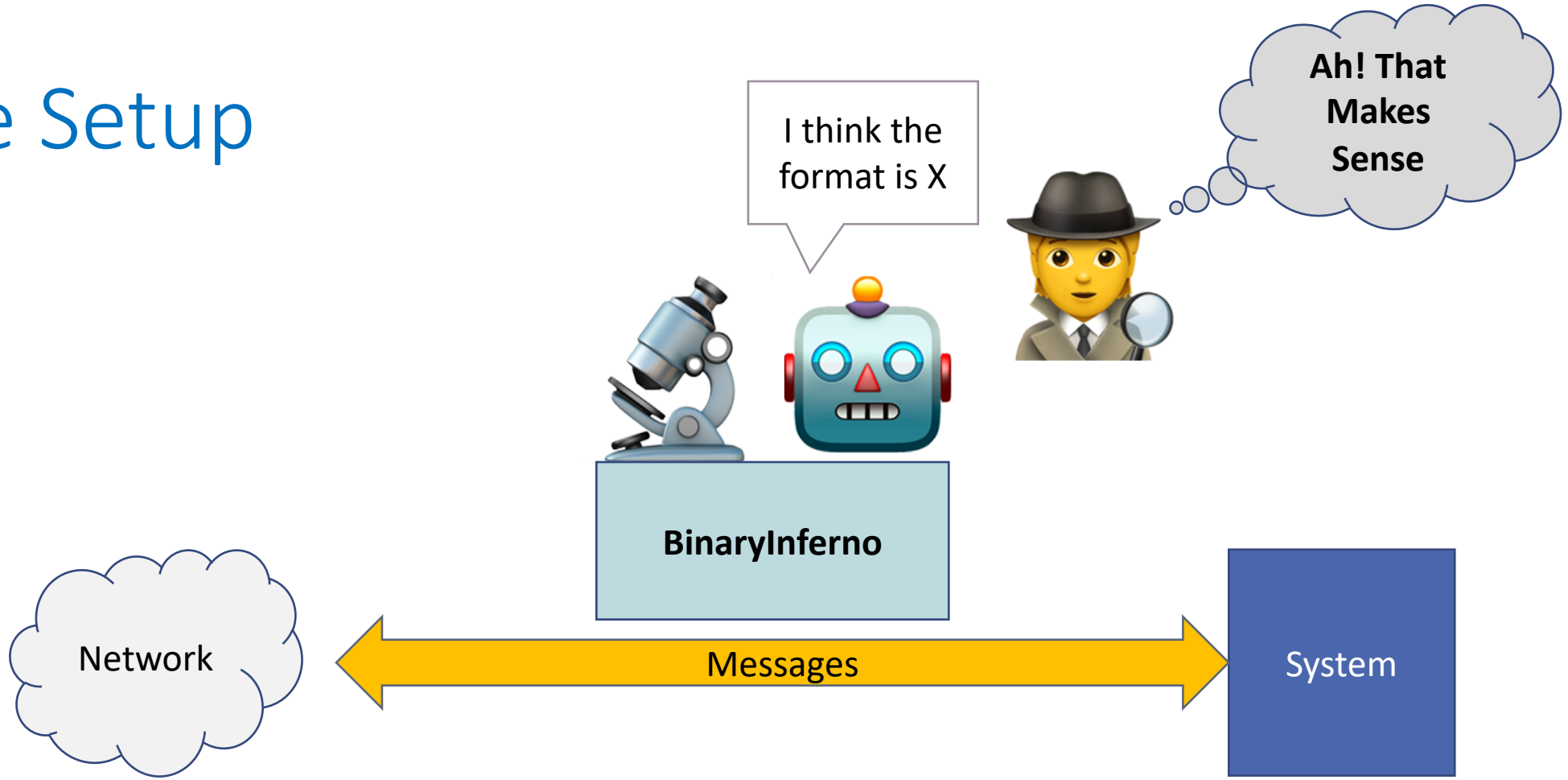
- General Protocol RE Steps
  - Group messages by format
  - Understand each format
    - Understand where the fields are in each format (There's a field at bytes 1-2)
    - Understand what the datatype is (It's an integer)
    - Understand what the field represents (It's the message length)
  - Model the state machine

Use Case: Task

Analyst has a sample of packets and wants to understand the protocol.

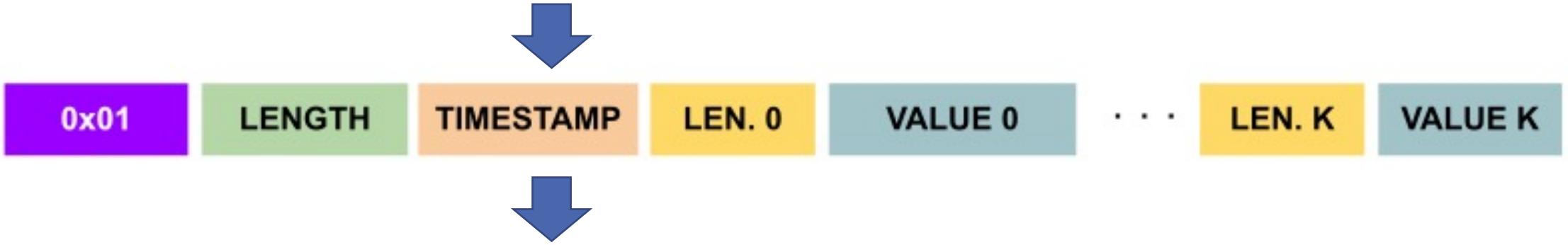
- General Protocol RE Steps
  - Group messages by format
  - Understand each format
    - Understand where the fields are in each format (There's a field at bytes 1-2)
    - Understand what the datatype is (It's an integer)
    - Understand what the field represents (It's the message length)
  - Model the state machine

The Setup


I think the format is X
Ah! That Makes Sense
BinaryInferno
Network
Messages
System

BinaryInferno

```
01000D60A67AED054150504C45
01000E60A67AF9064F52414E4745
01001160A67B0504504C554D0450454152
```




```
01 13 5/20/21 15:06:21 05 4150504C45
01 14 5/20/21 15:06:33 06 4F52414E4745
01 17 5/20/21 15:06:45 04 504C554D 04 50454152
```

