Packet Field Tree: a hybrid approach to automated protocol reverse-engineering

Alex Rohl*†, Matthew Roughan*, Martin White*, Alexander Chambers†
*University of Adelaide, Australia
†Defence Science & Technology Group, Australia

***Abstract*—Information and communications technology (ICT) systems exchange data through agreed network protocols. These constrain how ICT systems send network packets through a protocol specification. Protocol specifications are useful knowledge for building intrusion detection systems to monitor system behaviour, or for performing penetration testing to find system vulnerabilities. However, protocol specifications are typically verbose documents, and when new protocols are tested, considerable manual effort is required to translate these specifications into digital models of expected protocol behaviour. Automated protocol reverse-engineering (APRE) is a precursor to generating such digital models for the aforementioned applications. In this paper, we introduce the *Packet Field Tree (PFT)*, the first hybrid approach combining the advantages of heuristic and supervised-learning methods, and apply it to one aspect of the APRE problem concerning protocol formats. Our model infers the boundaries between fields of an unknown protocol with a superior mean perfection, an APRE-specific quasi-accuracy measure, of 0.27 over a variety of 21 protocols. This is an improvement of at least 0.11 compared to three existing APRE models. Although this is far from a perfect score of 1.00, highlighting the difficulty of APRE, our PFT also infers the field syntax types, which is a feature unique to our model. To measure this feature, we also propose the *PFT-Score*, a starting point for holistic evaluation of APRE outputs.**

I. Introduction

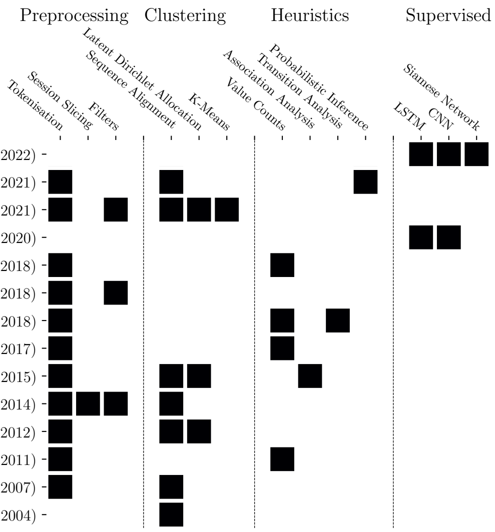
Network protocols are a fundamental component of the information and communications technology (ICT) ecosystem. Individual messages generated by a networked system are called *packets*, which are typically composed of multiple *layers* of encapsulated protocols. Engineering these protocols involves structures such as inter-packet state-machines, inter-layer encapsulation, and intra-layer protocol formats. An ongoing question in the research of these structures is, to what extent can each be automatically *reverse*-engineered from raw packet bytes? Our work attempts to improve on answering this question for intra-layer structures by automatically inferring their field boundaries and field syntaxes, the *protocol format*, within an unknown protocol trace.

There are many reasons why protocol reverse-engineering (PRE) may be required, such as ensuring cross-platform compatibility, analysing malicious communications or mining system vulnerabilities. Open-source manual PRE projects are predominately community-driven, rather than industry-driven, owing to potential legal issues. Frequently cited examples of such *manual* efforts include Samba (Tridgell, 1992) where reverse-engineering the Server Message Block (SMB) protocol was required, chat applications (Spencer, 1998) and smart toys (Rabet, 2019). PRE is often an ongoing effort since protocols may be indefinitely updated to support new functionality (Caballero and Song, 2013), as evidenced by the existence of multiple SMB versions. Therefore, automated methods are required to keep pace with continually evolving protocol suites. We introduce the first hybrid APRE model combining heuristic and supervised methods, the Packet Field Tree, and benchmark it on a variety of 21 protocols against existing techniques.

978-3-903176-58-4 ©2023 IFIP

II. Related Work

Since Beddoe (2004) first applied sequence-alignment, from the bioinformatics domain, to the APRE problem, there has been a surge of research interest in applying cross-domain techniques to APRE.



