EarlyCrow: Detecting APT Malware Command and Control over HTTP(S) Using Contextual Summaries

Almuthanna Alageel[1,2](✉) and Sergio Maffeis[1]

[1] Department of Computing, Imperial College London, London, UK
{a.alageel18,sergio.maffeis}@imperial.ac.uk
[2] National Center for Cybersecurity, King Abdulaziz City for Science and Technology, Riyadh, Saudi Arabia

Abstract. Advanced Persistent Threats (APTs) are among the most sophisticated threats facing critical organizations worldwide. APTs employ specific tactics, techniques, and procedures (TTPs) which make them difficult to detect in comparison to frequent and aggressive attacks. In fact, current network intrusion detection systems struggle to detect APTs communications, allowing such threats to persist unnoticed on victims' machines for months or even years.

In this paper, we present EarlyCrow, an approach to detect APT malware command and control over HTTP(S) using *contextual summaries*. The design of EarlyCrow is informed by a novel threat model focused on TTPs present in traffic generated by tools recently used as part of APT campaigns. The threat model highlights the importance of the context around the malicious connections, and suggests traffic attributes which help APT detection. 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 evaluate the effectiveness of EarlyCrow on unseen APTs obtaining a headline macro average F1-score of 93.02% with FPR of 0.74%.

Keywords: Advanced persistent threats · Network intrusion detection · Command and control

1 Introduction

Advanced Persistent Threats (APTs) are known to be the most sophisticated long-term attack campaigns targeting highly protective organizations [2]. APTs are generally aware of internal defenses related to their target [42], and usually do not send spam, participate in DDoS attacks, or aggressively propagate to other hosts to spread infections at scale [22].

APT malware are those malicious tools known to be used by APT campaigns. The most common is the Remote Access Trojan (RAT), typically composed of a builder, stub, and controller. The builder initiates a new instance

W. Susilo et al. (Eds.): ISC 2022, LNCS 13640, pp. 290–316, 2022.
https://doi.org/10.1007/978-3-031-22390-7_18

stub upon the infection. The stub runs on the victim machine and contains a hard-coded Fully Qualified Domain Name (FQDN) or IP to communicate to the RAT controller, which resides on the Command and Control (C&C) server [41]. Rootkits, spyware, downloaders, and keyloggers may also be part of an APT campaign. APT malware such as DarkComet includes these functions in one ecosystem [22], which may capture the audio, explore files and drop malicious tools through visiting URLs [23]. Griffon, used by FIN 7, can gather information, load Meterpreter, and take screenshots [28]. Hutchins et al. [31] propose a kill chain to defend against APTs at various stages, including reconnaissance, weaponization, delivery, exploitation, installation, and C&C. These stages normally iterate over a long time [35]. In order to limit the damage inflicted by an APT, it is essential to detect them at an early stage, and in particular as they establish communication with the C&C. By inspecting honeypot data, we find that the communication to C&C starts immediately once the machine is infected. Several automated tasks are performed, including establishing *fallback channels* and downloading further payloads from the C&C server. These activities intentionally behave as legitimate web browser activities, attempting to evade Network Intrusion Detection Systems (NIDSs).

In Sect. 2, we introduce the first public measurement study of APT malware C&C communication to investigate the deployed TTPs. We leverage our measurements to identify the features necessary to recognize such TTPs at the network level, and compare them with existing features from the literature. We found that the use of evasive TTPs leads to significant overlap with legitimate behavior, confusing the decision boundaries based on some known features. Based on this analysis, we build EarlyCrow, a tool to detect APT activity in network traffic. EarlyCrow generates four sets of data focused on connections, hosts, destinations, and URLs. Features from these sets are grouped to form a ContextualSummary. The ContextualSummary has multidimensional features that help in building more informative random forest trees used for classification as described in Sect. 3. We evaluate EarlyCrow on traffic from APT malware excluded from the measurement study and training set in order to test generalization and mimic a real-world scenario (Sect. 4). Fresh malware samples are also investigated to confirm the feature importance identified by our measurement study on the training set. We also investigate how the performance of EarlyCrow is affected by different deployment scenarios, where it has visibility on HTTP traffic or where it can only observe opaque HTTPS traffic.

In summary, our main contributions are:

- We present an evidence-based analysis of various TTPs used by APTs. These TTPs are known to be used to evade NIDS [14]. We also introduce a measurement study on various APT malware over popular and novel features to capture TTPs usage.

- We implement EarlyCrow[1], a tool to detect evasive malicious communication over HTTP(S). EarlyCrow focuses primarily on APTs but is also effective against stealthy botnets.
- We evaluate the classification performance of new and existing features for malicious traffic detection under different scenarios distinguishing ATP, botnet, and legitimate traffic.

2 Threat Model

Defining a relevant threat model, and focusing on a narrow set of attacks are recommended best practices when proposing a novel NIDS [43]. There are several ways to approach threat modeling for APTs at the network level. We consider four popular cases of APT that involve HTTP(S) traffic, each of which deploys at least one C&C TTP. In Case I, the infected machine contains APT malware with a hard-coded FQDN. The malware issues a DNS query to resolve the FQDN to an IP address. The subsequent communication to the C&C server can be via HTTP or HTTPS. After that, the malware may initiate a *fallback channel*, another popular TTP used by APTs [19], using either of the strategies described in Cases I–IV, only this time no longer for the initial communication. In Case II, the APT malware connects to a URL whose domain component is a hard-coded IP address, in order to bypass malicious domain detectors, and its fallback channel can be established using the *DNS over HTTPS* (DoH) TTP [21], as in CobaltStrike [33], which is used by SUNBURST [24]. Case III is similar to Case I in using a hard-coded FQDN, but the subsequent communication uses raw TCP rather than HTTP during the malicious operation. Case IV is similar to Case II in using direct IP without DNS resolution, but then uses raw TCP communication as in Case III. Both Case III and IV may use fallback channel with various TTPs, although not including those related to HTTP(S).

Additional TTPs introduced by MITRE and relevant to APTs can be combined with the use of a fallback channel: *web protocol* [13] where an adversary may use HTTP to avoid network filtering and mimic legitimate and expected connections, *non-application protocols* [20] such as Raw TCP, UDP or ICMP, *encrypted channel* [18] to hide C&C malicious content, *fast flux* [17] is a subtechnique of *dynamic resolution* to obtain different IPs for the same FQDN, and *data obfuscation through protocol impersonation* [15] to impersonate legitimate use of HTTP or to mimic a trustworthy entity using a fake SSL/TLS certificate.

This paper focuses on Case I and II, where at least one malicious HTTP(S) connection exists between the infected host and C&C server. Other cases are challenging to detect with low False Positive Rate (FPR) at the network level only, and require additional host-level logs.

2.1 TTP Relevant Data

The TTPs considered here are a group of host and network-level techniques used by APTs to evade Host and Network IDSs. To track TTPs at the network level,

[1] EarlyCrow code, datasets, and experiments are publicly available at [1].

an investigator needs to collect "static" Indicators of Compromise (IoCs) for known APTs, or analyze sequences of network packets and assess the likelihood of specific TTPs manifested by the traffic behavior. Security vendors publish IoCs of discovered APTs. Novel attacks can be discovered when suspicious TTPs are being observed, as for instance in the HTTP request and response behavior.

IoC-Like Data. APT campaigns dedicate one or more *FQDN(s)* to locate C&C servers. They may mimic the targeted organization interests, or use *dynamic resolution*, which is another TTP [16] used to communicate back to C&C servers [3,22]. The resolved FQDN holds at least one A resource record. Some APTs provide several A resource records to provide fallback channels for follow-up connections. *URLs* are known to be used as IoCs, and used in HTTP-based malware detection [7,34,38,40]. Some APT malware download an executable file or pass other malicious FQDNs, IPs, or configuration commands in URL parameters of subsequent requests. A typical URL structure includes FQDN, nested folders (which we will refer to as *depth*), filename, parameters and values with a delimiter (&) to separate between them and (=) to assign value to the parameter, and encoded strings which typically contain %-encoding.