BinaryInferno Considerations for Evaluation

- Limited Ensemble
  - Floats
  - Length Fields
  - Timestamps
  - Boundaries
- 3 common serialization patterns

Experimental Setup

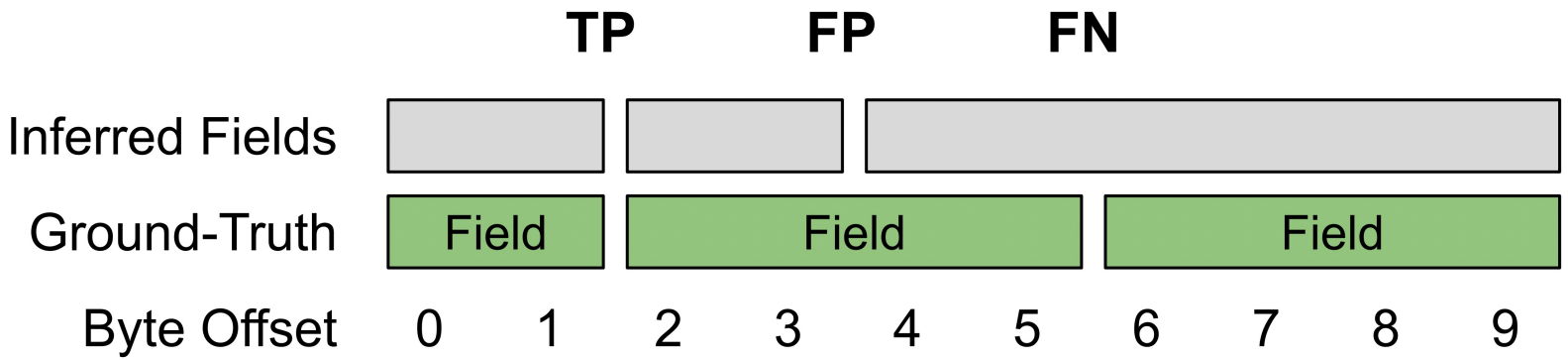
- How did you choose to evaluate it?
- What did you compare against?
- What did you run it on?

Evaluation: Measuring Performance

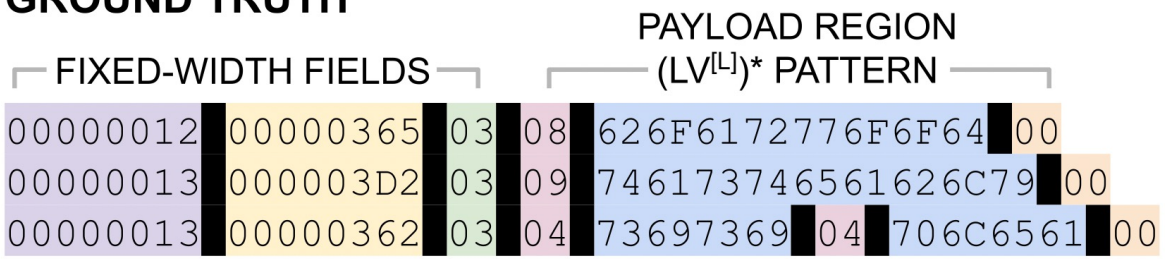
$$Precision = \frac{\text{Inferred True Field Boundaries (TP)}}{\text{Inferred Field Boundaries (TP + FP)}}$$

$$Recall = \frac{\text{Inferred True Field Boundaries (TP)}}{\text{True Field Boundaries (positives)}}$$

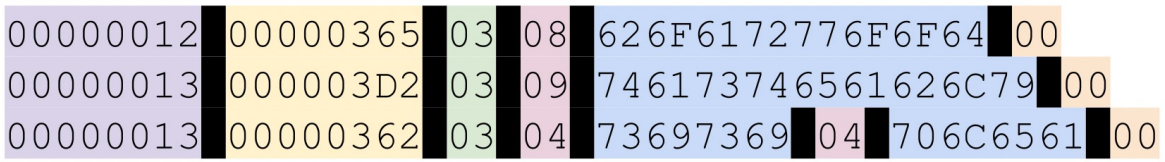
$$FPR = \frac{\text{Inferred False Field Boundaries (FP)}}{\text{Adjacent Field Bytes (negatives)}}$$