	Preprocessing			Clustering			Heuristics				Supervised		
	Tokenisation	Session Slicing	Filters	Sequence Alignment	Latent Dirichlet Allocation	K-Means	Value Counts	Association Analysis	Transition Analysis	Probabilistic Inference	LSTM	CNN	Siamese Network
ProsegDL, Zhao et al. (2022)											■	■	■
NetPlier, Ye et al. (2021)	■			■						■			
Dasgupta et al. (2021)	■		■	■	■	■							
Yang et al. (2020)											■	■	
GrAMeFFSI, Ladi et al. (2018)	■						■						
NEMESYS, Kleber et al. (2018)	■		■										
Nolan et al. (2018)	■						■		■				
Markovitz et al. (2017)	■						■						
Li et al. (2015)	■			■	■			■					
Netzob, Bossert et al. (2014)	■	■	■	■									
ProDecoder, Wang et al. (2012)	■			■	■								
ReverX, Antunes (2011)	■						■						
Discoverer, Cui et al. (2007)	■			■									
PI, Beddoe (2004)				■									

Fig. 1. **Techniques by Paper:** for the listed papers (and tool name where applicable) on the y-axis, we plot the class of technique on the x-axis. Most methods required some degree of tokenisation beyond considering raw bytes, such as using *n-grams* (Wang et al., 2012). However, Yang et al. (2020) and Zhao et al. (2022) apply supervised-learning methods that can infer patterns from raw bytes.

As illustrated in Figure 1, existing works apply clustering methods (Beddoe, 2004; Cui et al., 2007; Wang et al., 2012; Bossert et al., 2014), heuristic techniques (Markovitz and Wool, 2017; Nolan et al., 2018; Ladi et al., 2018; Ye et al., 2021) and supervised deep-learning (Yang et al., 2020; Zhao et al., 2022). In particular, no method implements both heuristic and supervised-learning techniques. The advantage of heuristic methods, as applied to APRE, is that they can reason with inputs of variable length; the advantage of supervised methods is that they improve with more training data. Our approach is the first to attempt a hybrid model that combines the advantages of both methods into a single algorithm.

III. Our Approach

Our APRE solution extends the approach by Ladi et al. (2018). We found the following limitations of their *GrAMeFFSI* algorithm:

1) "Highly-variable" fields are restricted to one-byte and no type inference is applied.
2) Once a branch contains a single remaining sequence of packet bytes, this is treated as a constant field and no further boundaries are inferred.

Our Packet Field Tree (PFT) algorithm, overcomes both limitations by extending GrAMeFFSI (§III-1) with a supervised model that is fitted on *Transition Aggregation N-Gram (TANG)* (Nolan et al., 2018) features and field-syntax labels for each byte in our set of *known* training protocols (§III-2). When inferring the *unknown* test protocol, the trained model is used to predict the probability of each syntax label for each byte in the tree. From these probabilities, the most likely protocol format is inferred by our optimisation algorithm (§III-3).

1) Unsupervised Component: Our initial construction reuses the method from Ladi et al. (2018). To have fixed-dimension inputs for supervised-learning methods, we restrict the nodes of the tree to 1-byte-length fields, which will later be combined into larger fields in Section III-3.

2) Supervised Component: Supervised-learning models require features and labels to be fitted. To create separate models for each protocol, the interrogated *unknown* protocol is left out for evaluation. From the remaining 20 *known* protocols, we split our data into a training and validation set of 10 protocols each. Wireshark's (Combs, 1998) internal syntax describes an explicit rule for field classification. Our training set has 7 classes for type classification: `ABSOLUTE_TIME`, `ETHER`, `IPv4`, `STRING`, `UINT16`, `UINT32`, `UINT8`. For each byte, the TANG features are computed and a field type is assigned. We fit 4 models on the respective training set: logistic regression (LR), gradient-boosted decision trees (GBDT), random forest (RF) and linear support vector machine (LSVM) classifiers from the Scikit-learn Python library (Pedregosa et al., 2011). To ensure the probabilistic output accurately reflects the model's confidence, we implement calibrated classifiers (Niculescu-Mizil and Caruana, 2005) in Scikit-learn. The best performing model on the validation set is chosen.

3) Optimal Field Selection: We implement a custom search algorithm that uses the enforced field-type sizes to determine the most likely field sequence. This is similar to the *field division* algorithm from Yang et al. (2020). To reconcile the type inference for the entire protocol structure, we resolve cases where sibling nodes in the PFT each initiate conflicting optimal field sequences. This is achieved by considering the most likely field-type labelling for every subtree of the PFT, recursing upwards to the root node, and choosing its determination of the optimal fields. Pseudocode is provided in Algorithm 1.