Traffic Data. Although multiple traffic-based TTPs were used by APTs in the past, it is challenging to capture them by configuring NIDS with straightforward rules. For this reason, we need to consider the context where malicious packets are sent. First, we need to cover the details of HTTP requests and responses, and then the traffic behavior of all protocols used for the same flow. *HTTP request and response context* involves consecutive HTTP transactions composed of several requests and responses. A request is mainly characterized by the URL, method type (e.g. GET, POST) and User-Agent (UA). Response headers specify among other properties, the content type and status codes. To detect APT malware, we need to efficiently store that information between two endpoints in one flow and enable the NIDS to extract valuable statistics at the packet level.

Due to the stealthiness and low-profile operation of APTs, we also need to provide a way to investigate *Traffic Behavior*. This can be achieved by storing packets arrival times, their lengths and other related information. Such a summary needs to cover the control and data planes of TCP, UDP and ICMP packets. With this summary on data points, NIDS designers can catch APTs TTPs such as *fallback channel* and using *non-application protocols*. For instance, a host contacting three different destinations after only one DNS query can be a sign of infection by the fallback channel technique. Another example is the *non-application protocol* TTP, when the APT malware opens a legitimate looking HTTP connection which is followed by a sequence of malicious raw TCP packets.

2.2 Measurements

We provide several measurements taken on the training set summarized in Table 2 and described in Sect. 4. Since our objective is to detect APTs at the early stage, all measurements are observed during the first 15 min of each connection. In Sect. 4.2, we will investigate these measurements and other proposed features, to see if they generalize to unseen malware.

Traffic Statistical Measurements. Statistical end-to-end observations may highlight the evasive behavior of APTs compared to legitimate actors. The presence of a slight deviation may reflect malicious use of three TTPs, including *non-application protocols*, *data obfuscation through protocol impersonation* and *web protocol*. Since this study focuses on malicious HTTP(S) usage, we measure the HTTP packets ratio across all classes. Other related protocols are also measured, including raw TCP and DNS ratios. Legitimate connections show a positive linear relationship between DNS and HTTP packets (Fig. 1). With every additional page requested by a user, such packets are exchanged with a remote web server in order to fetch additional resources. For APTs, we notice that DNS ratios are half or less than for legitimate or botnets, respectively. 95.2% of APTs do not exceed a 0.19 DNS ratio, compared to 0.38 for legitimate and 0.46 for botnets.

Next, we focus on DNS requests and conclude that almost no malicious behavior exceeds the legitimate, except for Conficker botnets, which use Domain Generation Algorithms (DGAs). 84% of APT or botnet traffic issues 2 or 6 requests at most, while legitimate traffic can generate up to 18. Once a domain is resolved to one or more IPs, a typical APT avoids requesting another DNS for the rest of HTTP communication unless they plan to establish another *fallback channel*. Another useful feature is the raw TCP ratio, which helps detect the *non-application protocols* TTP: a high ratio indicates the adversary uses HTTP as camouflage while still heavily relying on raw TCP. It is extremely rare for an APT to have a raw TCP ratio lower than 48.84%, whereas we observed minimum ratios of 2% of legitimate, and 0% of botnets.

Since we focus on the early stage of connections originating from the victim side, we found that around 70.58% of APTs receive 3.35 times more data than they send to the remote server, compared to 1.45 and 0.75 for legitimate and botnets, respectively. This is consistent with the threat model in [3], where an adversary uploads more tools on the victim's machine at the beginning of an APT campaign to continue other operations such as lateral movement, unlike botnets which may show more data exfiltration behavior. We also examine the number of resumed connections. Legitimate HTTP usage typically increases the number of resumed connections, since shortly after a web resource is downloaded, the TCP connection is terminated with FIN. Upon clicking another link, even for the same website, a new TCP three-way handshake is initiated. We count that as a resumed connection. With a web caching service, the scenario remains similar, although the server is contacted via a proxy or content delivery network (CDN). While legitimate and botnets connections may easily be resumed up to 21 times, APTs tend to terminate less (roughly 50% less). It seems plausible that APTs avoid frequent connection termination and resumption to increase stealthiness.

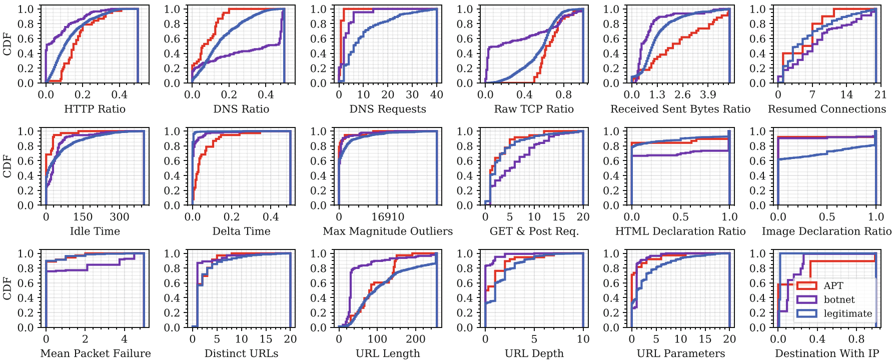


Fig. 1. Measurements for APT, botnets, and legitimate connections.

Time-Based Measurements. We measure *stealthiness and low profile* of APTs by monitoring time-based features. *Delta*, the packet inter-arrival time between a remote server and a host, is estimated based on the arrival time difference between packets, independently from their protocol. For 94.73% of cases, we found the mean delta time in seconds to be at most 23.5×10^{-2} for APTs, 6×10^{-2} for botnets and 0.5×10^{-2} for legitimate. Hence, APTs may act slower than botnets and legitimate by up to 4 and 47 times respectively.

A new metric, data packet exchange *idle time*, is proposed to measure the time difference between actual data packets. We found APTs idle time to be 3 and 6.57 times shorter than botnets and legitimate: 92% of cases have an idle time of at most 28, 84, and 184 s, respectively. Once APTs establish a communication channel, they send bursts of data packets (low idle time), then pause communications (high delta) until the next burst. We also measure the maximum magnitude of outliers which exceed the Simple Moving Average (SMA) with respect to the predefined bins described in Sect. 3.3. We found that for 84% of the cases, the maximum magnitude for APTs (0.338 KB) is half the one for botnets (0.676 KB), and 10% of the one for legitimate (3.33 KB). These three time-based features partially capture the *low and stealthy profile* of APTs compared to botnets or legitimate.

Remote Web Server. Analysis of contacted web servers may help identifying the *web protocol* and *fallback channel* TTPs. Typical web servers mostly adhere to best practices in setting up their HTTP configurations. APTs appear to be more professionally configured than botnets, but not as much as legitimate ones. For instance, the *packet failure* rate for legitimate servers and APTs (HTTP responses with status codes 4xx and 5xx) is relatively low. To be precise, 90% have at most one packet failure, while the botnets may receive as many as five. Total GET and POST requests are less similar. 92% of APTs and legitimates have 9 and 10 or less, respectively while the botnets have up to 14.

We also investigate the ratios of content types declarations. We focus on the ratios of HTML and images, since these are most frequently used in HTTP connections. 73% of APTs, legitimate and botnets declare HTML 2%, 2% and 98% of the time, so APT behavior in this case is similar to legitimate. However, due to the possible use of the *data obfuscation through protocol impersonation* TTP, we found that APTs and botnets are less likely to declare image type, which is not the case for web browsing activities. 70% of legitimate declare images 30% at most during a connection, while it is zero for both APTs and botnets.

Next, we measure the URL characteristics, due to their proven effectiveness for detecting malicious web servers. Measuring the distinct URLs accessed in a given network may highlight the rich number of web pages which is more likely to be legitimate [30,38]. We observe that APTs invest heavily in legitimate-looking pages, to evade NIDS that rely on URL-based features. For example, we find that 87% of botnets query only one URL, while legitimate and APTs query up to five and four, respectively. APTs have more resources than botnets in general. As depicted in Fig. 1, 90% of APTs have 3 nested folders (depth), close to legitimate, which is 4, while botnets have 1 at most. URL parameters differ even more: 87% of APTs and legitimate use 3 and 7, while botnets use only 1. Following that, URL length is determined by the length of FQDNs, depths, filenames, parameters, values, fragments, and strings. 90% of legitimate URL lengths are 249 or less, whereas APTs and botnets are up to 145 and 109. Finally, APTs deploy a fallback channel in several ways, as discussed in Sect. 2. We measure the number of HTTP(S) connections established to an IP without a previous domain resolution. 57.89% of APTs reached 32% of C&C with IP only, while it is 9% and 1% for botnets and legitimates. Therefore, it is unusual for legitimate to perform such behavior, while it is more common for APTs and occasional for botnets.

