Updating NEL Scripts After Breaking Changes in HTTP Archive

Martin Bednář

This document describes an update to the NEL (Network Error Logging) HTTP header collection and analysis tool.

It builds on earlier results by Kamil Jeřábek and Libor Polčák [1, 5], who started working on this topic together. Later, Matěj Jurík [2] continued their work. The research and scripts of all three authors were the starting point for this project. Before reading this work, it is recommended to first read the work of these authors to better understand the background and context of this project.

In this work, their original scripts were updated because the HTTP Archive made breaking changes to its database, specifically, it changed the database schema. The SQL scripts had to be modified to query the same data using the new structure.

This document also explains what has been achieved so far, what the current state of the project is, and what future work is needed. It shows possible directions for further development and research.

This work is divided into four chapters. Chapter 1 explains how to set up the Google Cloud environment, which is needed to run the data collection scripts. Chapter 2 describes the changes made to Matěj Jurík's original scripts after the HTTP Archive changed its data structure. Chapter 3 shows the results and compares the outputs from the original and updated scripts. The final Chapter 4 explains how this work improves the current analysis of NEL headers and suggests possible next steps, including open questions that still need to be solved.

Contents

1 Setting up a Google Cloud environment

Before collecting data from the HTTP Archive, you need to set up your Google Cloud environment in the Console[1].

The Python script `query_and_store.py`[2] is used to download and store HTTP requests as a parquet file from the HTTP Archive. However, before running this script, you must first prepare the environment in Google Cloud Console.

Based on my experience, this setup is not trivial. It requires creating a project, configuring permissions, and connecting services correctly. For this reason, the first section of this document is dedicated to guiding you through the Google Cloud Console setup step by step.

1.1 Create a Google Cloud account

Before you can use Google Cloud services like BigQuery and Cloud Storage, you need a Google Cloud account. If you are a new user, Google offers a free trial with $300 in credits valid for 90 days. Follow these steps to create your trial account.

- Go to `https://cloud.google.com/free`.
- Click the *Get started for free* button.
- If you already have a Google account, click *Sign in*. If you don't have one, click *Create account* and follow the instructions.
- After signing in, Google will ask for billing details. Choose your country. Enter a credit card or debit card (used only for identity verification, you will not be charged unless you manually upgrade later).
- Once billing is verified, Google will activate your trial. You will receive $300 in free credits for Google Cloud services. The credits are valid for 90 days.
- In the Google Cloud Console, go to *Billing* > *Reports*. You can see how much credit you've used. You can also set budgets and alerts to avoid unexpected charges.
- The trial ends when you use all $300 or after 90 days. You can choose to upgrade to a paid account to continue using services. If you don't upgrade, your resources will be paused but not deleted immediately.

This trial account is required to run the `query_and_store.py` script and access BigQuery and Cloud Storage. Make sure your account is set up before continuing to the next steps.

1.2 Create a Google Cloud Project

- Go to the Google Cloud Console.
- In the top navigation bar, click the project dropdown (next to the Google Cloud logo).
- Click *New Project*.
- Enter a Project Name (e.g. NEL) and choose your Billing Account if required.
- Click *Create*.
- Wait a few seconds until the project is created. You'll be automatically switched to it.

[1] `https://console.cloud.google.com`

[2] `https://github.com/martinbednar/NEL-Analysis-Scripts/blob/updateQueryAfterHttpArchiveBreakingChanges/query_and_store.py`

1.3 Create a Service Account with Admin Access to BigQuery and Storage

- In the left-hand menu, go to *IAM & Admin* > *Service Accounts*.
- Click *Create Service Account* at the top.
- Fill in service account name (e.g. NelServiceAccount).
- Click *Create and Continue*.
- In the Grant this service account access to project section. Add the following roles: BigQuery Admin, Storage Admin. You can use the search bar to find each role and click to add.
- Click *Continue*.
- Skip the optional user access section and click *Done*.

