Event-Flow Correlation for Anomaly Detection in HTTP/3 Web Traffic

Stanislav Špaček
Institute of Computer Science and Faculty of Informatics
Masaryk University
Brno, Czech Republic
spaceks@ics.muni.cz

Petr Velan
Institute of Computer Science
Masaryk University
Brno, Czech Republic
velan@ics.muni.cz

Martin Holkovič
Progress Software
Brno, Czech Republic
martin.holkovic@progress.com

Tomáš Plesník
Institute of Computer Science
Masaryk University
Brno, Czech Republic
plesnik@ics.muni.cz

***Abstract*—The new HTTP/3 protocol for web traffic was recently released, superseding the widely used HTTP/2. It now supports exclusively encrypted transmissions and brings a lot of other changes. Many of these changes promote user privacy but hinder security monitoring of network traffic. In the past, the direct correlation of events and flows generated by the HTTP/2 web traffic enriched the encrypted network flows with the content from the web server's event log. In this paper, we modify the event-flow correlation method for the HTTP/3 protocol. Then, we deploy and evaluate the correlation method on a real traffic dataset. We compare the results of the correlation with the results for HTTP/2 and discuss the differences in the correlation of the new protocol compared to the original one. The results show that event-flow correlation can still enrich HTTP/3 network flows, albeit it introduces new challenges to cope with.**

***Index Terms*—Network flow monitoring, host-based monitoring, event, flow, event-flow correlation, HTTP/3.**

I. Introduction

After six years of development, an RFC specifying a new version of the HTTP/3 web communication protocol was released in June 2022 [1]. This protocol brings several changes to increase the speed, efficiency, and security of web browsing. However, from the networking point of view, the most fundamental change is the transfer of functions such as session management, congestion control, and security. These functions are now provided by the QUIC protocol [2]. Consequently, the HTTP/3 is no longer based on the connection-oriented TCP protocol but uses the connection-less UDP. Such a significant change in web traffic poses a challenge to the security monitoring of network traffic as network monitoring methods need to adapt quickly to the new protocol.

Our research focuses on enriching the IP flow monitoring by correlating the flows to the events captured on the endpoints. For encrypted IP flows, it is possible to partially reconstruct their content with event-flow correlation and thus reduce the volume of data for encrypted traffic analysis, which is imprecise and resource-intensive. On the other hand, correlating flows to events helps to detect tampering with the logs, where adversaries delete or modify logs to hide their activity. In addition, this option is privacy-preserving for web-browsing users because only the web service provider has access to the logs.

In this paper, we describe the event-flow correlation of events captured on the web server to IP flows generated by real HTTP/3 traffic. The aim of our research is to answer the following question:

- *How accurately can we correlate events and IP flows of HTTP/3 web traffic?*

We used the event-flow correlation method from our previous research on HTTP/2 [3] to answer the research question. We deployed the method over a dataset of HTTP/3 web traffic generated by real users. Based on these preliminary results, we further adapted the correlation method to HTTP/3 to achieve the highest possible amount of correlated data.

Our results show that tampering with events on a web server may easily be detected, as it is possible to accurately find a related network flow to every event generated regularly. On the other hand, a large number of network flows still remained uncorrelated due to connections that never reached the web server application to generate an event. Future work will be needed to filter these flows out and refine the results.

To enable the result reproduction and verification of our research, we provide all data and code in a repository [4]. This includes the anonymized dataset, the code used to normalize and filter the data, and the correlation scripts.

II. Background and Related Work

When correlating the data obtained by network and host-based HTTP/3 monitoring, we work with their primary outputs: IP flows and events, respectively.

The IP flow is defined in RFC 5470 as a set of IP packets passing an observation point in the network during a specific time interval. The IPFIX protocol then defines the flow collecting and exporting process and specifies the number and format of the flow features [5]. In this research, we consider network flows to be bidirectional as defined in RFC 5103 [6]. Furthermore, we refer to the flow source as the client and the destination as the server of the HTTP communication.