3 EarlyCrow

EarlyCrow detects malicious HTTP(S) connections, and in particular APT malware. In this section we discuss the architecture of EarlyCrow, and how the features used by EarlyCrow are extracted and updated.

3.1 Architecture Overview

EarlyCrow is composed of four main processes, as depicted in Fig. 2. First, it starts with buffering and dispatching using PairFlow (Fig. 2, ❶), which summarizes a PCAP into contextually relevant fields including packet behavior, domain and URL list, UA, status code, and content type for HTTP. After the PairFlow HTTP variant is generated, these flows are preprocessed for profile pivoting (Fig. 2, ❷) to generate three profiles: Host, Destination, and URL. Then, two types of feature extraction follow (PairFlow and profile features in Fig. 2, ❸) to form a ContextualSummary (Fig. 2, ❹) which is the input for a random forest classifier. When another PairFlow is received, it will follow the

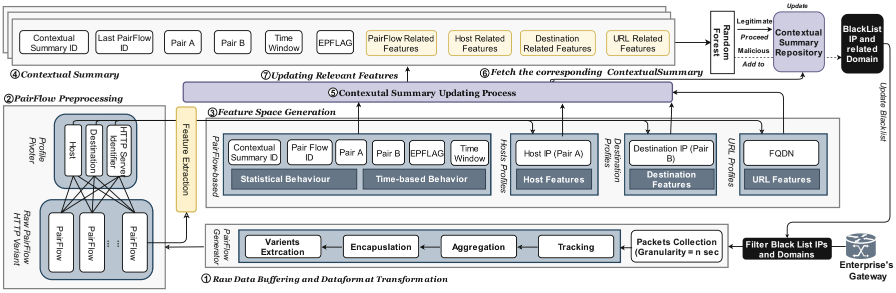


Fig. 2. Overview of the EarlyCrow architecture.

same workflow. A further step is required when the new PairFlow matches one of the previous ContextualSummary ID in the repository. The ContextualSummary updating process (Fig. 2, ❺) is responsible for updating the matched ContextualSummary to maintain the contextualization and reclassify again. The rest of this section discusses in detail the feature space generation and how the ContextualSummary is formed.

3.2 PairFlow

PairFlow is a proposed data format that allows the NIDS designer to quickly pivot flows into many profiles such as host, destination, and URL profiles. PairFlow data can also be used by detectors of malicious domains or IPs. Instead of detecting one flow according to the initiation and termination of TCP, protocol-based or time window, PairFlow digests all information to extract features later based on the whole context over time.

PairFlow receives raw PCAP data and stores these packets in a buffer until a time window of size t has passed. The buffer sends the current granular data with all the connections of a network during a time window, to the *Tracking* module to group unique pairs and label related packets. A unique pair refers to any (possibly bidirectional) connection observed between a host on the local network and a remote server. We take the *source* of the pair to be the local host, and the *destination* to be the remote server. Next, the *Aggregator* module adds a PairFlow ID and time window to the flow data. The Aggregator module is also responsible for marking packets according to their plane, extracting the domains and HTTP fields. Next, the *Encapsulation* module groups all these pieces of information contextually, so that all possible TTPs discussed in Sect. 2 can be analyzed later. Therefore, each pair of connections has a comprehensive description of their packets behavior (described in Sect. A.3), HTTP settings, accessed domains, and cipher suites setting. Finally, PairFlow outputs four additional JSON files which can be used by any external classifier. We only use the HTTP variant for EarlyCrow. The technical details for each component of PairFlow can be found in Appendix A.

3.3 PairFlow Features

EarlyCrow benefits from using the statistical features produced by PairFlow, which are presented in Appendix A and Table 5. It also extracts higher-level contextual features from the TCP and UDP planes.

Statistical Behavior. As we found in Sect. 2, where raw TCP ratio may reveal the *non-application protocols* TTP. We count the raw TCP ratio per PairFlow in addition to other protocols ratios such as DNS, which can also detect APTs malicious use of HTTP because it tends to request a domain resolution one time during a connection [3]. From Data Sub-Plane in Fig. 5, we calculate GET/POST requests and the fraction of status codes started with 1xx, 2xx, 3xx, 4xx, 5xx to identify the most salient behavior of such a connection. Using the control sub-plane, we count the termination of TCP connection FIN-ACK (0x11) during a PairFlow instead of the sequence of TCP handshaking, i.e., SYN, SYN-ACK, ACK (0x02, 0x12, 0x10) due to the lower computation cost. However, to exclude a typical HTTP flow (e.g., browsing sessions) and reduce false positives, we consider also the number of DNS requests during a given PairFlow, using the UDP plane. EarlyCrow calculates the number of declarations of content types and their ratios to the others in the data sub-plane. Examples of considered types include JavaScript, HTML, image, video, application, and text.

Time-Based Behavior. The challenge of time-based features is to identify APTs connections that operate at low-profile mode. First, we consider using a couple of time-based features from the PairFlow such as packet TTL, duration of the PairFlow, and delta packets inter-arrival time. We also measure the max/min/mean data packet exchange idle time using the data sub-plane, the difference between subsequent data packets' arrival time. During a typical web browsing session, there is little or no difference between delta packet inter-arrival time and data packet exchange idle time.

We propose additional time-based features that attempt to measure the stealthy behavior with time-series techniques. We present features based on the simple moving average (SMA). The purpose of the SMA is to average the data points over a time window of size t decided in advance, so that an analyst can identify when a data point is above or below such average. SMA_k can be described as follows: $\frac{1}{k}\sum_{i=n-k+1}^{n} p_i$, where p is the packet length, k is the number of previous data points in a time window, and n is the current data point. Since packets arrive asynchronously, in order to calculate an SMA we need to introduce a sampling rate such that packets arriving within two sampling events are combined together in a single point. For example, if the time window is one minute and we sample points every second then $k = 60$ and if we receive two packets of length respectively 128 and 32 between seconds 5 and 6, then $p_6 = 160$. After calculating the SMA, we can extract the number of outliers and their ratio and magnitude. Outliers are those points two times above the corresponding SMA_k. Therefore, we can capture the stealthy behavior of APTs, which has fewer outliers than legitimate and botnet traffic. However, it is also essential to find the number of packets below and above average. These features can capture the APTs that touch or slightly exceed the SMA, reflecting cautious operation.

3.4 Profiles Features

Profiles features are generated based on all PairFlows with longer time windows, for example lasting days, weeks, or even months. EarlyCrow queries the related information using a host IP, destination IP, and FQDN for the host, destination, and URL profiles, respectively. The purpose of the host profile is to identify whether that host has a sign of infection, such as discrepant information or a fallback channel. The destination profile may reflect those destinations that an enterprise can access and avoid some false positives. The URL profile helps identify the typical use of a given FQDN. FQDNs commonly accessed without parameters or values, especially with GET as method type, could signal the use of the *dynamic DNS* or *fast flux* technique to point to frequently changed IP addresses known to be used for APTs [3]. The URL profile helps pinpoint the malicious use of HTTP protocol from their past behavior. Nevertheless, APT cannot be easily detected based on such single features, so these will only contribute in part to the final classification.

Host Profile. The host profile aims to investigate the effect of infection on a machine behavior over $\hat{t}$ time, which should be longer than the selected granularity t time for PairFlow. Benign hosts should have specific characteristics in terms of resumed connections, DNS requests per flow, time difference of sequence connections, and type of UA used. When a host is infected with APT malware, its characteristics may move to another point further from the benign host centroid. For instance, it is suspicious for a host to initiate a connection by IP only, which is highly linked to a *fallback channel.* EarlyCrow investigates the number of resumed connections per flow for each host. Similarly, we extracted the DNS request per flow to identify a host with lower DNS requests than expected, which is also a sign of APTs using *dynamic resolution, DGA, and data obfuscation through protocol impersonation* TTPs.