1.4 Download the Service Account Key

- After creating the service account, you'll see it listed as in Figure 1.
- Click the three dots ⋮ next to your service account and choose *Manage keys*.
- Click *Add Key* > *Create new key*.
- Choose JSON as the key type.
- Click *Create*. A JSON file will be downloaded automatically.
- Save this file securely. You will need to reference it in the script's configuration.

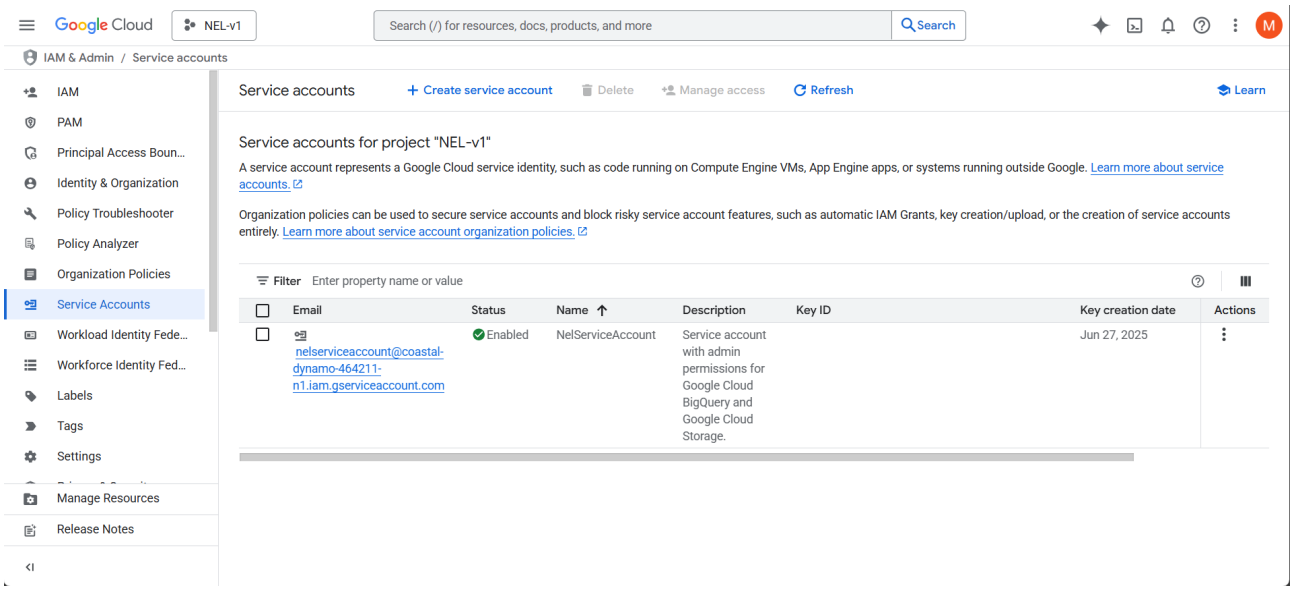


Figure 1: Newly created Google Cloud Service Account.

1.5 Create a Google Cloud Storage Bucket

- In the left-hand menu, go to *Storage* > *Buckets*.
- Click *Create* at the top.
- Name your bucket (e.g. NelDataBucket) – must be globally unique. Choose a location type: usually "Region" is fine. Select a region (e.g. europe-west1).
- Leave other settings as default unless you have specific needs.
- Click *Create*.
- Once created, copy the bucket name and paste it into the script's config file.

2 Adapting scripts to HTTP Archive Changes

This chapter explains how the Python script `query_and_store.py` was updated to stay functional after major changes in the HTTP Archive.

The script is used to download HTTP requests including NEL (Network Error Logging) HTTP headers from the HTTP Archive using SQL queries. However, at the end of 2024, the HTTP Archive changed its database structure. These changes made the original version of the script stop working.

In this chapter, it is described what was changed in the HTTP Archive, why the original script failed, and how the SQL queries were rewritten to match the new dataset format. This update was necessary to continue collecting data successfully.

2.1 Breaking Changes in the HTTP Archive

At the end of 2024 and beginning of 2025, the HTTP Archive made important breaking changes to its data structure [4]. These changes affected how data is stored and accessed.