A web server log event is not defined simply as the flow. Web server applications use different logging standards, often based on the World Wide Web Consortium's Common Logging Format (CLF) [7]. In our research, we use the Windows Server Internet Information Services, which uses

its proprietary standard [8]. However, the proposed approach is also valid for other web server applications if the data normalization process is adapted.

The number of publications related to HTTP/3 security monitoring remains small, as its final specification was completed very recently. Chatzoglou et al. evaluated HTTP/3 security against several known HTTP/2 attacks, concluding that some of them may be adapted to affect HTTP/3 web servers [9], [10]. Marx et al. describe the debugging of QUIC and HTTP/3, explicitly stating that the transition to UDP negatively affects network monitoring because some previously available TCP features are now encrypted [11]. On the other hand, publicly available network monitoring middleboxes can already monitor HTTP/3 traffic and export information available in HTTP/2, e.g., the Server Name Indication (SNI) [12].

The state-of-the-art algorithms for monitoring data correlation were described in surveys by Kotenko et al., and Albasheer et al. [13], [14]. Kotenko et al. focus on the correlation of alerts. However, the survey is still relevant for our research, as both an event and a flow may be abstracted as alerts from different sensors for a single occurrence. Albasheer et al. conclude that network monitoring systems produce many false positive detections, so further analysis or correlation is needed. Correlating the network monitoring data with host-based monitoring evaluated in this paper should lower the number of false positive detections in web traffic and thus improve intrusion detection in network traffic.

III. Methodology

In our research, we proceeded as follows. First, we analyzed samples of HTTP/3 traffic and determined the common feature vector for events and flows. As a correlation method, we chose the *no-sni* variant, which showed the best results in our previous research over the same common feature vector that we identified for HTTP/3 [3]. Current HTTP/3 traffic is initiated by HTTP/2 communication; therefore, we modified the *no-sni* method for a separate correlation of both versions of web traffic. Then we collected a dataset of HTTP/3 traffic from a web server in the campus network. Finally, we ran the correlation method on the dataset and evaluated the results based on the number of generated anomalies.

The data collection and processing are described in Figure 1. In the monitored environment, the clients communicate with the web server using the encrypted HTTP/3 protocol. The web server logs interactions with clients and thus serves as the source of the events. The traffic containing client requests and server responses are captured directly on the network interface of the web server. After capture, the event and flow data go through several processing stages. First, the network data is aggregated into flows from the source PCAP; the events require no such transformation. Then, both types of data go through the normalization and filtering process to transform all their features into a uniform format and filter out any errors and monitoring anomalies. Finally, the events and flows are correlated and divided between correlated data and anomalies based on the correlation results.

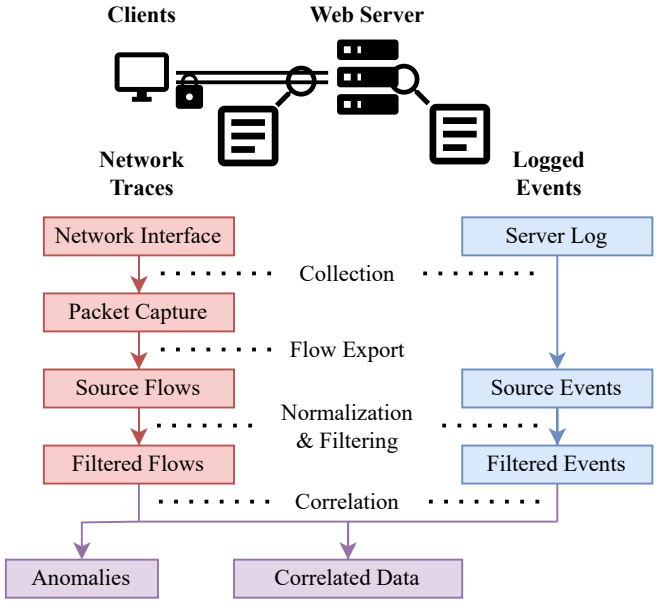

Fig. 1. The process of collection, preprocessing, and correlation of HTTP/3 events and flows.