In addition, we measure the Mean Time Difference of Sequenced Connections (MTDSC), which can help to identify *fallback channel.* MTDSC can be calculated as follows: $\frac{1}{n}\sum_{i=0}^{n} t_{i+1} - t_i$, where n is the number of new connections and t are their timestamps. The input timestamp should be the first packet sent or received from Control, UDP, or ICMP planes for any PairFlow, where the source is the same host. We also compute a ratio of connected destinations using IP only to those with FQDN. The feature can capture the APTs behavior of using DNS requests to locate the IP address of C&C; once the first channel is established, APT malware sends another IP as a fallback channel and starts another three-way handshake. It can also indicate the malicious use of *DoH.* As pointed out in Sect. 2, any client that uses HTTP will have an optional UA in a request packet, and it could be a (non-)browser, malicious string, or just an empty. Similar to [38], we extract several features for UA, including the distinct number of UAs and their popularity among an enterprise.

Destination Profile. The destination profile analyzes the servers contacted by internal hosts to find the characteristics of the provided services. We are interested in determining if it is normal for a destination to have fewer/more DNS requests, short/long data packet exchange idle times, high/low packet failure rates, sending/receiving dominant, and high/low resumed connections.

For instance, we measure the number of DNS requests per flow for a destination to investigate if such destination is using *dynamic resolution, DGA or data obfuscation through protocol impersonation* TTPs. An APT destination tends to have fewer DNS requests than usual. Once the domain is resolved and TCP establishment has been completed, it is rare to request more DNS packets. The legitimate use of HTTP(S) is to query the DNS packet every time they visit each page. Therefore, the number of DNS requests is directly proportional to HTTP packets. It is also essential to measure the destination data packet exchange idle time to identify legitimate web servers with a reasonable time to be idle for browsing. Again, the data packet exchange idle time here focuses only on the meantime of zero data exchange packets from a destination point of view without considering the control ones.

As pointed out in Sect. 2, some APTs use *protocol impersonation* such as HTTP as a camouflage to communicate with C&C. Thus, identifying the packet failure for each destination can explain if the failure comes from the destination itself or the PairFlow in Sect. 3.3. The objective is to find if a destination mimics web browsing activities while mainly communicating with the victims through raw TCP as *non-application protocol*. Another important aspect for each destination is calculating the number of resumed connections. Browsing behavior has frequently more resumed connections than the APT ones as we presented in our measurement study (Sect. 2.2). Finally, we observe the number of hosts connected to each destination. Popular web servers and botnets destinations are routinely contacted by a considerable number of hosts. In contrast, APTs typically infect as few as possible hosts, hence receiving few connections to their destinations.

URL Profile. We present URL-based features which are separated from those in the destination profile, as many FQDN-based URLs share the same IP or vice versa. The URL profile summarizes the standard behavior of resources and the traffic statistics for each FQDN or IP-based URL. We count here how many URLs are reached during a connection and how many are distinct. A malicious C&C server typically has fewer than a legitimate one (Sect. 1).

We also check if a URL has a query string, filename, and whether it has an executable extension, then calculate the fraction of the number of each field compared to the distinct number of URLs. A legitimate URL is likely to possess a filename with a variety of extensions. Other statistical features, i.e., Min/Max/Mean, are also calculated on URL length, depth, number of parameters, values, and fragments.

Table 1. EarlyCrow features. Note that features reused from the literature are computed from PairFlow data rather than from other data formats.

ID	Feature	New?	ID	Feature	New?
		I. PairFlow features			
1	Total bytes	[7,8,46]	28–31	Number and ratio below and above average	✓
2	Sent/received ratio	[7,8,38,46]	32–33	Number 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	[38]	38–40	Data packet exchange idle time Max/Min/Mean	✓
14–15	Ratio of frequent GET and POST	[38,40]	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, and Empty	[38]	46–49	Delta packets interarrival time Max/ Min/Mean/SD	Similar to [7]
27	Number of resumed connections	✓	50	Number 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	Inverse average of UA popularity	[39]
55–57	Max/Min/Mean of resumed connections per flow for a host	✓	61–62	Fraction of UA 1, and 5	[39]
58	Number of DNS request per flow for a host	✓	63	Ratio of UAs	[39]
		II. Destination profile features			
64	Number of hosts connected to destination	[39]	72	Number of distinct URLs associated to a destination	✓
65–67	Destination received/sent Max/ Min/Avg	✓	73–75	Destination Max/Min/Mean packets failure	✓
68–70	Destination data packet exchange idle time Max/Min/Mean	✓	76–81	Max/Min/Mean number and ratio of DNS request per flow for a destination	✓
71	Number of resumed connections per flow for a destination	✓			
		II. URL profile features			
82	Fraction of URLs filename	[38]	94–96	URLs values Max/Min/Mean	[7,38,40]
83	Fraction of URLs filename exe	✓	97–99	URLs fragments Max/Min/Mean	[38]
84	Number of distinct extensions	[38]	100	Fraction of query	[38]
85–87	URLs length Max/Min/Mean	[7,30,38,40]	101	Number of strings	✓
88–90	URLs depth Max/Min/Mean	[7,38,40]	102	Number of URLs and distinct ones	[32]
91–93	URLs parameters Max/Min/Mean	[7,30,38,40]			

3.5 ContextualSummary

When all features are extracted for a received PairFlow and profile-based features are prepared, the ContextualSummary module collects these features in one bundle to be dispatched to the classifier. When a new PairFlow is received, EarlyCrow checks the ContextualSummary repository to identify if the pair had been already processed in the past. If so, the PairFlow will be processed as described in the previous sections. Then, it will be dispatched to the updating process module to combine the new flow with the previous ones as described in the next section. The purpose is to track the same connection over time to catch malicious behavior. For example, if a malicious actor bypasses EarlyCrow for the first flow, it will be tracked over time until it gets blocked. Indicators associated to positive detections may stay in the ContextualSummary repository and the blacklists for training the classifier. In Table 1, we summarize all features included in ContextualSummary.

3.6 ContextualSummary Updating Process

While PairFlows are stored in a repository, the ContextualSummary gets updated over time, using different rules for Host, Destination, and URL Profiles. If an incoming PairFlow has no associated ContextualSummary (Fig. 2, ❻), a new one is created. Otherwise the new PairFlow is considered for feature extraction, causing an update of the corresponding features of the associated ContextualSummary (Fig. 2, ❼). The time window is expanded with the new PairFlow to describe the overall time window covered by the ContextualSummary. However, updating profile-based features could cause higher time complexity because these profiles are to be updated for every different ContextualSummary. Therefore, new profile-based features are recalculated every $\hat{t}$ time, such that $\hat{t} > t$, where t is the selected granularity for EarlyCrow. For instance, we can configure $\hat{t}$ at 15 min in our experimental settings, which is higher than t by 50% if t at 10 min.

EarlyCrow considers different methods to update features according to their data type. Numerical features are updated by using a weighted average. As shown in Fig. 2, each ContextualSummary stores the last PairFlow ID as a counter of previous ones to be used for the weighted average formula. EPFLAG-based features, Boolean data types, are updated with OR operation with an incoming one to summarize the overall protocol used during ContextualSummary. For instance, APTs often have the DNS packets at the first PairFlow, but not the subsequent one, as we discussed in Sect. 2. Therefore, updating the ContextualSummary does not reset EPFLAG for the DNS. For Host-Profile features, strings of UA are stored to accurately extract other related UA, such as the number of distinct UAs which cannot be updated without having access to their strings.

4 Evaluation

This section evaluates EarlyCrow in a standard setting to investigate how our system performs against APTs and botnets. We evaluate EarlyCrow performance on the three datasets described below. The same experiments are performed on a baseline, inspired by Made [38], which is a NIDS detecting C&C used by botnets, ransomware, and APTs. Since we assume EarlyCrow to run in parallel with a malicious-domain detector, we omit domain-related features, which would also not be relevant for the considerable portion of traffic conforming to the Case II pattern of Sect. 2.

4.1 Datasets

APT malware attacks a few targets in discontinued time-frames spanning months or years, unlike other malware and common attacks. Therefore, the chance of

Table 2. Dataset characteristics used for measurements (training set) and for unseen malware evaluation (testing set).