IV. Evaluation

To measure the success of our method, we apply field-boundary measures slightly modified from those used by Ye et al. (2021). Rather than using their *correctness* measure, we separate this into *completeness* and *conciseness* to verify if the model is inferring more or less fields than necessary, respectively:

$$\textbf{Completeness: } \frac{\text{\# of inferred fields within a true field}}{\text{\# of inferred fields}}$$

$$\textbf{Conciseness: } \frac{\text{\# of true fields within an inferred field}}{\text{\# of true fields}}$$

We use the same *perfection* measure as Ye et al. (2021):

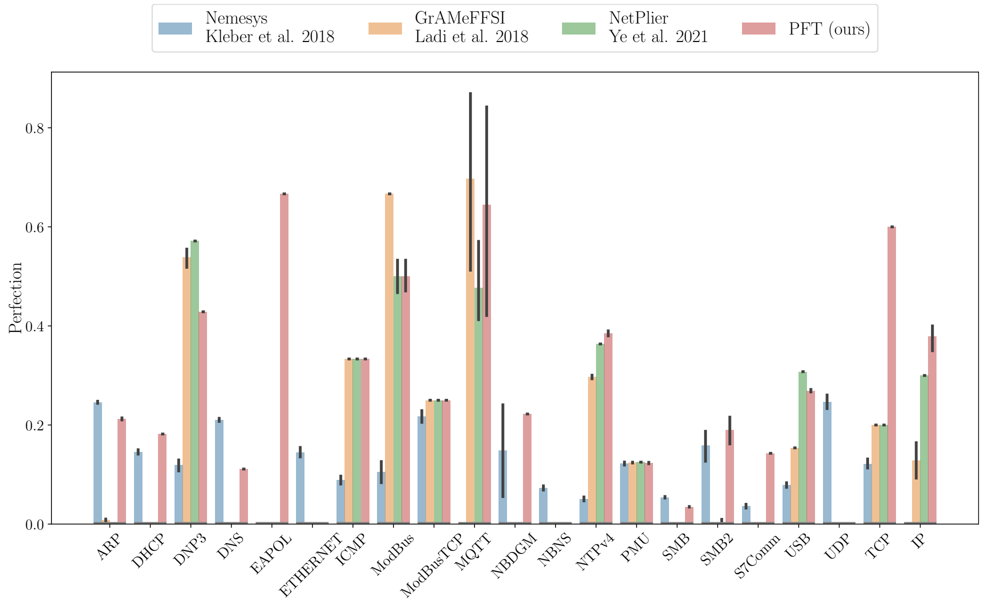
$$\textbf{Perfection: } \frac{\text{\# of inferred fields that match a true field}}{\text{\# of true fields}}$$

Across the 21 protocols in our dataset, we compute these measures for each APRE model with respect to the ground truth protocol format, as illustrated in Figure 2. For 11 protocols, GrAMeFFSI and NetPlier inferred no fields perfectly. Our PFT provided the highest perfection for 7 protocols and equal highest for a further 3 protocols. Our mean and median perfection scores were higher than the next best APRE model by 0.11 and 0.10, respectively. Whilst this may be a limited improvement, we have gained the fidelity of field-syntax and field-branching inference.

To provide an evaluation of the field-syntax and field-branching inference of our model, we propose the *PFT-Score*. This measure is intended to combine the following areas of evaluation in the literature: 'Cluster', 'Field Boundary' and 'Field Type', by considering branching, number of nodes in a branch, and the node syntax label, respectively. In Table I, from the number of *predicted* nodes (PN), we calculate the number of nodes required to be deleted (PND) and inserted (TNI), to match with the *true* syntax nodes (TN). A node is considered matched if the inferred and true syntax label are the same. We combine these 4 values such that a score of 0 or 1, means all inferred nodes are incorrect or correct, respectively:

$$\textbf{PFT-Score: } 1 - \omega\frac{\text{PND}}{\text{PN}} - (1-\omega)\frac{\text{TNI}}{\text{TN}},$$

where $0 \leq \omega \leq 1$. We equally weight the cost of an end user inserting and deleting nodes to match the true field tree, by choosing $\omega = 0.5$. However, we invite the research community to suggest reasoned weights. As seen in Table I, the PFT scored the highest for Ethernet, TCP and EAPOL, with 0.87, 0.68 and 0.58, respectively. However, these are simplistic protocols as indicated by the number of true nodes: 3, 5 and 3, respectively. We encourage the development of new APRE techniques to directly improve on our PFT-scores.