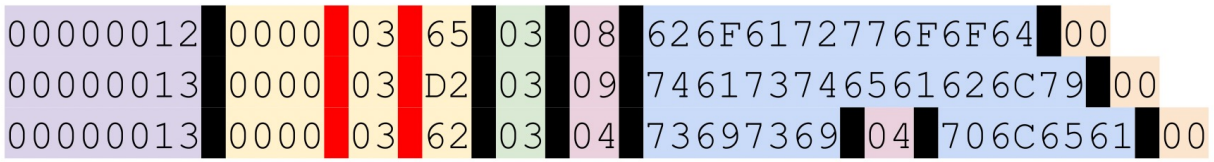
GROUND TRUTH



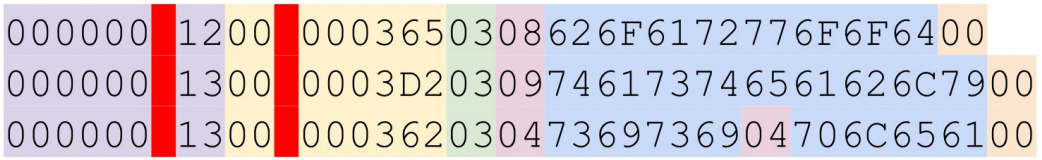
BI+



BI



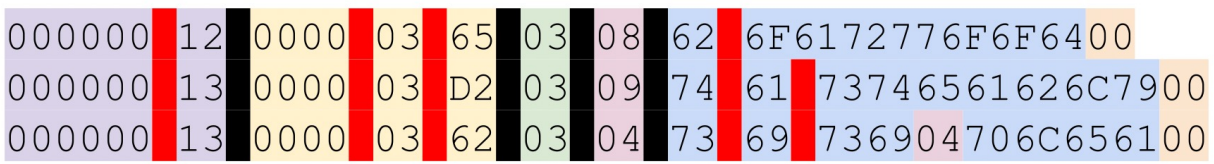
AWRE



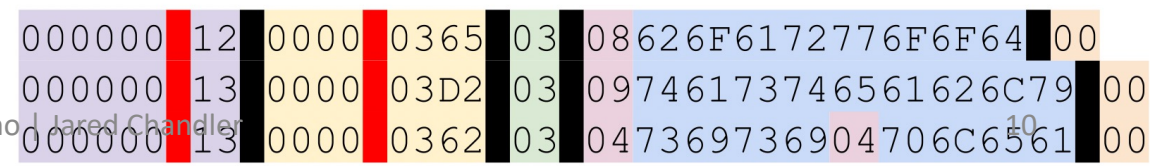
FIELDHUNTER

NEMESYS

NETPLIER



NETZOB



Experimental Setup

- How did you choose to evaluate it?
- What did you compare against?
- What did you run it on?

Evaluation

Protocol	Category
• bgp	Network
• dhcp	Network
• dnp3	Industrial Control
• mavlink	UAV/Drone
• mirai c2	Malware
• modbus	Industrial Control
• ntp	Network
• smb	Network
• smb2	Network
• tutorial	Teaching

Protocol Reverse Engineering Tools

- AWRE [WOOT 19]
- FIELDHUNTER [IFIP 15]
- NEMESYS [WOOT 18]
- NETPLIER [NDSS 21]
- NETZOB [SSTIC 12]

Representative Data: Lessons Learned

- Network data is Sensitive
- Desensitized data may not be representative
- Public samples can be very large or very small
  - 2 packets → 10gb
- Diversity Important for reverse engineering from protocol samples

Collection

- ~~What do you want?~~ What can you get?
  - Proprietary / Sensitive
- How much can you get?
  - How diverse is the data?
- Is there an unambiguous format?

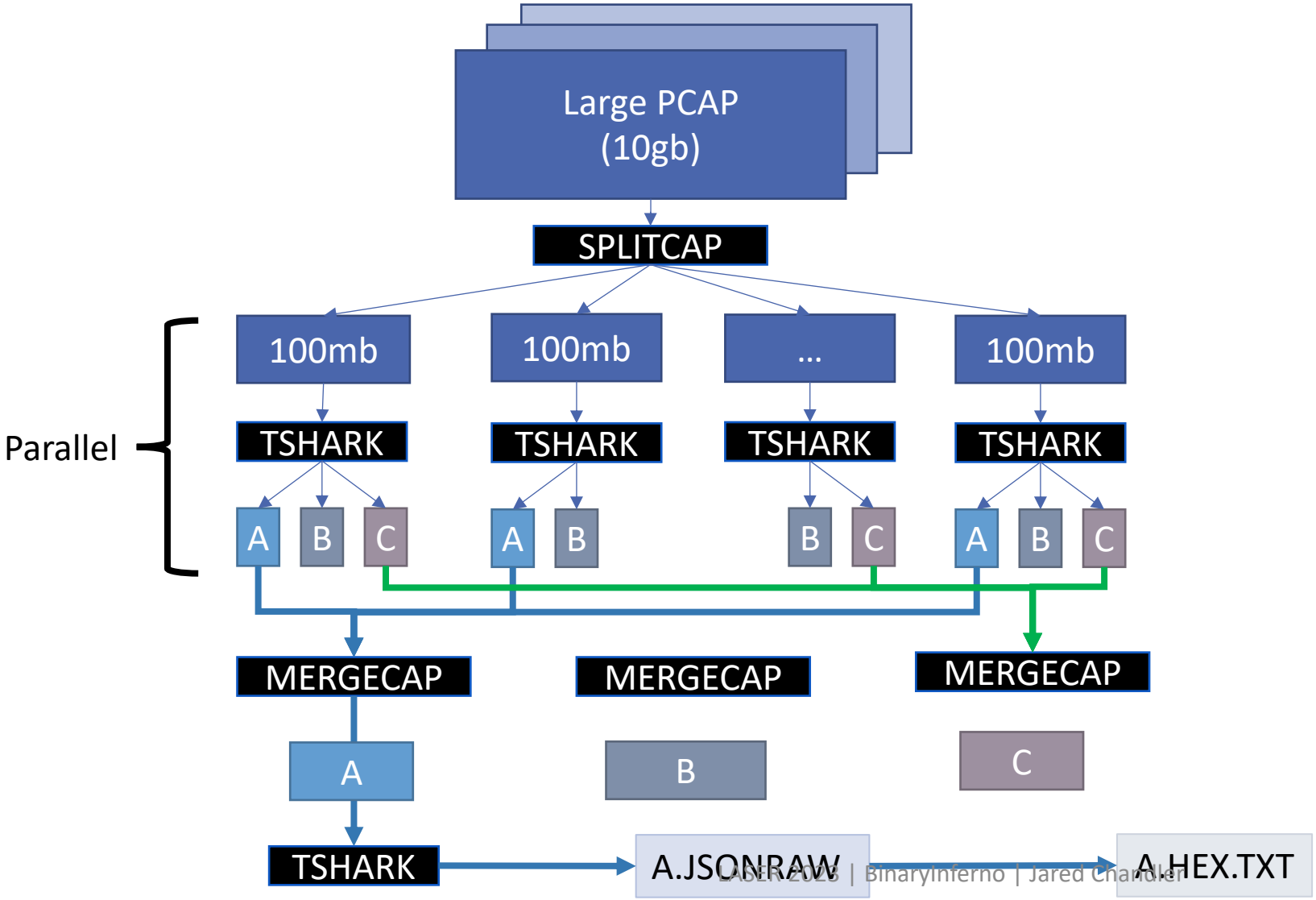
Data Sources

- Make it yourself (So we build a botnet)
  - Time consuming
  - Need to generate sufficient diversity
  - Need to generate enough of it
- Public PCAPs
  - May not have enough messages (2 packets is enough, right?)
  - Limited diversity of field values (Hope you like 192.168.0.x)
- Big PCAPs from Security Exercises
  - Lots of data, but limited protocol diversity
  - Malformed packets (Because exploits)