IV. HTTP/3 Event-Flow Correlation

This section details the theoretical foundations of HTTP/3 event-flow correlation; it deals with the enumeration of common features, the definition of correlation methods, and the correctness analysis of the correlation results.

A. Common Features of HTTP/3 Events and Flows

Each event and IP flow of the HTTP/3 protocol is defined by specific values stored in a fixed feature vector. These features include, for example, the IP address and port of both the server and the client, the time stamp, and the volume of transferred data. In Table I, we show all the common features we identified for the versions of the HTTP protocol - unencrypted, encrypted with TLS 1.2, TLS 1.3, and the QUIC protocol. The HTTP/3 standard exists only in an encrypted form and uses the QUIC protocol, so the HTTP/3 corresponds to the last column of the table. Features we use for the HTTP3 correlation in this paper are highlighted in bold.

We did not use the SNI for correlation for two reasons. Firstly, because there is only a single site on the server in our dataset, and secondly, because in our previous research, it was shown that SNI has only a marginal impact on the accuracy of the correlation [15]. For the transferred data volume, the relations between these features in events and flows are not trivial and require more future work to be used in correlation.

B. HTTP/3 Event-Flow Correlation Method

We precisely determine the relationship between an event and a flow by using five common features: server/client IP address, server/client port, and an event timestamp (or an interval in case of a flow). Among these features, the client port is critical to unambiguously assigning a related event to a flow. The client port is pseudorandomly chosen, and there

TABLE I
THE FEATURES COMMON TO HTTP EVENTS AND FLOWS AND THEIR AVAILABILITY IN NETWORK TRAFFIC FOR DIFFERENT ENCRYPTION PROTOCOLS. FEATURES IN BOLD WERE USED FOR CORRELATION.

Feature		Description	Availability in HTTP			
Event	Flow		Plain	TLS 1.2	TLS 1.3	QUIC
time-generated	**[START_NSEC, END_NSEC]**	The time/interval of occurrence in milliseconds	✓	✓	✓	✓
s-ip	**L3_IPV4_DST**	The IP address of the logging web server	✓	✓	✓	✓
s-port	**L4_PORT_DST**	The server port number that is configured for the service	✓	✓	✓	✓
c-ip	**L3_IPV4_SRC**	The IP address of the client that made the request	✓	✓	✓	✓
c-port	**L4_PORT_SRC**	The port of the client that made the request	✓	✓	✓	✓
sc-bytes	BYTES_B	The number of bytes that the server sent	✓	✓	✓	✓
cs-bytes	BYTES_A	The number of bytes that the server received	✓	✓	✓	✓
cs-host	HTTP_REQUEST_HOST	The server name identifier (SNI)	✓	✓	?	?

is a possibility that it will be repeated after a sufficiently long time and with a sufficiently high number of flows between the server and the client. This may cause correlation anomalies, but a sufficiently small correlation time-window will reduce the probability to an acceptable level. The correlation method we designed for HTTP3 is based on the *no-sni* method in our previous research [3]. The algorithm of the method is described by the Algorithm 1.

Correlating events to flows exclusively if they occur between the start and end of their related flow performs poorly in the real environment due to latency, jitter, and time synchronization drift in the network. As we found out during this research, this also applies if the capture of the event and flow takes place on a single server, even though it is not affected by the inaccuracies of the network. It is, therefore, necessary to use a time window for correlation.

The time window we use is defined by earliness and lateness. The *earliness* (*lateness*) is the lower (upper) bound of the time-window, and it specifies the time interval by which an event may precede (follow) a flow to be still considered related. A too small or a too large time-window will produce correlation anomalies (ERR1 and ERR2, respectively, described in Section IV-C) – relations that do not correspond with the HTTP/3 communication protocol.