Model	Perfection		Completeness		Conciseness	
	mean	median	mean	median	mean	median
Nemesys	0.12	0.12	0.52	0.58	**0.74**	0.75
GrAMeFFSI	0.16	0.01	0.68	0.72	0.70	**0.80**
NetPlier	0.16	0.00	**0.85**	**0.94**	0.51	0.50
PFT (ours)	**0.27**	**0.22**	0.80	0.83	0.62	0.71

Fig. 2. **Perfection, Completeness and Conciseness comparisons.** The error bars capture the range of scores due to the variation in protocol formats. In the perfection plot, we see the results are protocol dependent, and for all models, there are cases where zero fields are correctly inferred. Given this variability, we compute both the mean and median for each measure. Our PFT method, had the highest mean and median perfection.

	Protocol																				
	ARP	DHCP	DNP3	DNS	EAPOL	Ethernet	ICMP	IP	Modbus	ModbusTCP	MQTT	NBDGM	NBNS	NTPv4	PMU	SMB	SMB2	S7Comm	TCP	USB	UDP
Predicted Nodes (PN)	63	48	8	29	4	12	18	34	5	4	28	45	23	437	284	20	65	32	4	21	11
True Nodes (TN)	9	22	7	9	3	3	6	10	5	4	14	9	9	11	8	30	33	7	5	13	4
Pred. Nodes Deleted (PND)	60	36	5	27	2	9	16	30	3	3	22	43	23	431	284	11	54	30	1	15	11
True Nodes Inserted (TNI)	6	11	4	7	1	0	4	6	3	3	8	7	9	5	8	21	22	5	2	7	4
PFT-Score ($\omega_1 = 0.5$)	.19	.38	.41	.15	**.58**	**.87**	.22	.26	.40	.25	.32	.13	.00	.28	.00	.38	.25	.17	**.68**	.37	.00

TABLE I

PFT Graph-Based Evaluation: The PFT inferred a great number of nodes for NTPv4 and PMU, suggesting there were many 'enumeration' nodes in the prefix tree that failed to be reconciled in our 'optimal field selection' process.

V. Discussion & Further Work

Wireshark not only provides syntax information, but also each field's semantic information. The exact same PFT approach could be applied to infer field semantics, such as `Length`, `Counter` and `Address`. Furthermore, A graph-based measure could give insight into the 'reverse-engineering difficulty' of a protocol and allow us to provide fair assessments of methods with respect to where each protocol lies on the spectrum of APRE difficulty. These challenges motivate the need for an APRE benchmark dataset. This need is emphasised by the further application of data-driven methods, that improve with higher quality and larger volumes of data.

VI. Conclusion

We introduced the Packet Field Tree (PFT), the first hybrid APRE approach combining heuristic and supervised methods, and provided comparisons with existing models for 21 different protocols. Our model offered the highest mean perfection of 0.27, when compared to three baseline models. In addition, the PFT produced a protocol specification in a graph format that includes field-syntax inference. We proposed the PFT-Score to measure the success of such inference for future APRE methods. The contribution of syntax fidelity in our model is critical information for an operator to build an intrusion detection system or perform penetration testing.

References

Beddoe, M. A. (2004). Network Protocol Analysis using Bioinformatics Algorithms. Technical report, McAfee, https://github.com/bitpeach/Protocol-Informatics.

Bossert, G., Guihéry, F., and Hiet, G. (2014). Towards automated protocol reverse engineering using semantic information. In *9th ACM Symposium on Information, Computer and Communications Security (ASIA CCS)*, pages 51–62, Kyoto Japan. ACM.

Caballero, J. and Song, D. (2013). Automatic protocol reverse-engineering: Message format extraction and field semantics inference. *Computer Networks*, 57(2):451–474.