T-Shark

- Command line interface to Wireshark functions
- Read / Write PCAPS
- Filter PCAPs
- Various Output Formats
  - Raw
  - JSONRaw (Precise but Huge)

Collection & Processing

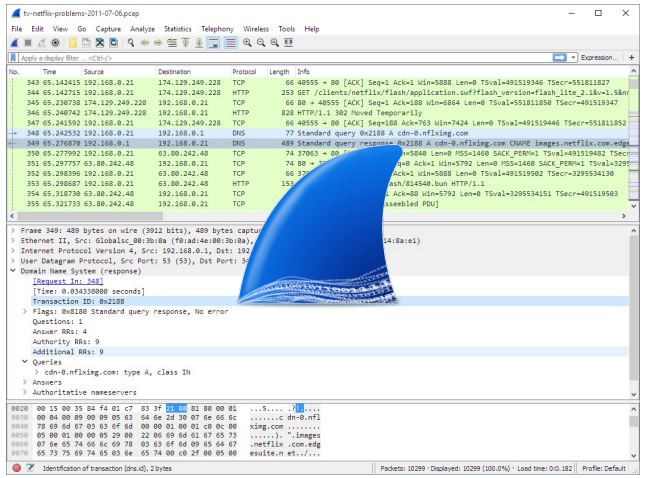


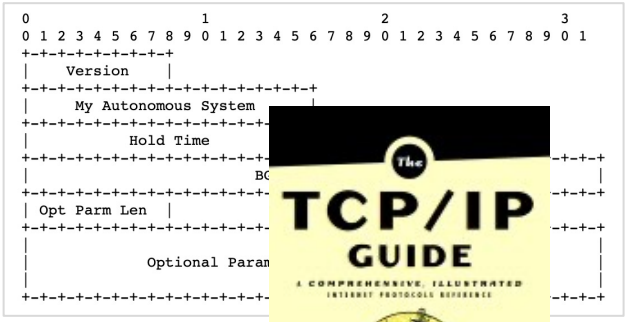
Wireshark Isn’t Perfect as Ground Truth

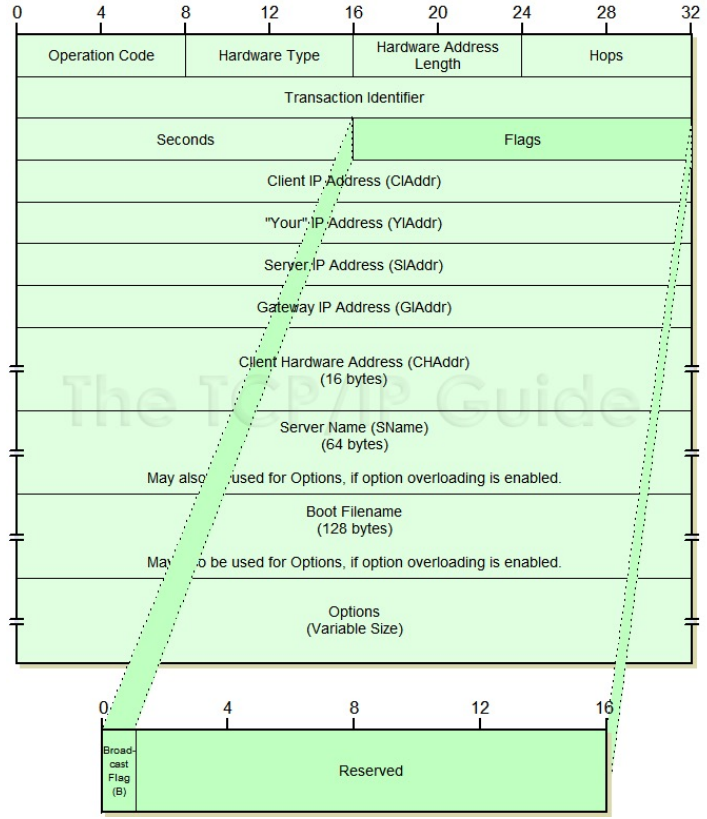
- A Wireshark dissector won’t exist for small / weird protocols
- Different Dissectors → Different Authors
- It may be captured in some other way besides tcpdump
- Wireshark tries to ”do the right thing” with corrupt packets
- Filtering is tricky
  - DNS → DNS anywhere in any packet
  - May just be TCP DATA on a particular port

Specs for Ground Truth

- RFCs (ASCII Art is really helpful)
- Wireshark Dissector
- Security Analysis / Bulletins / Blogs
- Documentation