The optimal time-window for correlation must be measured separately for each unique environment because it depends on environment-specific quantities, e.g., latency, jitter, and time synchronization drift. The time-window determined in our environment is therefore only applicable here; however, the process for its determination can be reused. We describe the process in Section VI. It should also be noted that after any fundamental change in the environment, the time-window needs to be recalculated.

C. HTTP/3 Relation Corectness

The event-flow correlation forms relationships between events and flows, but we cannot automatically consider all relationships created this way to be correct. We focus on the cardinality of a relationship created by the event-flow correlation to determine its correctness. To define correct and anomalous relations and analyze the causes of such anomalies, we start from the following two assumptions:

1) *Each HTTPS flow triggered at least one event at the web server.*
2) *Each event captured on a web server was caused by exactly one HTTPS flow.*

The cardinality options for the relationship between events and flows are shown in Table II. The error *ERR1* indicates flows and events which break the first rule, as no counterpart for them was found in correlation. The error *ERR2* includes events that break the second rule as they have been related to multiple flows at once.

The *ERR1* correlation error can be caused during the data collection or correlation process. Events or flows may be missing from the dataset due to a monitoring outage or discarded during normalization due to having an incomplete correlation feature vector. Furthermore, the network traffic may include flows that classify as HTTPS but do not reach the web server application because of, e.g., a failed TLS handshake or client interruption. During the correlation process, an error of this type may be caused by a too strict time-window. Some relationships will not be established if the time-window is too small because the flow and event were captured too far apart. Flows and events that remain uncorrelated are further referred to as single events and single flows.

The *ERR2* error can be caused only during correlation and applies only to events; such a relation is considered correct for the flow. The error is caused by the inability to correctly assign an event to a flow when all their correlation features match. It occurs if such identical events and flows appear closer apart than the correlation time-window. For example, this can be caused by a web crawler repeatedly requesting the same resource from the server. We refer to such events associated with multiple flows as polygamous events.

V. DATASET

First, we created a web server on the university site and configured HTTP/3 advertisement support on it. Then we created a single website containing several HTML-based web pages filled with open source content – videos, images, and games. We make the website available in a separate repository [4]. We monitored log events and web traffic on the server for six days, from June 2nd, 2022, to June 8th, 2022. On the second day of monitoring, June 3rd, 2022, the promotion of websites

Algorithm 1: Correlation algorithm

if $flow.L3\_IPV4\_DST = event.s\_ip$ **and** $flow.L3\_IPV4\_SRC = event.c\_ip$ **and**
$flow.L4\_PORT\_DST = event.s\_port$ **and** $flow.L4\_PORT\_SRC = event.c\_port$ **and**
$flow.START\_NSEC - earliness \leq event.time\_generated$ **and**
$flow.END\_NSEC + lateness \geq event.time\_generated$ **then**
  match $flow$ with $event$
end if

TABLE II
CARDINALITY OF ALL POSSIBLE HTTPS EVENT-FLOW RELATIONS $(m, n > 1)$.

Events	Flows	Correctness	Description
1	0	ERR1	An unmatched event
0	1	ERR1	An unmatched flow
1	1	OK	An event matched with a flow
m	1	OK	Events matched with a flow
1	n	ERR2	An event matched with multiple flows
m	n	ERR2	Events matched with multiple flows

took place, in which colleagues within the organization were asked to view and generate a large amount of traffic. The rest of the dataset contains regular low-volume traffic.

The result was a raw network traffic and events dataset generated from 53 unique IP addresses. We filtered the resulting dataset, normalized it to JSON format, and anonymized it. We describe these processes here only briefly. They are defined in detail in the code of the respective scripts published in a separate repository [4].