Combs, G. (1998). Wireshark. wiki.wireshark.org.

Cui, W., Kannan, J., and Wang, H. J. (2007). Discoverer: Automatic Protocol Reverse Engineering from Network Traces. In *16th USENIX Security Symposium*, Boston, MA. USENIX Association.

Ladi, G., Buttyan, L., and Holczer, T. (2018). Message Format and Field Semantics Inference for Binary Protocols Using Recorded Network Traffic. In *26th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–6, Split. IEEE.

Markovitz, M. and Wool, A. (2017). Field classification, modeling and anomaly detection in unknown CAN bus networks. *Vehicular Communications*, 9:43–52.

Niculescu-Mizil, A. and Caruana, R. (2005). Predicting good probabilities with supervised learning. In *22nd International Conference on Machine Learning (ICML)*, pages 625–632, Bonn, Germany. ACM Press.

Nolan, B. C., Graham, S., Mullins, B., and Kabban, C. S. (2018). Unsupervised time series extraction from controller area network payloads. In *88th IEEE Vehicular Technology Conference (VTC-Fall)*, pages 1–5.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Rabet, J. (2019). Adventures in buttplug penetration (testing). Technical report, DEF CON 27, Las Vegas, defcon.org/html/defcon-27/dc-27-speakers.html#smea.

Spencer, M. (1998). Pidgin. pidgin.im.

Tridgell, A. (1992). Samba. www.samba.org.

Wang, Y., Xiaochun Yun, Shafiq, M. Z., Wang, L., Liu, A. X., Zhang, Z., Yao, D., Zhang, Y., and Guo, L. (2012). A semantics aware approach to automated reverse engineering unknown protocols. In *20th IEEE International Conference on Network Protocols (ICNP)*, pages 1–10, Austin, TX, USA. IEEE.

Yang, C., Fu, C., Qian, Y., Hong, Y., Feng, G., and Han, L. (2020). Deep learning-based reverse method of binary protocol. In *Security and Privacy in Digital Economy*, pages 606–624, Singapore. Springer Singapore.

Ye, Y., Zhang, Z., Wang, F., Zhang, X., and Xu, D. (2021). NetPlier: Probabilistic Network Protocol Reverse Engineering from Message Traces. In *Network and Distributed System Security Symposium*, Virtual. Internet Society.

Zhao, S., Wang, J., Yang, S., Zeng, Y., Zhao, Z., Zhu, H., and Sun, L. (2022). ProsegDL: Binary Protocol Format Extraction by Deep Learning-based Field Boundary Identification. In *30th IEEE International Conference on Network Protocols (ICNP)*, pages 1–12, Lexington, KY, USA. IEEE.

Appendix

Algorithm 1 Optimal Field Selection for the Packet Field Tree

```
Require: $\mathbb{P}$ (A mapping from a PFT node to a $7 \times 1$ probability array), fields (the 7 Wireshark syntax classes used) and field_lengths (their respective byte lengths).
  best_prob ← zero for all nodes in PFT
  best_prob(Null) ← 1                                  ▷ Base case
  best_syntaxes(Null) ← []
  for $n_{\text{curr}}$ in PFT do                   ▷ node-depth highest to lowest
      for $f$ in Range$(0, 7)$ do
          f_length ← field_lengths[f]                  ▷ Inductive case
          if MAX_DEPTH + 1 $<$ f_length + $n_{\text{curr}}$.depth then
              continue                  ▷ field too long, hence invalid
          end if
          total_prob = 1
          for path in $n_{\text{curr}}$.all_desc_paths_of_len(f_length) do
              f_probs ← [$\mathbb{P}(n_{\text{field}})[f]$ for $n_{\text{field}}$ in path]
              $n_{\text{prevs}}$ ← path[f_length].children
              prob ← $\prod_{n \in n_{\text{prevs}}}$ best_prob$(n)$ $\times \prod_{p \in \text{f\_probs}} p$
              total_prob ← total_prob × prob
              if total_prob $>$ best_prob($n_{\text{curr}}$) then
                  best_prob[$n_{\text{curr}}$]← total_prob
                  best_syntaxes[$n_{\text{curr}}$]← fields[$f$]
              end if
          end for
      end for
  end for
  OptimalPFT ← best_syntaxes[PFT.root]
```