Appendix: Technical Details on PairFlow and EarlyCrow Datasets

A PairFlow

EarlyCrow defines a novel multipurpose network flow format called PairFlow, which is leveraged to build the contextual summary of a PCAP capture, representing key behavioral, statistical and protocol information relevant to APT TTPs. We discuss the details of each component in the following.

A.1 Tracking

Packets Retrieving. The tracking module identifies all unique pair connections on the network and filters out those using non-IP protocols. For each unique pair connection, PairFlow tracks, bidirectionally, all packets related to a pair. These packets are designated with an initial Flow ID. The Flow ID holds unchanged for all packets during the same time window for a given pair connection. Each packet will maintain its individual index for the aggregation step later. Packets with the same Flow ID may also use different protocols. Therefore, each one has a one hot encoding flag called Encoding Protocol Flag (EPFLAG) used later for further filtering. These flags started with EPFLAG_Protocol, where a protocol is a subset of {TCP, UDP, DNS, ICMP, HTTP, SSL/TLS}.

DNS Requests and Responses. The tracked packets do not include DNS requests and responses, which are responsible for locating the IP address needed to establish a connection. That is due to the pair connection being between the host and the DNS server, which is different than the destination. Similar to [2], to track these DNS packets, a destination of the present pair will be used as a Local PTR to find all DNS response packets from the PCAP repository. Once found, the DNS response resource records will be used to find all related DNS requests. Now, Any packets belonging to the pair connection are attached and sorted according to their arrival time. Those packets outside of time window are not included.

A.2 Aggregation

Header Generation. Besides the individual packet id from the PCAP, every packet is also designated with a Flow ID composed of a ContextualSummary ID (CSID) and a PairFlow ID (PFID). The former is unique for the lifetime of a pair, while the latter is unique for a time window. Any packets from that PairFlow will always have the same Flow ID. To assign the PFID, the Aggregation module will check the ContextualSummary repository to find if the pair has been processed in the past. If so, the incoming PFID will be the last used PFID for the same pair and ContextualSummary ID, incremented by 1. Otherwise, a new and unique ContextualSummary will be created, and the PFID will start with 0.

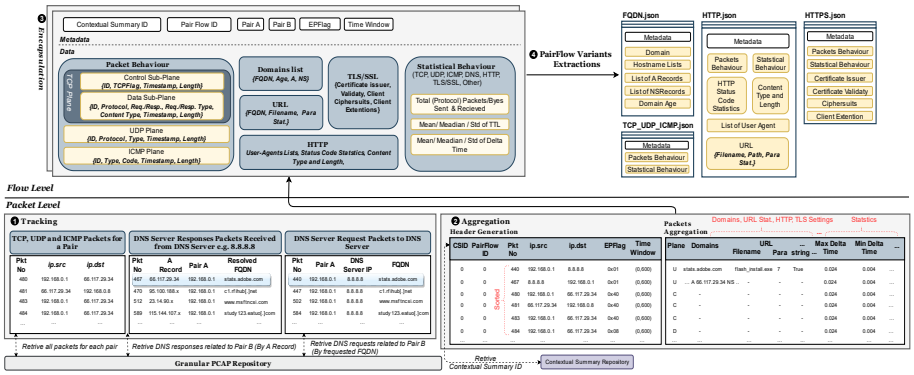

Fig. 1. Overview of the PairFlow workflow.

Packets Aggregation. The Aggregator module creates a PairFlow to store PairFlow ID, sorted packet index, pair connection, time window, EPFlag, FQDNs, URL, UAs, SSL/TLS settings, and initial flow-based statistics. The initial flow-based statistics include the number of protocol-based packets (i.e., TCP, UDP, ICMP, HTTP, SSL/TLS, DNS packets), total (encrypted) bytes, total (encrypted) bytes sent/received. Time-based statistics include packet Time to Live (TTL) and delta packets interarrival time max/min/median and the flow duration at the same time window. Similar to [4], we separate TCP packets into data and control packets to be used later in the encapsulation process. Finally, preprocessed flows are dispatched to the encapsulation step for further processing.