Events were collected from Windows Server 2022 with IIS 10.0. We used the basic IIS logging settings with three optional features enabled. The key feature was the client port, and we also enabled logging of the volume of transferred data for both directions – client-server and server-client.

Network traffic was collected directly on the web server using Windows native *pktmon* tool. After the data collection phase ended, the collected files were converted to standard PCAP files, merged, and processed by Flowmon flow exporter [12] to obtain bidirectional flow records in CSV format. To avoid splitting long connections into multiple flows, which would have harmed the correlation with the web traffic log, we used a custom configuration of the exporter described in our previous work [15].

The flow records contained not only standard fields such as IP addresses, protocol, ports, packets, bytes, and timestamps but also extended information from TLS and QUIC protocols such as SNI, TLS client/server version, TLS client/server session ID, used cipher suites, and handshake type. The extended properties were used to filter out incomplete connections during the normalization phase.

As part of normalization, all events and flows were converted into a fixed structure in JSON format. At the same time, all time stamps were also normalized into a uniform Epoch format for easy comparison. As the last step, the dataset was divided according to TCP/UDP origin to HTTP/2 and HTTP/3, respectively. This division allowed us to correlate the two logs independently since it is impossible for them to share events. Also, it generated separate results that can be compared.

VI. EVALUATION

In this chapter, we seek to configure the correlation algorithm to detect the highest possible number of event-flow relationships in the HTTP/2 and HTTP/3 protocols and generate the lowest possible number of anomalies. First, we determine the optimal time-window for correlation. The time-window depends on the network and server environment and is, therefore, the same for both protocols. Then we identify and eliminate incomplete flows that were interrupted or terminated before they reached the web server application and, therefore, could not create an event.

A. Time-Window Measurement

To compute the optimal time-window, we used a method minimizing the total number of the erroneous *ERR1* and *ERR2* correlation results. First, we examined the separate effect of earliness and lateness. We fixed one parameter at the value 0, and for the second, we substituted values (in seconds) from the interval $< 0, 1, .., 20, 500, \infty)$. The ∞ value is represented by any value greater than the longest possible temporal distance of an event from a flow in our dataset. The IIS server only logs in whole seconds, so achieving better precision than in seconds was not possible.

It turned out that increasing the *lateness* had no effect on the number of successful correlations – it identified only a single new relationship. We manually examined it and classified it as an outlier. Consequently, we fixed the *lateness* to 0 seconds for further experiments.

The effect of *earliness* on the number of anomalies is shown in the Figure 2 for HTTP/3 and in the Figure 3 for HTTP/2. The number of ERR1 anomalies (uncorrelated data) is modeled by the single events and single flows curves, while the ERR2 anomalies are represented by the polygamous events curve. The ERR2 anomalies (events correlated to multiple flows) did not occur in our data, with the exception of earliness = ∞, neither for HTTP/2 nor for HTTP/3. This is caused by the relatively low frequency of queries in the dataset. There were no conflicts where a single client would repeat a port for a query. The ERR 1 anomaly curves also have a similar course for HTTP/2 and HTTP/3. Increasing earliness from 0 to 2 seconds marks a steep drop in single flows and events. A gradual decrease follows up to 6 seconds of earliness, with single events falling to no occurrences. Further increasing the earliness up to the ∞, thus encompassing the entire dataset, did not affect the correlation results.