Label	Set	Malware families
Malicious (567,090 packets)	Training	Bitsadmin (0.09%), Carbank (0.05%), Conficker (27.56%), Mivast&Sakula (0.93%), NanoCore (0.13%), njRAT (28.45%), PlugX (0.11%), Remcos (0.87%), Sogou (3.65%), Virut (9.59%), Zebrocy (0.98%),
	Testing (unseen)	Ammyy (1.01%), ChChes (0.13%), CobaltStrike (0.39%), Dridex (0.23%), Emotet (0.02%), Empire (1.70%), FlawedAmmy (0.24%), ImminentMonitor (11.27%), MagicHound (0.40%), OnionDuke (0.14%), PoisonIvy (0.25%), Ramnit (0.21%), StrongPity (11.38%), Zeus (0.04%)
Legitimate (766,641 packets)		Training: 70%, Testing: 30%

finding a real network infected with various APT campaigns is unrealistic. We resort to raw PCAP captures from two different honeypot networks, each of which includes legitimate, APTs and botnets C&C connections (Table 2). These APTs are often temporarily inactive. Due to this, we run them during multiple time windows (April 2020–January 2021, October–November 2019) until each campaign's activities are resumed, and their command and control are activated.

APTraces. We run different active malware using Any.Run[2] sandbox machines to generate PCAP files. These malware families are known to be used by 48 APT campaigns, and they were active and tied to an APT campaign at the time of the capture. These include RATs (*njRAT, Imminent Monitor, Cross-RAT, Mivast & Sakula, NanoCore, PlugX, PoisonIvy*) and trojans (*Empire, OnionDuke, MiniDuke, Remcos, StrongPity, Zebrocy*). We also consider legitimate connections from the same sandbox to avoid data bias based on the victim machine, configuration settings, or temporal bias [5] against legitimate.

Malware Capture Facility Project (MCFP). The MCFP[3] includes malware used in APTs such as (*Magic Hound and Cobalt*), admin tools (*Ammyy*), and RATs (*njRAT*). We also add botnets captures that use HTTP(S) for C&C communication (*Conficker, Dridex, Emotet, Ramnit, Sogou, Virut, Zeus*) and normal traffic (CTU-Normal-12, 20–22). After PairFlow compiles the PCAPs and generates HTTP variant files, we build three combined datasets: APTs vs. Legitimate, botnets vs. Legitimate, and Malicious (APTs or botnets) vs. Legitimate.

4.2 Classification Performance

Classifiers are evaluated in two modes. First, HTTP-Mode, which assumes the administrator connects the NIDS to a web proxy to decrypt HTTPS and accesses features such as UA, HTTP response codes, content type, and URL. Second,

[2] https://any.run.

[3] https://www.stratosphereips.org/datasets-overview.

Table 3. Classification performance.

Classifier name	Known malware						Unseen malware					
	FPR	Prec.	Recall	Acc.	$F1$	$mF1$	FPR	Prec.	Recall	Acc.	$F1$	$mF1$
I. Dataset: APTs vs. Legitimate												
EarlyCrow	0.40	94.20	93.69	99.17	99.17	93.89	0.74	94.48	91.67	98.11	98.08	93.02
Baseline	0.49	92.4	89.09	98.75	98.73	90.45	0.00	98.04	75.00	96.22	95.63	82.33
EarlyCrow-HTTPS	0.51	92.85	93.18	99.03	99.03	92.79	0.74	94.68	92.81	98.28	98.26	93.72
Baseline-HTTPS	0.72	82.90	68.96	97.19	96.72	73.18	0.00	96.70	56.82	93.47	91.10	60.29
II. Dataset: Botnets vs. Legitimate												
EarlyCrow	0.48	96.49	95.40	98.92	98.91	95.90	0.19	96.77	92.01	99.26	99.24	94.25
Baseline	0.57	94.64	86.92	97.61	97.49	90.24	0.19	95.08	78.85	98.35	98.16	85.06
EarlyCrow-HTTPS	0.42	96.79	95.02	98.92	98.90	95.84	0.19	95.49	81.48	98.53	98.40	87.12
Baseline-HTTPS	0.96	90.73	80.76	96.39	96.14	84.79	0.00	48.25	50.00	96.51	94.79	49.11
III. Dataset: Malicious vs. Legitimate												
EarlyCrow	0.86	95.41	94.79	98.29	98.29	95.06	0.93	94.77	91.60	97.51	97.46	93.11
Baseline	0.93	93.76	88.20	96.97	96.86	90.68	0.19	95.89	76.10	94.85	94.15	82.62
EarlyCrow-HTTPS	0.93	95.07	94.76	98.23	98.23	94.89	0.93	94.27	89.22	97.01	96.92	91.54
Baseline-HTTPS	0.95	91.21	78.66	95.13	94.689	83.47	0.00	95.22	54.76	90.53	86.86	56.18

HTTPS-Mode, where the administrator places the NIDS at the network edge without deciphering HTTPS. Because of imbalanced classes of APT (3.9%) and botnet (8.3%) compared to legitimate, we focus on macro average F1-score ($mF1$) in Table 3.

Known Malware. We randomly split the training and testing sets ten times. Then, we take the average performance under two constraints. First, the malware should be presented in both sets. Second, the infected hosts and the destination C&C server should be unique and not leaked from training to testing. EarlyCrow obtains the best performance with $mF1$ of 93.89%, 95.9%, and 95.06% for the three datasets. Even in HTTPS mode, which cannot take advantage of plaintext HTTP features such as headers or URL details, EarlyCrow still outperforms the baseline on both three tasks, scoring 92.79%, 95.84%, and 94.89% respectively. Note that EarlyCrow can operate almost similar on both modes on known malware. However, we will investigate that on unseen malware.

Unseen Malware. We train our classifiers on the training set that used for our measurement study. Then we evaluate the performance against unseen malware described in Table 2. EarlyCrow obtains the best performance with $mF1$ of 93.02%, 94.25%, and 93.11% for the three datasets. On all three tasks in HTTPS mode, EarlyCrow surpasses the baseline, achieving 93.72%, 87.12%, and 91.54%. For EarlyCrow, the performance loss between known and unseen is marginally low (1.96% of $mF1$) in the third dataset, while the baseline suffers a loss of 8.04%.

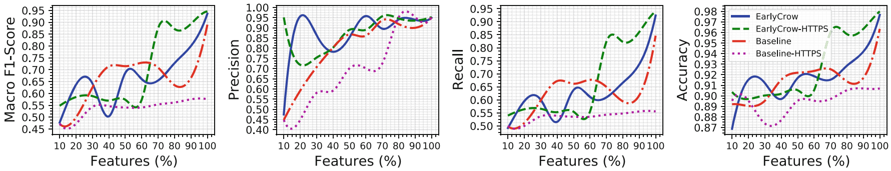


Fig. 3. Effect of using only the top % of features.

4.3 Discussion

We limit our discussion to the results of *unseen malware* on the third dataset presented in Table 2, which evaluates the generalization of EarlyCrow to mimic the real-world environment.

Features Diversity. The detection of APTs necessitates a spread of features, as presented in Sect. 2. In Fig. 3, we show the extent to which additional features affect the performance of the various classifiers. The first 10% of features for EarlyCrow show rapid improvements in terms of precision but with poor recall. Also, stronger features between 48% and 62% can improve the performance of *mF*1 up to 92.26% for EarlyCrow-HTTPS. Adding more features afterward increases the detection rate, enabling more unseen APTs to be detected. Furthermore, a system with diverse and strong features will require more time and resources to defeat as opposed to one that relies on a few particular features [47].

Top Features. We investigate the feature importance of the third dataset in HTTPS mode, which comprises APTs, botnets, and legitimate samples, because it is the one closest to a realistic scenario for APT hunting. Figure 4 illustrates the top features based on their information gain. MTDSC is an effective feature that reveals 82% of hosts infected with APTs and botnets spend up to 73.7 and 38.5 s, which are higher than 1.1 s of typical benign hosts, confirming the expected HTTP browsing. The longer time for APTs indicates using a *fallback channel*, which is generally established after a long time. Next, the average number of DNS requests for a host per connection is lower in APTs than botnets and legitimate, with 90% at most 2, 6, and 19, respectively. Interestingly, hosts infected with APTs have higher connections reaching further destinations by IP without *domain resolution* as *fallback channels*. This is consistent with our measurement study in Sect. 2.2. 70% of APTs use such an approach, with 88% or less for their connections, compared to only 1% for the legitimate.