A.3 Encapsulation

The encapsulation phase explicitly groups packet behavior, FQDN and URL, HTTP(S) and initial statistical behavior implicit in preprocessed flows in order to make contextual information readily available. The data types involved include list of strings and tuples, Boolean and numeric fields, as shown in Table 1.

Packet Behavior Packet Behavior encapsulates all packets according to their protocol type (TCP, UDP, and ICMP) in a list of tuples. The first element is the packet index for traceability of a given packet inside the original PCAP for further investigation. The *TCP plane* involves the control and data sub-planes as shown in Figure 1. Each packet in the data sub-plane holds protocol name, request/response and their type, content type, timestamp, and packet length for each packet. For example, an HTTP request packet can be described as (460854, 'HTTP', 'Request', 'GET', 'Empty Content', 1066.51, 383) and its response (460895, 'HTTP', 'Response', 200, 'text/javascript', 1066.86, 429). This helps the upper system work on time series traffic and monitor the anomaly

for a given PairFlow. Further packet-level statistical analysis such as counting GET/POST, HTTP Response types, content analysis can be achieved as described in Section 3.3. The control sub-plane provides the behavior of the initial connections before the data exchange begins, the TCP continuation, or the termination of the TCP connection. For example, when TCP establishes a connection with three-way handshaking, it will summarize SYN, SNYACK, ACK packets as follows (72095, '0x02', 215.73 sec, 74),(72126, '0x12', 215.78 sec , 70 B), (72127, '0x10', 215.78 sec, 66 B). Then it will follow a stream of packets with TCP flag = 0x10 (ACK) until the connection is disconnected with flag FIN. This will be useful for analyzing any problem with time series or monitoring the discontinuity of such a PairFlow as we can see in Section 3.3. *UDP plane* records all UDP-based packets with protocol name, packet type, timestamp, and packet length. For example, if there are two packets for DNS which are request and response for a specific domain, they will be summarized as follows: (21160, 'DNS', 'DNS Request', 141.44 sec, 75 B), (21219, 'DNS', 'DNS Response', 141.54, 547 B). *ICMP Plane* is similar to the *UDP plane* but for the ICMP only. However, the type and code are reporting ICMP settings for each packet. The plane can be helpful for any classifier detecting ICMP-based attacks.

FQDN and URL As depicted in Figure 1, *Domain list* encapsulates all FQDNs related info in a list of tuples. Each tuple holds an FQDN, its A and NS resource records, and the domain age extracted from the WHOIS file. This helps malicious domain detectors, which often rely on FQDN name, relative DNS zone, and WHOIS files. *URL* encapsulates each relevant element of URL during a connection in a tuple which includes FQDN, web page filename,the number of parameters, values and fragments, and whether it contains encoded strings or not.

HTTP(S) *HTTP* encapsulates HTTP-level information for a given connection, in particular, distinct HTTP server names, status codes, content types and User Agents. *TLS Protocols* summarizes the security settings between a client and server. Cipher suites for both client and server are stored in a list. Cipher suites includes the key exchange/agreement (e.g. RSA, Elliptic-curve Diffie–Hellman (ECDH), Elliptic Curve Digital Signature Algorithm (ECDSA)) , authentication (e.g. RSA), block/stream ciphers (e.g. AES, RC4) with their block cipher mode (e.g. CBC) and message authentication (e.g. MD5, SHA-x). Extension types are also listed for each connection which summarizes the cipher suite settings such as extended master secret, session tickets, and Elliptic Curve (EC) point formats. Supported Groups are also stored, known as the EC setting (e.g., secp256r1, secp521r1).

Initial Statistical Behavior A few essential fields are important to be summarized statistically. We calculate max, min, mean packet TTL, delta packets interarrival time, and duration for a given PairFlow. We also calculate the

total (encrypted) bytes and the ratio of sent/received (encrypted) bytes. Max, min, median of cipher suites bytes, and server and client extension bytes are also calculated. We also provide a statistical summary of individual protocol number of packets such as raw TCP, raw UDP, ICMP, DNS, HTTP, TLS, and SSL. We summarize statistical fields in Table 1.