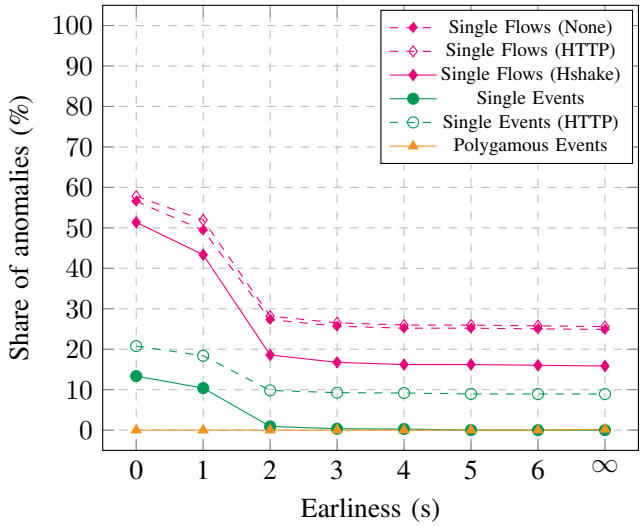

Fig. 2. HTTP/3: The share of correlation anomalies based on the *earliness* (s) and flow filters (in round brackets). The number of single events was affected by the HTTP error filter, but not by the handshake filter. Polygamous events were unaffected by the flow filters. Solid curves mark the best result.

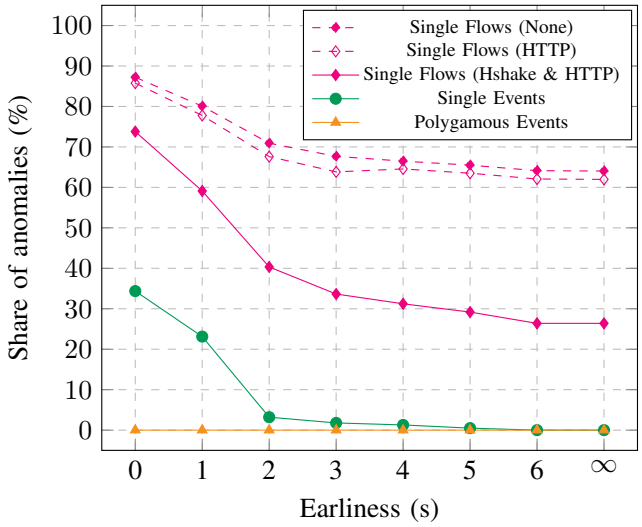

Fig. 3. HTTP/2: The share of correlation anomalies based on the *earliness* (s) and flow filters (in round brackets). The number of single and polygamous events was unaffected by the flow filters. Solid curves mark the best result.

While the shape of the anomaly curves is similar for HTTP/2 and HTTP/3, their levels differ. At 0-second *earliness*, almost 90% of the flows and 35% of the HTTP/2 events fail to correlate. For HTTP/3, only 55% of the flows and 15% of the events remain uncorrelated. At 6-second *earliness*, where the number of single events drops to 0 for the first time, the number of single flows is still 65% and 25% for HTTP/2 and HTTP/3, respectively. HTTP/2 traffic contains a large number of flows that do not reflect in the web application log.

The measurements determined that our dataset's optimal time-window(earliness, lateness) is (6, 0). Correlation with this time-window achieved the lowest number of ERR1 and ERR2 anomalies for both HTTP/2 and HTTP/3. An interesting fact is that, according to this window, the web server application recorded events up to 6 seconds before the requests were captured in network traffic. We address possible reasons for such behavior in Section VII.

B. Flow Prefiltering

Due to the high percentage of single flows at the time when all events were correlated, we focused on the classification of these flows. We consider them regular flows that ended before they created an event in the web server application because we do not expect malicious behavior in our dataset. This can be caused by, for example, scanners, crawlers, and connections failing during the handshake or being terminated prematurely by the client. To reduce the number of ERR1 anomalies, it is, therefore, necessary to classify these flows as benign from the web server's point of view and exclude them from the correlation; for this, we designed and evaluated two filters to apply before correlation.

The first filter, the HTTP error filter, checks whether a flow caused an event in the HTTP API log of Windows Server. These events may indicate that the connection was terminated before it was forwarded to the web server application itself. The second filter, the Handshake filter, checks whether complete and successful TCP and TLS handshakes have occurred. It verifies that at least the minimum required amount of packets had been exchanged and checks the contents of the TCP and TLS flags to see if they indicate a premature termination of the connection. Both filters are defined in detail by the Python code of the filter script [4].