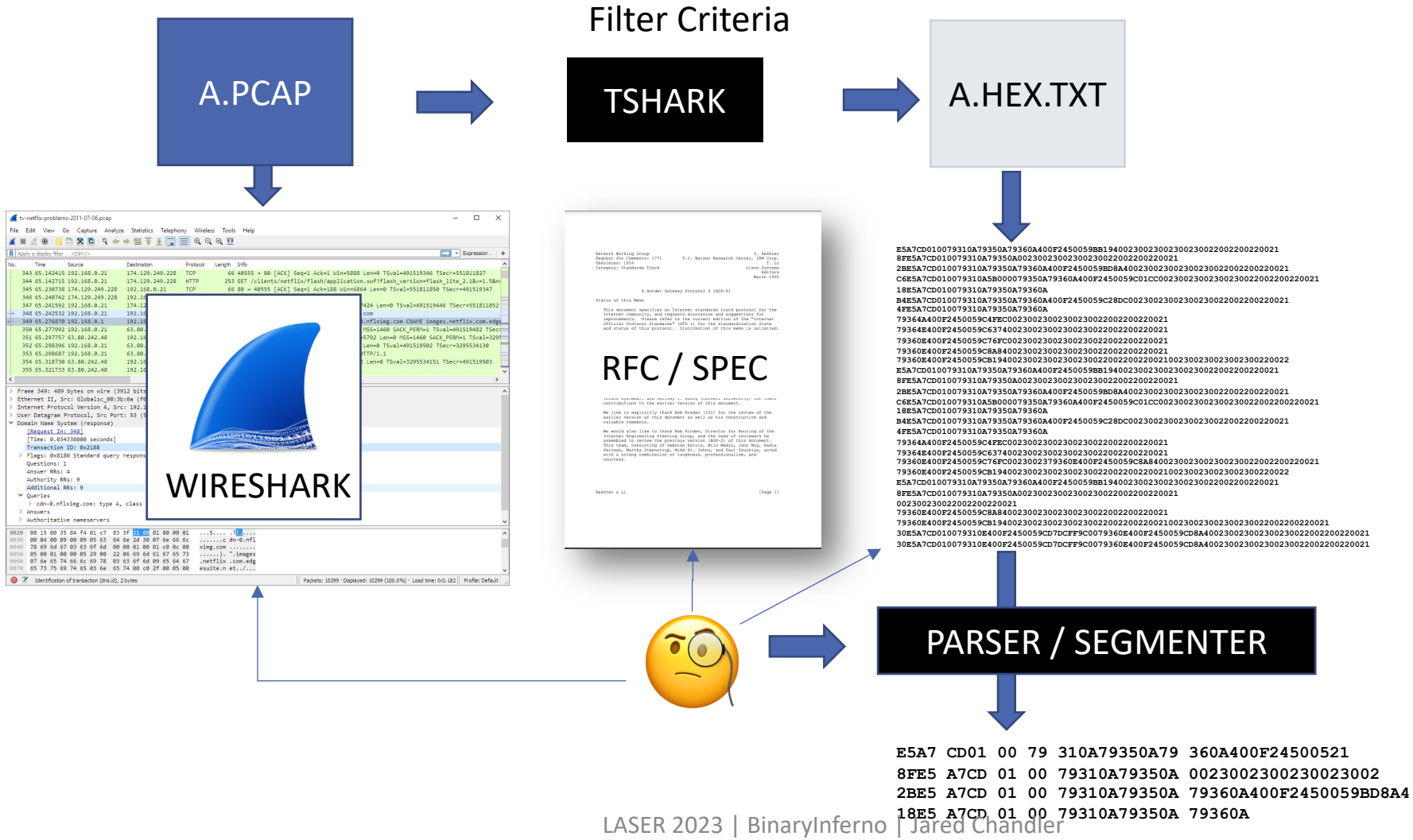




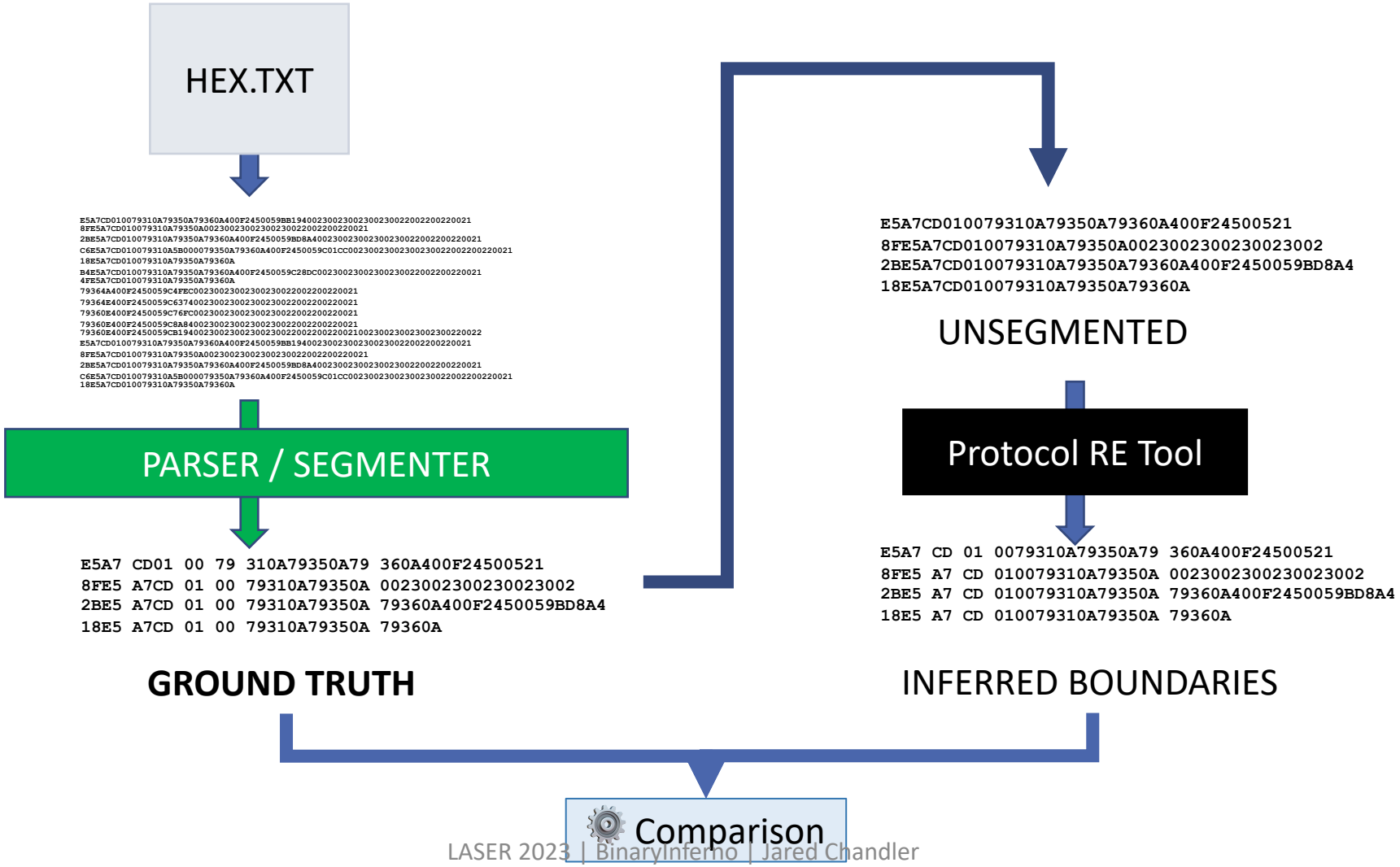


http://www.tcpipguide.com/free/t_DHCPMessageFormat.htm

Parallel Construction to get Good Data for Test



In Terms of Evaluation



Collection & Processing: Lessons Learned

- Don’t trust anything
- You need both the data and the ground truth
  - Find the spec
  - Validate on example messages
  - **Write a parser, use that to find malformed messages**
    - **“Why didn’t our technique work? Out of Spec data was included in the input”**
- Security Captures have malformed data (stuff out of spec)
- T-Shark is useful to get stuff out but…
  - Will parse malformed stuff
  - Parses it fully, have to reconstruct resolutions posthoc
  - Slow (json_raw output)

Tools for Comparison: Lessons Learned

- Dependency Hell
- Everyone uses their own intermediate format
- Beware anyplace there's a transform or filtering
- The technique and the evaluation are effectively one.