ID	Field	Type	ID	Field	Type
	I. Informative Fields				
1	Flow ID	N	17	HTTP Servers	LS
2-3	Source & Destination	S	18	Status Codes	LS
4	Packet Data Points	LT	19	Content Type	LS
5	EPFLAG	S	20-21	Server and Client Ciphersuit	LS
6-12	EPFLAG raw TCP, raw UDP, ICMP, DNS, HTTP, TLS and SSL	B	22-23	Server and Client Extension Type	LS
13	FQDN	LS	24-25	Client and Server Signature Algorithm and Hash	LS
14	Nameservers Resource Records	LS	26-27	Client and Server Supported Group	LS
15	A Resource Records	LS	28	ALPAN Next Protocol	LS
16	URL	LT	29	EC Point Format	LS
	II. Statistics Fields				
30	Total Bytes	N	44-48	TTL Max/Min/Mean/SD	N
31-32	Total Sent/Received Bytes	N	49-52	Delta Packets Interarrival time Max/Min/Mean/SD	N
33	Total Encrypted Bytes	N	53-56	Content Length Total/Max/Min/Median	N
34-35	Total Encrypted Sent/Received Bytes	N	57-59	Server and Client Cipher Suites Bytes Max/Min/Median	N
36-42	Number of raw TCP, raw UDP, ICMP, DNS, HTTP, TLS and SSL packets	N	60-62	Server and Client Extensions Bytes Max/Min/Median	N
43	Duration	N			

Table 1. Summary of PairFlow data fields. (B: Boolean, LS: List of Strings, LT: List of Tuples, N: numerical.)

A.4 Variants Extraction

PairFlow processing also exports four *variant* json files which can be used by any external classifier, as depicted in Figure 1. FQDN.json includes all domains and their hostname lists that have been accessed during a given PairFlow. In addition, resource records such as A, NS are also included and domain age extracted from WHOIS file, which appears to be useful for domain detection [1]. TCP-UDP-ICMP.json is dedicated for those classifiers use time-series for detection [4, 8]. All three planes are presented here in addition to related statistical fields such as packet TTL and delta packets interarrival time. HTTP.json is employed for those interested to detect malicious HTTP connections [8, 10]. Other classifiers may deploy HTTPS.json for detecting encrypted communications without deciphering the traffic [2]. The following section will discuss how EarlyCrow used mostly the HTTP variant. A detailed study of the other variants is left for future work.

B Dataset Troubleshooting

B.1 Causal Analysis

A link between a pair of edges can be causal or anticausal. A causal relationship is described by $P(Y|X)$ when $X \rightarrow Y$, where X is a sample and Y is the label. Literally, 'X is a direct cause of Y which means modifying the value of X will change the likelihood of Y. Causal links refer to predict effect (Y) from cause (X). In contrast, anticausal, $P(Y|X)$ when $Y \rightarrow X$, meaning anticausal links predict cause from the effect such as predict X (sample) from Y (label). Clearly, anticausal is out of our scope because it does not influence the data generating process of our dataset, which will use X (ContextualSummary) only on predicting Y (APTs, botnets and legitimate). Therefore, we build a causal model focusing on causal links as depicted in Figure 2. We attempt to balance between accuracy and clarity on the level of abstraction. We aim to identify the bais, including confounder and collider, in our dataset and control them. We will explain Figure 2 after introducing some related concepts.

Confounder may be an unadjusted common cause , while collider involves conditioning a common effect such as selection bias. Let us consider that A (say, PairFlow) is a common cause of B and X, Destination Profile and ContextualSummary, respectively. A is called a confounding factor, and it creates a spurious correlation between B and X, resulting in $B \perp\!\!\!\perp X$ (literally, "B is dependent on X"). However, they become independent when A is controlled: $B \perp\!\!\!\perp X|A$. Consider the instance when X is a common effect of both A and B. We call X as a collider in this case. Conditioning on X adds a connection between A and B, as they can now explain away the influence of each other on the observed result, X (i.e., $A \perp\!\!\!\perp B|X$).