While 60% of legitimate connections are repeated six times resumed or less, botnets are rarely disconnected, and APTs are weakly imitating legitimate *web protocol TTPs*, with two-thirds lower. Next, APTs and botnets are considerably slower than legitimate, with mean delta inter-arrival times at most 33.5×10^{-2}, 46×10^{-2}, and 0.5×10^{-2} seconds at 95% of their probability. We confirm that APTs tend to switch from HTTP to raw TCP for malicious operations representing *non-application protocols*. Within a PairFlow, we find that 50%

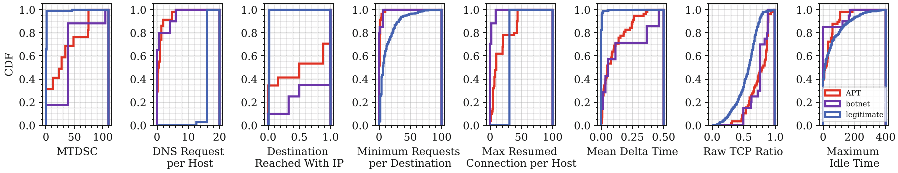


Fig. 4. Cumulative distribution of top features gains on the testing set for EarlyCrow-HTTPS.

of APTs rely on 81.09% (58.35% for legitimate) of the whole exchange packets on raw TCP, indicating the adversary use HTTP as camouflage while still relying on TCP for many tasks. Nonetheless, the APTs and botnets are faster regarding the difference between data packets. They tend to be shorter/faster than legitimate, where 90% of them take 104, 124, and 168 s, respectively.

Evasion Attacks. In Table 4, we break down the results of EarlyCrow-HTTPS on unseen malware. 92% of unseen malware are detected with at least one C&C communication, and 64% of different malware are fully detected. However, one server belonging to StrongPity is not detected in HTTPS. We found StrongPity is not using a fallback channel, and its measurement reflects a legitimate one. In HTTP mode, EarlyCrow managed to detect StrongPity because of its malicious URL characteristics, such as using `.exe` file extension, and lacks a rich web server (i.e., No. of URLs distinct) compared to the proportional data volume. Also, some C&C servers belonging to OnionDuke and Zeus managed to evade detection. These servers are established as fallback channels with minimum data transfers, which evade many features. Since the malware is detected on a specific machine, we recommend a SOC analysis to sanitize the victim machine from the malware to stop other possible C&C communications.

Table 4. Detection rate on unseen malware over HTTPS.

Malware	C&C servers	Detection (%)	Malware	C&C servers	Detection (%)
Ammyy	8	100	ImminentMonitor	4	75
ChChes	1	100	Magic-Hound	3	100
CobaltStrike	2	100	OnionDuke	6	33.34
Dridex	2	100	PoisonIvy	1	100
Emotet	13	53.84	Ramnit	2	100
Empire	5	100	StrongPity	1	0
FlawedAmmy	4	100	Zeus	3	33.34

4.4 Limitations

APT Campaigns. EarlyCrow is geared toward detecting the early stages of infection. Nevertheless, it is difficult to conclude which suspicious activities are due to advanced adversaries and which are due to mainstream malware variants. Hence, we recommend tracking APT campaigns and malware activities over a longer period of time on a live system. This could be done by deploying EarlyCrow on the network of several likely targets, such as a sovereign entity or large financial institution, over months and periodically reevaluating EarlyCrow reports against each APT campaign to drive further improvement. By maintaining our repository publicly accessible [1], we encourage collaboration with the open source and research communities to run EarlyCrow on their targeted networks and to share their findings for further improvements.

Adversarial Robustness. Previous work has studied adversarial attacks against deep learning based NIDS [12,26,27,36], and discussed robustness for traditional machine learning such as random forest [9,37,38]. Although EarlyCrow is motivated by the techniques used by APTs to evade NIDS, *Practical* and *Feature-space Attacks* [11,25] specifically targeting PairFlow fields and EarlyCrow features may still be possible. For instance, Random Interval-Time (RIT) [8,44] and Random Duplication (RD) [29] were used against botnets, and could be tested against APTs. The former generates adversarial samples by altering packets' arrival times, which APTs could easily do, although it may have less effect on lower volume traffic. The latter duplicates the number of packets randomly, and may be less useful to APTs, but still useful as a measure of robustness.

Practical and Feature-space Attacks [25] could also be considered. In a Practical Black-box Attack (PBA) the adversary knows what traffic features are selected by the classifiers, including in our case, the PairFlow fields from Table 5. Moreover, adversaries can access most features published in the past to adapt their traffic according to the targeted feature extraction to evade NIDS. We refer such assumption to Practical Gray-Box Attack (PGA) for those features used in the literature presented in Table 1. Another two attack configurations can be considered [11,25,48], including Feature-space Grey-box Attack (FGA), Feature-space Black-box Attack (FBA). FGA may attack all features produced by EarlyCrow, while the FBA is produced by the state of art baseline, i.e., reproducible Made and non-novel features in Table 1. However, there are several variations in finding the optimization of evasion attacks. We suggest to adopt variants of Euclidean norm [10,45] (l_p) for black-box configuration and free-range [49] for Gray-box.

Execution Time. EarlyCrow can speed up its execution by running its modules (of Fig. 2, ❸) in parallel using Hadoop, similar to [30], optimizing memory hierarchy, and pipelining the main processes (Fig. 2, ❶–❸). The current implementation aims to prove the concept and focuses on detection performance. Further investigations of how to improve and measure execution speed and memory footprint in a production environment are left for future work.

5 Related Work

There is very limited previous work on detecting APTs at the network level. Detecting C&C in general is the closest area. In our approach we test several features from the literature which can be relevant for APTs including URLs and UA features [32,38–40], traffic exchange bytes [7,8,38,46], HTTP content types [38,39], and GET and POST ratio [38,40]. Besides directly using such features, EarlyCrow pivots them into host, destination and URL profiles, and combines them in contextual summaries.

Some previous works focuses on detecting APTs in addition to other kinds of malicious communications [38,39]. Oprea et al. [39] propose a belief propagation (BP) algorithm to detect early-stage infection of APTs. They model enterprise communication using a bipartite graph with two vertices, hosts, and domains based on simulated attacks. Once the detector identifies a malicious remote host or domain based on several features, BP identifies communities of malicious domains with similar features that are part of the same attack campaign. Domain scores are calculated as a supervised linear regression weighted sum of features. As discussed in Sect. 4, APTs tend to infect a lower number of hosts than botnets. Therefore, EarlyCrow consider other features based on different TTPs discussed in Sect. 2.

EarlyCrow is closer to Made [38], which instead uses web proxy logs at the edge of an enterprise network to detect malicious C&C communications, including APTs. Made leverages features related to the communication, HTTP request, response and its content, URL, and UAs. These are used by a random forest classifier to assign a risk score for each connection. As discussed in Sect. 4, Made is not as effective on HTTPS traffic, which is nowadays harder to intercept and decrypt due to technical and legal requirements. In addition, EarlyCrow considers five other TTPs besides the *Web Application Protocol* TPP at the heart of Made.

ExeceScent [37] detects C&C domains by clustering incoming requests into five templates, including median URL path, URL query component, User-Agent, other headers, and destination network. These templates are used to estimate similarity scores to predefined Control Protocol Templates (CPT) centroids. However, this is open to evasion if an adversary copies the UA of the victim machine from the Windows Registry [9]. In addition, it is not possible to extract most HTTP header features when HTTPS is in use, which hinders the generalization process and may result in mixing APTs with legitimate ones in many clusters. A related approach [7] adopts similar features, only using histogram bins which also can be evaded using HTTPS.

BAYWATCH [30] is a filtering system to detect the beaconing of infected hosts. Universal and local whitelist are filtered, then beaconing can be detected using Fourier transform and Gaussian mixture model, awarding a high agglomerative hierarchical clustering score for strong periodicity. BAYWATCH filters URLs and domains that are likely to be legitimate. Unprocessed connections with all previous features are sent to a random forest for classification. BAYWATCH can be computationally expensive for only beaconing behavior, and many APTs also have non-beaconing connections. EarlyCrow detect malicious connections regardless of their pattern. Finally, Kitsune [36] adopts an ensemble of autoencoders, proving the efficiency of unsupervised deep learning to detect classic attacks such as ARP poisoning and SYN DoS, which are rarely used by APTs. As discussed in Sect. 1, we avoid using deep learning because of the scarce dataset representing various APTs TTPs, which is essential for deep learning models.