- If it's meant to be used as a tool, provide an example where the ground truth is unknown.

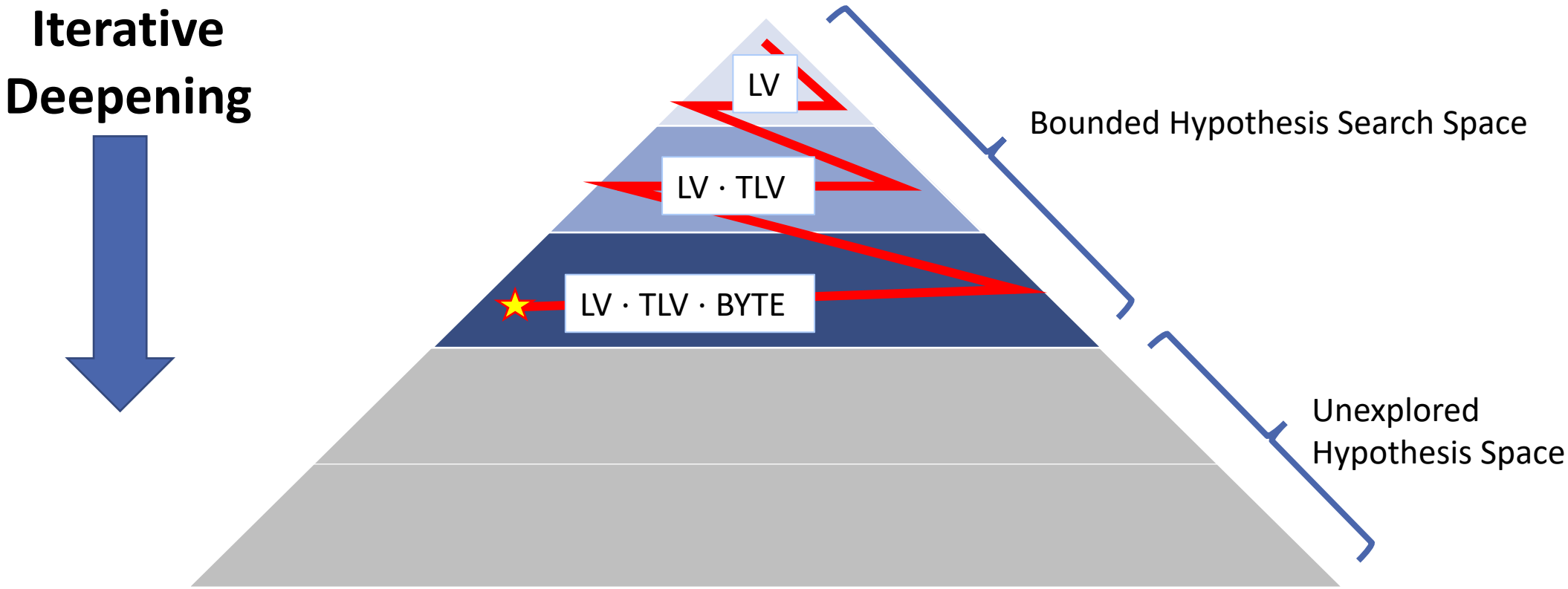
Diverse Set of Tools for Comparison

- AWRE → Wireless Radio Protocols at the Bit Level
- FIELDHUNTER [IFIP 15] → Semantics → Fields
- NEMESYS [WOOT 18] → Heuristics
- NETPLIER [NDSS 21] → Full Stack Protocol Reverse Engineering
- NETZOB [SSTIC 12] → “REPL” with some automation