The amount of ERR1 anomalies after applying the HTTP error filter to IP flows is shown in Figure 2 and Figure 3. For HTTP/2, this filter reduced the number of single flows by a few percent, while the number of single events remained zero. On the other hand, applying this filter to HTTP/3 traffic increased both single flows and single events. This was caused by the fact that for HTTP/3, the Windows Server HTTP API only logs that there was some error, not mentioning its category, severity, or any other detail aside from a timestamp. Therefore, an error in the log does not necessarily indicate flow termination. Consequently, this filter is unusable for HTTP/3, and we have not worked with it for this type of traffic.

Filtering based on TCP and TLS handshakes had a more significant impact on the number of anomalies than the HTTP error filter. For both HTTP/2 and HTTP/3, it reduced the ratio of single flows and did not filter any flows that would generate events on the web server, so the number of single events did not increase. This filter turned out to be essential for HTTP/2 where, when used simultaneously with the HTTP error filter, it reduced the number of anomalies in the optimal time-window by almost 40%. It had less impact on HTTP/3 traffic but still improved the results over unfiltered traffic by almost 10%.

In our environment, the time-window(earliness, lateness) = (6, 0) was established as optimal for correlation. The HTTP error filter and Handshake filter were applied to HTTP/2 IP flows before correlation. For HTTP/3 IP flows, only the Handshake filter was used. With this setting, the smallest number of ERR1 anomalies was achieved: 26.39% and 16.03%

single flows for HTTP/2 and HTTP/3, respectively. A related IP flow was found for all web application-generated events, regardless of the HTTP protocol version. ERR2 correlation anomalies did not occur over the used dataset.

VII. Lessons Learned

Our dataset collection took place on a single server, so we did not anticipate problems with time synchronization between events and flows. However, the measurements of the optimal time-window for correlation show that the web server application assigned timestamps to the events significantly lower than those recorded for their related flow. As a result, this would mean that the web server recorded events up to 6 seconds before the corresponding request arrived, which does not correspond to reality.

We checked the configuration and code of the network traffic probe, and we can confirm that it was measuring time precisely. We suspect the IIS web server application as the culprit, as it already advertises low accuracy of time measurement (in seconds). However, it is proprietary software, a black box, and documentation does not clarify what factors can influence the time measurement or how to increase its precision. We were also unable to repeat this desynchronization for single controlled requests, where everything behaved correctly. It follows that, in a general environment, time can only be used as an indicative quantity for correlation to limit apparent errors. To accurately determine the relationship, specific features are necessary, e.g., the IP addresses, ports, and SNI.

The SNI flow feature remains available for network traffic monitoring in the recent and, according to the RFC 9001, even future versions of QUIC for HTTP/3 [16]. Some effort was put into obfuscating the SNI in the QUIC implementation of TLS, and it is no longer available as clear text. However, the encoding process described in the RFC 9001 can be reversed by any middlebox, as it will know all the features necessary for decoding. Indeed, the probe software we used to export the IP flows was already able to decode and display the SNI for HTTP/3 with the same ease as for HTTP/2.

There was a surprising amount of network flows that could not be paired to any web request event. Although we managed to identify most of them as scans and incomplete connections, without the ability to decrypt the traffic, we cannot be entirely sure why some of these requests were not successful. TCP traffic has the advantage of additional information in the Windows connection log and also TCP flags. As for the QUIC protocol, we do not have sufficient information to determine the cause. This finding shows that the identification of malicious traffic such as the BPFDoor [17] is not a straightforward problem.

VIII. Conclusion

The HTTP/3 protocol in its final specification is the more efficient and secure successor of HTTP/2, and it should gradually replace it entirely. We have shown how service administrators can gain insight into HTTP/3 web traffic on their own networks using event-flow correlation. In our environment, it was possible to correlate 100% of events and 83.97% of network flows of HTTP/3, which is an even better result than for HTTP/2, where 100% of events and 73.61% of flows were correlated. We conclude that the event-flow correlation reduces the volume of data that must be classified by encrypted traffic analysis, even for the most current web traffic.