6 Conclusions

We presented a threat model for APTs which focuses on the TTPs used by adversaries to avoid existing NIDS. As part of our measurement study, we demonstrated the significant overlap between APT and legitimate behaviors, and clarified their characteristics. Taking this into account, we designed and implemented EarlyCrow, a tool which can detect APT malware network activities that are missed by current deployed defense mechanisms. Our results demonstrate the importance of using diverse features based on contextual fields to detect unseen APT malware. We recommend using EarlyCrow as an additional layer of defense, besides SIEM, Host Intrusion detectors (HIDS), and domain detectors. While EarlyCrow is motivated by the NIDS-avoiding behavior of APTs, adversarial attacks specifically targeting PairFlow fields and EarlyCrow features may still be possible. A study of adversarial defenses and their robustness, and deployment issues is left to future work.

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

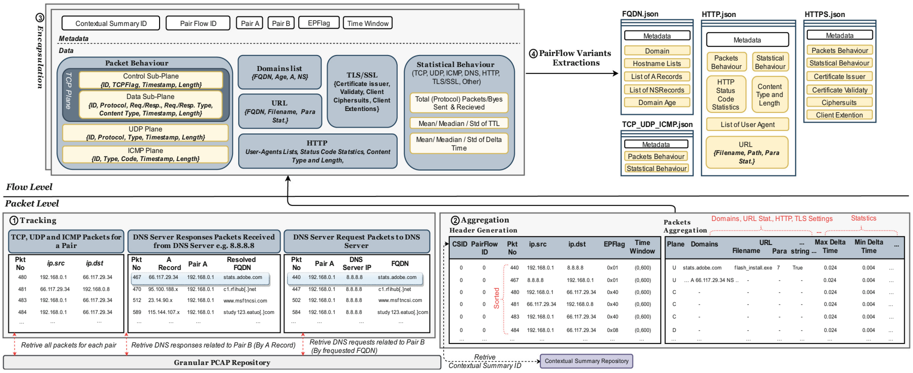
Packets Retrieving. The tracking module identifies all unique pair connections on the network and filters out those using non-IP protocols (Fig. 5, ❶). For each unique pair connection, PairFlow tracks, bidirectionally, all packets related to a pair. These packets are designated with an initial Flow ID. 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 [4], 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 (Fig. 5, ❷). If so, the incoming PFID will be the last used PFID for the same pair and ContextualSummary ID, incremented by one. Otherwise, a new and unique ContextualSummary will be created, and the PFID will start with zero.

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 [6], 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.


③ Encapsulation
Contextual Summary ID
Pair Flow ID
Pair A
Pair B
EPFlag
Time Window
Metadata
Data
Packet Behaviour
TCP Plane
Control Sub-Plane
{ID, TCPFlag, Timestamp, Length}
Data Sub-Plane
{ID, Protocol, Req./Resp., Req./Resp. Type, Content Type, Timestamp, Length}
UDP Plane
{ID, Protocol, Type, Timestamp, Length}
ICMP Plane
{ID, Type, Code, Timestamp, Length}
Domains list
{FQDN, Age, A, NS}
URL
{FQDN, Filename, Para Stat.}
TLS/SSL
{Certificate issuer, Validaty, Client Ciphersuits, Client Extentions}
HTTP
User-Agents Lists, Status Code Statstics, Content Type and Length,
Statistical Behaviour
(TCP, UDP, ICMP, DNS, HTTP, TLS/SSL, Other)
Total (Protocol) Packets/Byes Sent & Recieved
Mean/ Meadian / Std of TTL
Mean/ Meadian / Std of Delta Time
④ PairFlow Variants Extractions
FQDN.json
Metadata
Domain
Hostname Lists
List of A Records
List of NSRecords
Domain Age
TCP_UDP_ICMP.json
Metadata
Packets Behaviour
Statstical Behaviour
HTTP.json
Metadata
Packets Behaviour
Statstical Behaviour
HTTP Status Code Statistics
Content Type and Length
List of User Agent
URL
{Filename, Path, Para Stat.}
HTTPS.json
Metadata
Packets Behaviour
Statstical Behaviour
Certificate Issuer
Certificate Validaty
Ciphersuits
Client Extention
Flow Level
Packet Level
① Tracking
TCP, UDP and ICMP Packets for a Pair
DNS Server Responses Packets Received from DNS Server e.g. 8.8.8.8
DNS Server Request Packets to DNS Server
Retrive all packets for each pair
Retrive DNS responses related to Pair B (By A Record)
Retrive DNS requests related to Pair B (By frequested FQDN)
Granular PCAP Repository
② Aggregation
Header Generation
Sorted
Packets Aggregation
Domains, URL Stat., HTTP, TLS Settings
...
Statstics
Retrive
Contextual Summary ID
Contextual Summary Repository

Pkt No	ip.src	ip.dst
480	192.168.0.1	66.117.29.34
481	66.117.29.34	192.168.0.8
483	192.168.0.1	66.117.29.34
484	192.168.0.1	66.117.29.34
...	...	...

Pkt No	A Record	Pair A	Resolved FQDN
467	66.117.29.34	192.168.0.1	stats.adobe.com
470	95.100.188.x	192.168.0.1	c1.rfihub[.]net
512	23.14.90.x	192.168.0.1	www.msftncsi.com
589	115.144.107.x	192.168.0.1	study123.eatuo[.]com
...	...	...	...

Pkt No	Pair A	DNS Server IP	FQDN
440	192.168.0.1	8.8.8.8	stats.adobe.com
447	192.168.0.1	8.8.8.8	c1.rfihub[.]net
502	192.168.0.1	8.8.8.8	www.msftncsi.com
584	192.168.0.1	8.8.8.8	study123.eatuo[.]com
...	...	...	...

CSID	PairFlow ID	Pkt No	ip.src	ip.dst	EPFlag	Time Window
0	0	440	192.168.0.1	8.8.8.8	0x01	(0,600)
0	0	467	8.8.8.8	192.168.0.1	0x01	(0,600)
0	0	480	192.168.0.1	66.117.29.34	0x40	(0,600)
0	0	481	66.117.29.34	192.168.0.8	0x40	(0,600)
0	0	483	192.168.0.1	66.117.29.34	0x40	(0,600)
0	0	484	192.168.0.1	66.117.29.34	0x08	(0,600)
...	...	...	...	...	...	...

Plane	Domains	URL		...	Max Delta	Min Delta	...
		Filename	Para	string ...	Time	Time	
U	stats.adobe.com	flash_install.exe	7	True	0.024	0.004	...
U	... A 66.117.29.34 NS ..	-	-	-	0.024	0.004	...
C	-	-	-	-	0.024	0.004	...
C	-	-	-	-	0.024	0.004	...
C	-	-	-	-	0.024	0.004	...
D	-	-	-	-	0.024	0.004	...
...	...	...	...	...	...	...	...

Fig. 5. Overview of the PairFlow workflow.

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 (Fig. 5, ❸). The data types involved include list of strings and tuples, Boolean and numeric fields, as shown in Table 5.

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 Fig. 5. Each packet in the data sub-plane holds protocol name, request/response and their types, 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 Sect. 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 s, 74), (72126, '0x12', 215.78 s, 70 B), (72127, '0x10', 215.78 s, 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 Sect. 3.4.

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 s, 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 Fig. 5, *domain list* encapsulates all FQDNs related information 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 strings, 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 UAs. *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 5.

Table 5. Summary of PairFlow data fields (B: Boolean, LS: List of Strings, LT: List of Tuples, N: Numerical).

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	Client and server ciphersuites	LS
6–12	EPFLAG raw TCP, raw UDP, ICMP, DNS, HTTP, TLS and SSL	B	22–23	Client and server extension types	LS
13	FQDN	LS	24–25	Client and server signature algorithms and hashes	LS
14	Resource records: type nameserver	LS	26–27	Client and server supported groups	LS
15	Resource records: type A	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	Client and server ciphersuites bytes Max/Min/Median	N
36–42	Number of raw TCP, raw UDP, ICMP, DNS, HTTP, TLS and SSL packets	N	60–62	Client and server extensions bytes Max/Min/Median	N
43	Duration	N			

A.4 Variants Extraction

PairFlow processing also exports four *variant* JSON files which can be used by any external classifier (Fig. 5, ❹). 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 [3]. TCP-UDP-ICMP.json is dedicated for those classifiers use time-series for detection [6,30]. 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 [30,38]. Other classifiers may deploy HTTPS.json for detecting encrypted communications without deciphering the traffic [4]. A detailed study of the other variants is left for future work.