Experimental Issues

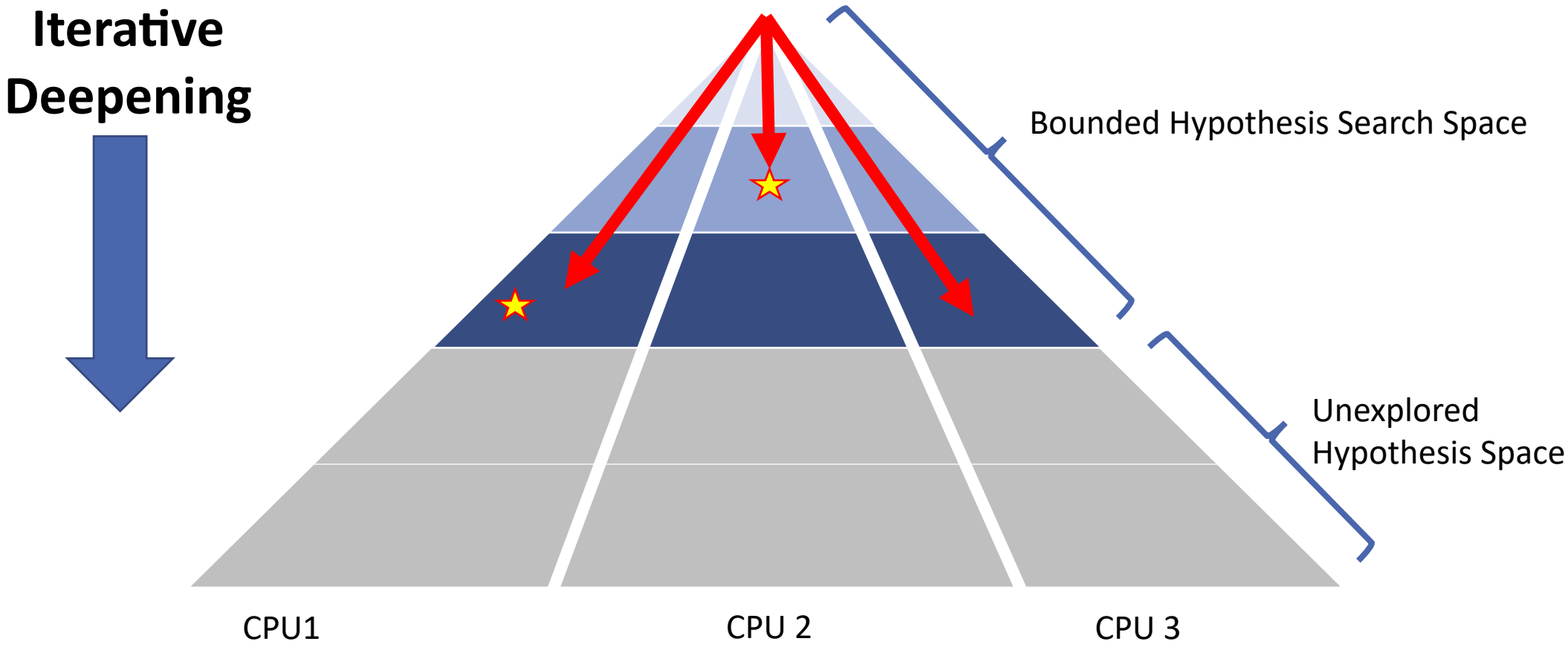
- Parallelization Parameters
- Typical “Subtle Implementation Bugs”

Search Hypothesis Space Exploration



★ = Hypothesis consistent with message samples

Hypothesis Space Exploration in Parallel

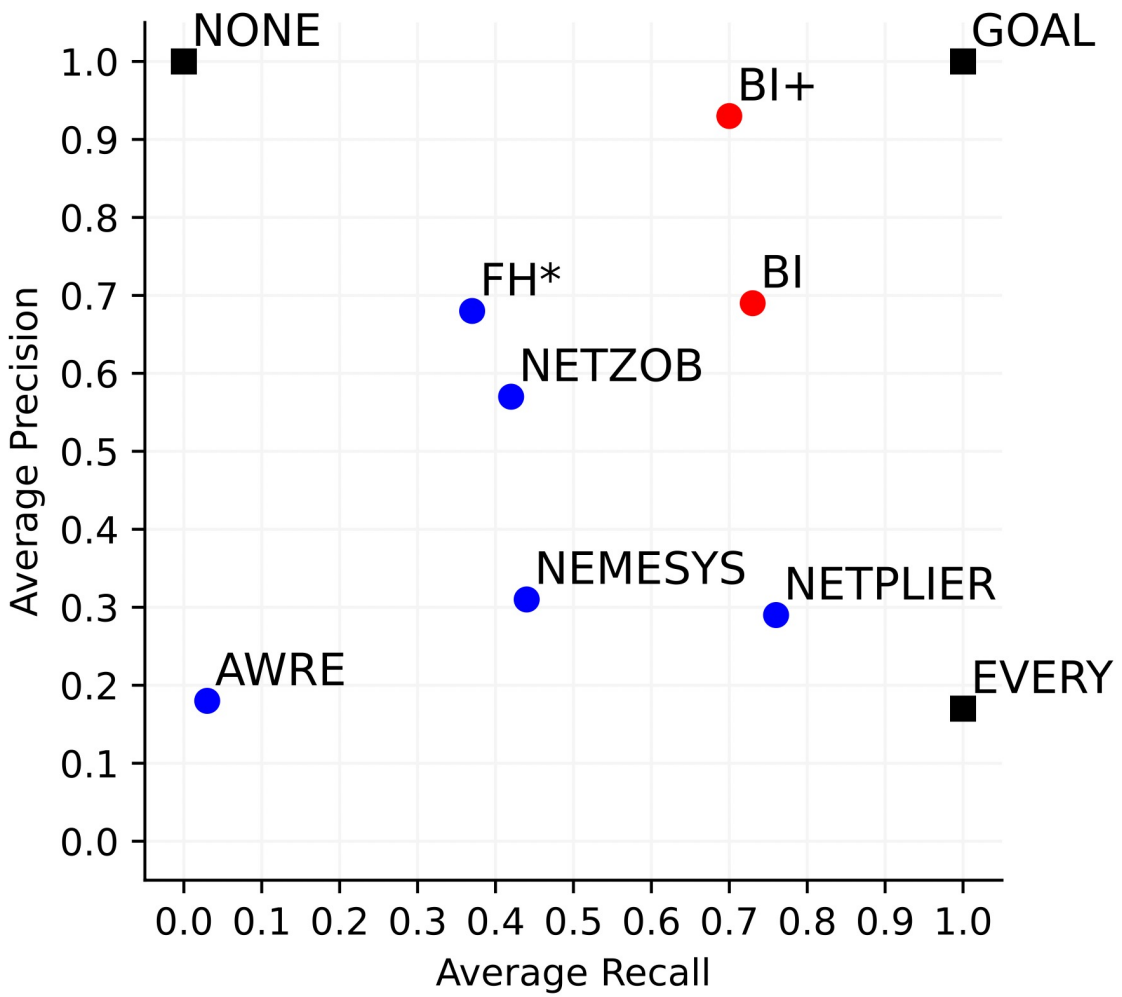


★ = Hypothesis consistent with message samples

Experimental After-thoughts

- Ok, so these tools all work to varying degrees on protocol data…
- We have an idea of “how good” these systems can be
- But is that the end of the story?
- What about how bad?
- What happens when I give these tools random data?