Due to the various capture environments for our considered dataset, it is vital to control collider and confounder caused by data shift. The data shift in causal direction can be population, annotation, acquisition shifts, or sample selection bias. Let us assume Z is a vector of the PCAP raw features. *Population Shift* may occur when $P_{train}(Z) \neq P_{test}(Z)$. We ensure to mitigate such bias by focusing on samples that belong to the same protocol - in our case HTTP(S) - have to be used in each sample. *Annotation Shift* is given by $P_{train}(Y|X) \neq P_{test}(Y|X)$ where samples are labelled based on different metadata. For example, APTrace may include a malicious flow that is also presented in MCFP with a different label source. In this case, control measures can be taken, such as relabelling all flows by referring to the same security intelligent platform source. *Acquisition Shift* can occur due to different platform configurations, which may affect several features, including packet TTL, duration, and UA-based features. Data harmonization, such as extracting domain-invariant representations, is the foremost approach to mitigate such bias to validate $\vec{F} \perp\!\!\!\perp P_f$, where $\vec{F}$ is a feature vector, and P_f is the platform. *Sample Selection Bias* is when $P_{train}(X,Y) \not\equiv P_{test}(X,Y)$, meaning the training set does not represent the target population. We can avoid it with a random sample selection with stratified training and testing split for each class.

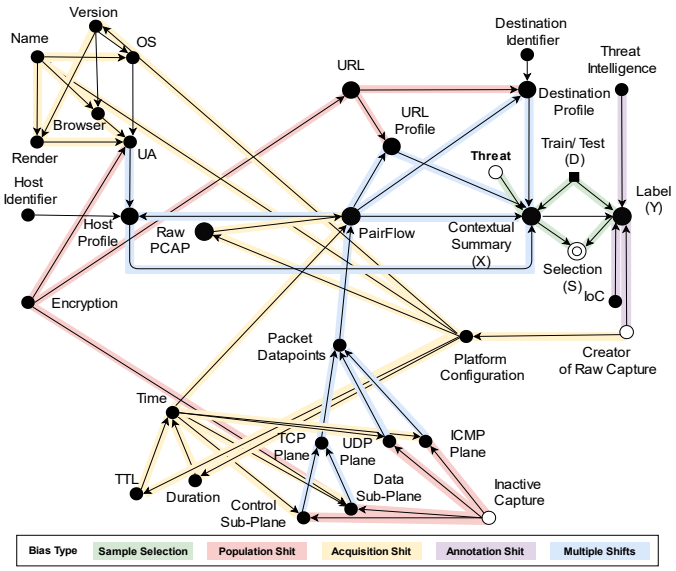

Fig. 2. Causal model diagram. Empty circles denote hidden variables, whereas solid circles refer to explicit ones.

In Figure 2, we point out all possible cause and effect links, where some of which are shaded with specific colors reflecting the type of potential bias as annotated in the legend box. Y (Label) is a common effect of three causes, threat intelligence, IoC, and creator of raw capture. Y is a collider and the potential bias is the *annotation shift* because we may have conflict among labels based on different sources. The platform configuration is a common cause of UA, Time, and Raw PCAP variables. In this case, we call the platform configuration variable a confounder that produces an association between all three variables. The type of shift between these variables is *acquisition shift* due to different platforms that we collect our captures from. The encryption variable is also a confounder. It is a common cause of UA, URL, control, and data sub-planes. For instance, UA and URL are hidden inside the payload when encryption is applied. The details of the data sub-plane are not available without decryption, leaving the data plane with SSL/TLS packets. The potential bias is the *population shift*. Another population shift is caused by the inactive capture variable, which is also a confounder of control, data, UDP, and ICMP planes. When a capture is collected, but there is no sign of attacks or communication with C&C it affects the behavior of these planes to act as legitimate or relatively low number of malicious packets exchange. When two or above shifts are possible, we call such link as *multiple shifts*. However, mitigating the individual bias will reduce the multiple shifts.