References

1. EarlyCrow github repository. https://github.com/ICL-ml4csec/EarlyCrowAPT
2. Ahmad, A., Webb, J., Desouza, K.C., Boorman, J.: Strategically-motivated advanced persistent threat: definition, process, tactics and a disinformation model of counterattack. Comput. Secur. **86**, 402–418 (2019)
3. Alageel, A., Maffeis, S.: Hawk-Eye: holistic detection of APT command and control domains. In: ACM SAC, pp. 1664–1673. ACM (2021)
4. Anderson, B., McGrew, D.: Identifying encrypted malware traffic with contextual flow data. In: ACM AISec, pp. 35–46 (2016)
5. Arp, D., et al.: Dos and don'ts of machine learning in computer security. In: USENIX Security (2022)
6. 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)
7. Bartos, K., Sofka, M., Franc, V.: Optimized invariant representation of network traffic for detecting unseen malware variants. In: USENIX Security, pp. 807–822 (2016)
8. 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)
9. Bortolameotti, R., et al.: DECANTeR: DEteCtion of anomalous outbouNd HTTP TRaffic by passive application fingerprinting. In: ACSAC, pp. 373–386 (2017)
10. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: IEEE S&P, pp. 39–57. IEEE (2017)
11. Clements, J., Yang, Y., Sharma, A., Hu, H., Lao, Y.: Rallying adversarial techniques against deep learning for network security. arXiv preprint arXiv:1903.11688 (2019)
12. Clements, J., Yang, Y., Sharma, A.A., Hu, H., Lao, Y.: Rallying adversarial techniques against deep learning for network security. In: IEEE SSCI, pp. 01–08. IEEE (2021)
13. The MITRE Corporation: Application layer protocol: web protocols. https://attack.mitre.org/techniques/T1071/001/. Accessed 18 Dec 2021
14. The MITRE Corporation: Command and control. https://attack.mitre.org/tactics/TA0011/. Accessed 18 Dec 2021
15. The MITRE Corporation: Data obfuscation: protocol impersonation. https://attack.mitre.org/techniques/T1001/003/. Accessed 18 Dec 2021
16. The MITRE Corporation: Dynamic DNS. https://attack.mitre.org/techniques/T1568/. Accessed 18 Dec 2021
17. The MITRE Corporation: Dynamic resolution: fast flux DNS. https://attack.mitre.org/techniques/T1568/001/. Accessed 18 Dec 2021
18. The MITRE Corporation: Encrypted channel. https://attack.mitre.org/techniques/T1573/. Accessed 18 Dec 2021
19. The MITRE Corporation: Fallback channel TTP. https://attack.mitre.org/techniques/T1008/. Accessed 18 Dec 2021
20. The MITRE Corporation: Non-application layer protocol. https://attack.mitre.org/techniques/T1095/. Accessed 18 Dec 2021
21. The MITRE Corporation: Protocol tunneling. https://attack.mitre.org/techniques/T1572/. Accessed 18 Dec 2021
22. Farinholt, B., Rezaeirad, M., McCoy, D., Levchenko, K.: Dark matter: uncovering the DarkComet RAT ecosystem. In: WWW, pp. 2109–2120 (2020)

23. Farinholt, B., et al.: To catch a ratter: monitoring the behavior of amateur DarkComet RAT operators in the wild. In: IEEE S&P, pp. 770–787. IEEE (2017)
24. FireEye: Highly evasive attacker leverages solarwinds supply chain to compromise multiple global victims with sunburst backdoor, 13 December 2020. https://www.mandiant.com/resources/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor
25. Han, D., et al.: Evaluating and improving adversarial robustness of machine learning-based network intrusion detectors. IEEE J. Sel. Areas Commun. **39**(8), 2632–2647 (2021)
26. Hashemi, M.J., Cusack, G., Keller, E.: Towards evaluation of NIDSs in adversarial setting. In: ACM Big-DAMA, pp. 14–21 (2019)
27. Hashemi, M.J., Keller, E.: Enhancing robustness against adversarial examples in network intrusion detection systems. In: IEEE NFV-SDN, pp. 37–43. IEEE (2020)
28. Heinemeyer, M.: Fin7.5: the infamous cybercrime rig "FIN7" continues its activities. https://securelist.com/fin7-5-the-infamous-cybercrime-rig-fin7-continues-its-activities/90703//. Accessed 18 July 2021
29. Homoliak, I., Teknøs, M., Ochoa, M., Breitenbacher, D., Hosseini, S., Hanacek, P.: Improving network intrusion detection classifiers by non-payload-based exploit-independent obfuscations: an adversarial approach. EAI Endorsed Trans. Secur. Saf. **5**, 17 (2018)
30. Hu, X., et al.: BAYWATCH: robust beaconing detection to identify infected hosts in large-scale enterprise networks. In: IEEE/IFIP DSN, pp. 479–490. IEEE (2016)
31. Hutchins, E.M., Cloppert, M.J., Amin, R.M.: Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. In: Leading Issues in Information Warfare & Security Research, vol. 1, p. 80 (2011)
32. Invernizzi, L., et al.: Nazca: detecting malware distribution in large-scale networks. In: NDSS, vol. 14, pp. 23–26 (2014)
33. Jansen, W.: Abusing cloud services to fly under the radar. https://research.nccgroup.com/2021/01/12/abusing-cloud-services-to-fly-under-the-radar/. Accessed 18 Dec 2021
34. Ma, J., Saul, L.K., Savage, S., Voelker, G.M.: Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In: KDD, pp. 1245–1254. ACM (2009)
35. Milajerdi, S.M., Gjomemo, R., Eshete, B., Sekar, R., Venkatakrishnan, V.: HOLMES: real-time APT detection through correlation of suspicious information flows. In: IEEE S&P, pp. 1137–1152. IEEE (2019)
36. Mirsky, Y., Doitshman, T., Elovici, Y., Shabtai, A.: Kitsune: an ensemble of autoencoders for online network intrusion detection. In: NDSS (2018)
37. Nelms, T., Perdisci, R., Ahamad, M.: ExecScent: mining for new C&C domains in live networks with adaptive control protocol templates. In: USENIX Security, pp. 589–604 (2013)
38. Oprea, A., Li, Z., Norris, R., Bowers, K.: MADE: security analytics for enterprise threat detection. In: ACSAC, pp. 124–136 (2018)
39. 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)
40. 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)
41. Rezaeirad, M., Farinholt, B., Dharmdasani, H., Pearce, P., Levchenko, K., McCoy, D.: Schrödinger's RAT: profiling the stakeholders in the remote access trojan ecosystem. In: USENIX Security, pp. 1043–1060 (2018)

42. Schindler, T.: Anomaly detection in log data using graph databases and machine learning to defend advanced persistent threats. In: GI-Jahrestagung (2017)
43. Sommer, R., Paxson, V.: Outside the closed world: on using machine learning for network intrusion detection. In: IEEE S&P, pp. 305–316. IEEE (2010)
44. Stinson, E., Mitchell, J.C.: Towards systematic evaluation of the evadability of bot/botnet detection methods. In: WOOT, vol. 8, pp. 1–9 (2008)
45. Szegedy, C., et al.: Intriguing properties of neural networks. CoRR abs/1312.6199 (2014)
46. Tegeler, F., Fu, X., Vigna, G., Kruegel, C.: BotFinder: finding bots in network traffic without deep packet inspection. In: CoNEXT, pp. 349–360 (2012)
47. Wang, J., Qixu, L., Di, W., Dong, Y., Cui, X.: Crafting adversarial example to bypass flow-&ML-based botnet detector via RL. In: RAID, pp. 193–204 (2021)
48. Wang, Z.: Deep learning-based intrusion detection with adversaries. IEEE Access **6**, 38367–38384 (2018)
49. Zhou, Y., Kantarcioglu, M., Thuraisingham, B., Xi, B.: Adversarial support vector machine learning. In: KDD, pp. 1059–1067. ACM (2012)