Acknowledgment

This research was supported by the CONCORDIA project that has received funding from the European Union's Horizon 2020 research and innovation programme under the grant agreement No. 830927 and by the ERDF project "CyberSecurity, CyberCrime and Critical Information Infrastructures Center of Excellence" (No. CZ.02.1.01/0.0/0.0/16_019/0000822).

References

[1] M. Bishop, "Hypertext Transfer Protocol Version 3 (HTTP/3)," Internet Engineering Task Force, Jun. 2022. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc9114

[2] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," Internet Engineering Task Force, May 2021. [Online]. Available: https://datatracker.ietf.org/doc/rfc9000/

[3] S. Špaček, P. Velan, P. Čeleda, and D. Tovarňák, "HTTPS Event-Flow Correlation: Improving Situational Awareness in Encrypted Web Traffic," in *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2022, pp. 1–7.

[4] S. Špaček, P. Velan, M. Holkovič, and T. Plesník, "Event-Flow Correlation for HTTP/3 Web Traffic - Data and Code," Sep. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.7072548

[5] C. Huitema, "Architecture for IP Flow Information Export," Internet Engineering Task Force, Mar. 2009. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc5470

[6] B. Trammel and E. Boschi, "Bidirectional Flow Export Using IP Flow Information Export (IPFIX)," Internet Engineering Task Force, Jan. 2008. [Online]. Available: https://tools.ietf.org/html/rfc5103

[7] W3C, "Logging Control In W3C httpd," World Wide Web Consortium (W3C). [Online]. Available: https://www.w3.org/Daemon/User/Config/Logging.html

[8] Microsoft, "Internet Information Services." [Online]. Available: https://www.iis.net/

[9] E. Chatzoglou, V. Kouliaridis, G. Kambourakis, G. Karopoulos, and S. Gritzalis, "A hands-on gaze on http/3 security through the lens of http/2 and a public dataset," *arXiv preprint arXiv:2208.06722*, 2022.

[10] E. Chatzoglou, V. Kouliaridis, G. Karopoulos, and G. Kambourakis, "Revisiting quic attacks: A comprehensive review on quic security and a hands-on study," 2022.

[11] R. Marx, M. Piraux, P. Quax, and W. Lamotte, "Debugging quic and http/3 with qlog and qvis," in *Proceedings of the Applied Networking Research Workshop*, 2020, pp. 58–66.

[12] Flowmon Networks. Flowmon Probe. [Online]. Available: https://www.flowmon.com/en/products/flowmon/probe

[13] I. Kotenko, D. Gaifulina, and I. Zelichenok, "Systematic literature review of security event correlation methods," *IEEE Access*, 2022.

[14] H. Albasheer, M. Md Siraj, A. Mubarakali, O. Elsier Tayfour, S. Salih, M. Hamdan, S. Khan, A. Zainal, and S. Kamarudeen, "Cyber-attack prediction based on network intrusion detection systems for alert correlation techniques: A survey," *Sensors*, vol. 22, no. 4, p. 1494, 2022.

[15] S. Špaček, P. Velan, P. Čeleda, and D. Tovarňák, "Encrypted Web traffic dataset: Event logs and packet traces," *Data in Brief*, vol. 42, p. 108188, 2022.

[16] M. Thomson and S. Turner, "Using TLS to Secure QUIC," Internet Engineering Task Force, May 2021. [Online]. Available: https://datatracker.ietf.org/doc/rfc9001/

[17] Beaumont, Kevin. BPFDoor. [Online]. Available: https://doublepulsar.com/bpfdoor-an-active-chinese-global-surveillance-tool-54b078f1a896