B.2 Control Measures

We take several control measures to improve our datasets motivated by the causal model reported in Figure 2.

Population shift: We exclude any flows that do not use HTTP from the inputs to the PairFlow-HTTP variant. We limit all captures to the same duration for the duration-related features to remove the bias of packet statistics and the duration of a flow itself. For instance, it is biased to compare a flow captured for 15 minutes to the other during 2 hours, which may influence the statistics for the benefit of a longer capture. We also consider recently recommended practices to troubleshoot datasets for IDS [7]. Therefore, we removed samples without data transfer or failed TCP connections to C&C servers. *Annotation Shift:* The sandbox already generates an IoC file in APTrace. However, we label those missed via sandbox by manually checking every IP and domain using other threat intelligence platforms and APTs industry reports following the recommendation by [3]. For MCFP, we label the flows based on the original project and manually label them using historical data retrieved from security intelligence platforms. To ensure annotation shift is mitigated, we validate all labels for datasets by *IBM X-Force Exchange*[1] and *AlienVault Open Threat Exchange*[2]. Finally, we need to remove the bias for the Host profile in EarlyCrow through various captures. Instead of considering source IP as a host identifier, we used a tuple of $(multiple\ labels, source)$ to avoid mixing data with the same internal IP but in a different network or capture. The multiple label field refers to the filename of a PCAP capture. *Sample selection bias:* To remove the bias caused by APTs only as a malicious class, we added botnets C&C captures to measure if our system can detect both activities. Then we stratified the training and testing process to ensure $P_{train}(X, Y) \equiv P_{test}(X, Y)$. In Section 4, we report the weighted and macro average F score due to the binary labels' imbalanced samples and shed light on malicious flow detection. *Acquisition Shift:* The differences in platform configuration cause bias. We consider only features with domain-invariant representations to ensure $\vec{F} \perp\!\!\!\perp P_f$. Therefore, we neglect features on names and versions of OS, Browser, and renderer. We keep the distinct number of UA per host and its popularity-based features. Packet TTL is another domain variant, which is dependent on the platform. We omit features based on TTL, although we kept it available in PairFlow for the research community but not for our assessment in the next section.

References

1. Alageel, A., Maffeis, S.: Hawk-eye: Holistic detection of apt command and control domains. In: ACM SAC. pp. 1664–1673. ACM (2021)
2. Anderson, B., McGrew, D.: Identifying encrypted malware traffic with contextual flow data. In: ACM AISec. pp. 35–46 (2016)