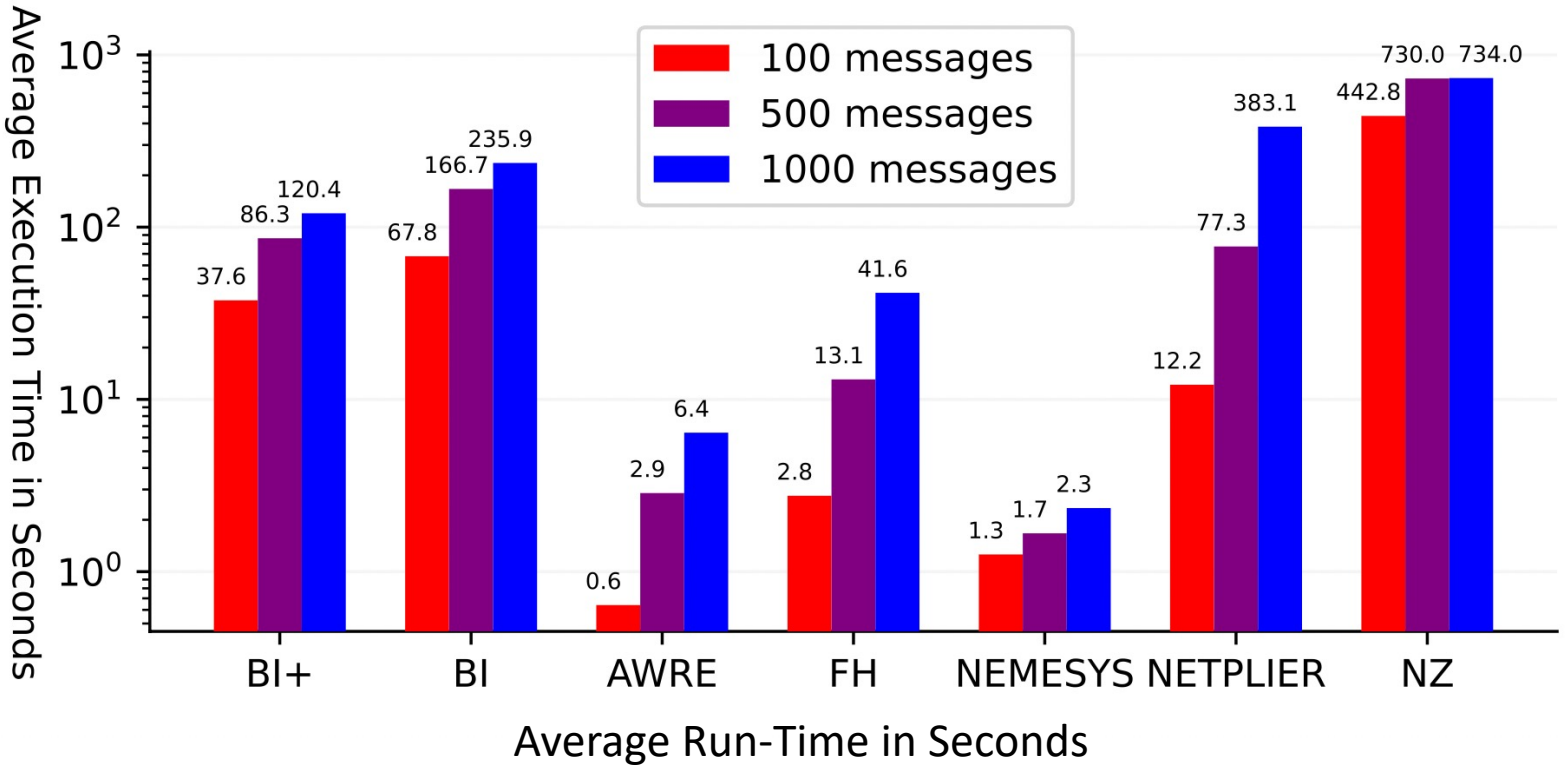
Average Performance on Top-Level Samples



Tool	# Msgs.	Prec.	Rec.	FPR	F1
BI+	1000	**0.93**	0.70	**0.005**	**0.78**
	500	**0.91**	0.69	**0.005**	**0.77**
	100	**0.82**	0.63	0.02	**0.70**
BI	1000	0.69	0.73	0.04	0.70
	500	0.69	0.72	0.04	0.69
	100	0.60	0.62	0.05	0.59
AWRE	1000	0.18	0.03	0.04	0.05
	500	0.18	0.03	0.04	0.05
	100	0.08	0.01	0.05	0.02
FIELDHUNTER*	1000	0.68	0.37	0.01	0.45
	500	0.68	0.37	0.01	0.45
	100	0.48	0.29	**0.01**	0.34
NEMESYS	1000	0.31	0.44	0.11	0.34
	500	0.31	0.45	0.11	0.34
	100	0.30	0.44	0.10	0.33
NETPLIER	1000	0.29	**0.75**	0.22	0.38
	500	0.27	**0.73**	0.22	0.37
	100	0.25	**0.83**	0.26	0.37
NETZOB	1000	0.57	0.42	0.03	0.47
	500	0.57	0.42	0.03	0.48
	100	0.53	0.45	0.04	0.45

* Results where FieldHunter found no fields are excluded from averages.

Average Execution Times



Testing on Random Data

10 Fixed-Width Datasets

10 Variable-Length Datasets

Tool	False Positive Rate
NETPLIER	0.45
NEMESYS	0.40
AWRE	0.02
NETZOB	0.01
FH	0.0
BI	0.0

Recap

- Did you use experimentation artifacts borrowed from the community?
  - Implementations
- Did you attempt to replicate or reproduce results of earlier research as part of your work?
  - Minimally, only to verify that other tools were functional
- What can be learned from your methodology and your experience using your methodology?
  - Getting representative data for this protocol analysis can be hard
- What did you try that did not succeed before getting to the results you presented?
  - Naively assumed that data was clean
  - Parallelization parameters

- Did you produce any intermediate results including possible unsuccessful tests or experiments?
  - We used synthetic data as part of the development of our approach
  - ”Base cases” / Simple test cases.

Conclusion

Jared Chandler

PhD Candidate
Tufts University
Department of Computer Science

jared.chandler@tufts.edu

Acknowledgements: This material is based upon work partly supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-19-C-0073. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. Distribution Statement A: Approved for public release. Distribution is unlimited.