[1] https://exchange.xforce.ibmcloud.com
[2] https://otx.alienvault.com

3. Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L., Rieck, K.: Dos and don'ts of machine learning in computer security. In: USENIX Security (2022)
4. AsSadhan, B., Moura, J.M., Lapsley, D., Jones, C., Strayer, W.T.: Detecting botnets using command and control traffic. In: IEEE NCA. pp. 156–162. IEEE (2009)
5. Bartos, K., Sofka, M., Franc, V.: Optimized invariant representation of network traffic for detecting unseen malware variants. In: USENIX Security. pp. 807–822 (2016)
6. Bilge, L., Balzarotti, D., Robertson, W., Kirda, E., Kruegel, C.: Disclosure: detecting botnet command and control servers through large-scale netflow analysis. In: ACSAC. pp. 129–138 (2012)
7. Engelen, G., Rimmer, V., Joosen, W.: Troubleshooting an intrusion detection dataset: the cicids2017 case study. In: IEEE SPW. pp. 7–12. IEEE (2021)
8. Hu, X., Jang, J., Stoecklin, M.P., Wang, T., Schales, D.L., Kirat, D., Rao, J.R.: Baywatch: robust beaconing detection to identify infected hosts in large-scale enterprise networks. In: IEEE/IFIP DSN. pp. 479–490. IEEE (2016)
9. Invernizzi, L., Miskovic, S., Torres, R., Kruegel, C., Saha, S., Vigna, G., Lee, S.J., Mellia, M.: Nazca: Detecting malware distribution in large-scale networks. In: NDSS. vol. 14, pp. 23–26 (2014)
10. Oprea, A., Li, Z., Norris, R., Bowers, K.: Made: Security analytics for enterprise threat detection. In: ACSAC. pp. 124–136 (2018)
11. Oprea, A., Li, Z., Yen, T.F., Chin, S.H., Alrwais, S.: Detection of early-stage enterprise infection by mining large-scale log data. In: IEEE/IFIP DSN. pp. 45–56. IEEE (2015)
12. Perdisci, R., Lee, W., Feamster, N.: Behavioral clustering of http-based malware and signature generation using malicious network traces. In: NSDI. vol. 10, p. 14 (2010)
13. Tegeler, F., Fu, X., Vigna, G., Kruegel, C.: Botfinder: Finding bots in network traffic without deep packet inspection. In: CoNEXT. pp. 349–360 (2012)

ID	Feature	New?	ID	Feature	New?
	I. PairFlow Based Features				
1	Total Bytes	[5, 6, 13]	28-31	No and ratio below and above AVG	✓
2	Sent/Received Ratio	[5, 6, 13, 10]	32-33	No and ratio outliers	✓
3-9	Ratio of raw TCP, raw UDP, ICMP, DNS, HTTP, TLS and SSL packets	✓	34-37	outliers Magnitude Max/Min/Mean/SD	✓
10-13	Ratio of HTTP Response packets with 2xx, 3xx, 4xx, 5xx	[10]	38-40	Data packet exchange idle time Max/Min/Mean	✓
14-15	Ratio of frequent GET and Post	[12, 10]	41	Active duration	✓
16-19	Content Length Total/Max/Min/Median	✓	42-45	Packet TTL Max/ Min/ Mean/ SD	✓
20-26	Ratio of Content Type Javascript, HTML, Image, Video, App, Text, Empty	[10]	46-49	Delta packets interarrival time Max/ Min/ Mean/ SD	Similar to [5]
27	No of resumed connections	✓	50	No of DNS request	✓
	II. Host Profile Features				
51-53	Max/Min/Mean Time Difference of Sequenced Connections	✓	59	Distinct UAs per host	✓
54	Ratio of Connected Destination IP only to FQDN	✓	60	nAvg UA Popularity	[11]
55-57	Max, Min, Mean of Resumed Connections per flow for a host	✓	61-62	Frac UA 1, 5	[11]
58	No of DNS request per flow for a host	✓	63	Ratio UAs	[11]
	II. Destination Profile Features				
64	No. of hosts connected to destination	[11]	72	No. of Distinct URLs associated to destination	✓
65-67	Destination Received /Sent Max/Min/Average	✓	73-75	Destination Max/Min/Mean Packets Failure	✓
68-70	Destination Data Packet Exchange Idle Time Max/Min/Mean	✓	76-81	Max/Min/Mean No and ratio of DNS request per flow for a destination	✓
71	No of resumed connections per flow for a destination	✓			
	II. URL Profile Features				
82	Frac URLs filename	[10]	94-96	URLs Values Max/Min/Mean	[5, 12, 10]
83	Frac URLs filename exe	✓	97-99	URLs Fragments Max/Min/Mean	[10]
84	Number of distinct extensions	[10]	100	Frac query	[10]
85-87	URLs Length Max/Min/Mean	[5, 12, 8, 10]	101	Num hasString	✓
88-90	URLs Depth Max/Min/Mean	[5, 12, 10]	102	Num of URLs and Distinct ones	[9]
91-93	URLs Parameters Max/Min/Mean	[5, 12, 8, 10]			

Table 2. EarlyCrow features. Note that features reused from the literature are computed from PairFlow data rather than from other data formats.