The old datasets (like pages, requests, and others) were replaced by a new unified dataset called `httparchive.crawl` [3]. This new dataset includes all the data from the previous tables and is designed to be more efficient and easier to use.

Starting in November 2024, only the new `httparchive.crawl` dataset receives monthly updates. The old datasets stopped updating after that. In February 2025, public access to the legacy datasets was removed completely.

Because of this change, the original Python script `query_and_store.py` (created by Matěj Jurík) stopped working. The script was written for the old dataset structure, and its SQL queries no longer matched the new schema.

2.2 Updating SQL queries

To keep the `query_and_store.py` script working with the new `httparchive.crawl` dataset, several important changes were made. All updates to the script are included in Pull Request[3].

To collect NEL (Network Error Logging) headers from HTTP requests, the script must extract them from the HTTP Archive dataset. In the old version of the HTTP Archive, all HTTP headers were stored as a single serialized string. This means that the headers were saved together in one long text field, without any structure. Because of this format, the only way to extract specific headers (like NEL or Report-To) was to use regular expressions (regex). These patterns searched the string and tried to find the right header values.

This method worked, but it was not very efficient or reliable. It depended on the exact format of the string and could break easily if the structure changed. After the dataset structure changed, two different approaches were developed to handle header extraction.

2.2.1 Original Approach: Using Regular Expressions

This method was designed to closely follow the original logic used in earlier versions of the script. In the legacy HTTP Archive datasets, all HTTP headers were stored as a single serialized string. Because of this format, the only way to extract specific headers – such as NEL or Report-To – was by using regular expressions (regex) to search the string and extract the desired values.

When the HTTP Archive schema changed in late 2024, headers were no longer stored as serialized strings. Instead, they were saved as structured key-value pairs in a dedicated table. This change made it possible to query headers directly using SQL.

However, to keep the script as close as possible to the original version and to minimize the number of changes, a transitional solution was introduced. In this first method the new key-value headers are

[3] https://github.com/JurikMatej/IBP-NEL-Analysis-Scripts/pull/1

concatenated back into a single string, mimicking the old format. The original regex patterns are then applied to this string to extract the required headers.

This approach allowed the script to remain functional with minimal modifications. It preserved the original logic and structure, making it easier to maintain and verify the updated version. It also ensured that existing regex patterns could be reused without rewriting the entire extraction logic.

While this method works, it is not ideal. Serializing structured data back into a string and then parsing it with regex is less efficient and more error-prone than working with the data directly. For this reason, a second, more robust method was developed – based on direct SQL queries – which is now recommended for long-term use.

2.2.2 New Dataset Format: Key-Value Header Table

In the new `httparchive.crawl` dataset, HTTP headers are stored in a much better way. Each request now has its headers saved as a table of key-value pairs. This means each header name and its value are stored separately and clearly.

Thanks to this change, the script could be updated to use a more direct and reliable method for extracting headers. Headers are queried directly as key-value pairs using SQL. There is no need for serialization or regex. This approach is faster, cleaner, and more reliable.

The second method is now preferred because it takes full advantage of the improved dataset structure. It avoids the problems of regex and works directly with the data in its native format.

These changes ensure that the query runs correctly in BigQuery without errors. The updated script is now compatible with the current version of the HTTP Archive and ready for use.

3 Results and comparison of methods

In this chapter, the results of NEL headers analysis are presented and different methods used to extract NEL headers from the HTTP Archive are compared.

3.1 Overview of methods

We worked with three versions of SQL queries:

- OLD – the original query by Matěj Jurík (no longer functional after the schema change).
- NEW1 – Bednar's first updated query, which still concatenates headers into a string and uses regex for extraction.
- NEW2 – Bednar's latest query, which removes concatenation and regex, and instead queries headers directly as key-value pairs in SQL.

3.2 Comparison of results

The results show that changing the method does not produce identical outputs. However, the differences are small. For example:

- Cloudflare market share in April 2021:
  - Jurík (OLD): 78 %
  - Bednar (NEW1): 77,95 %
    → Difference: only 0,05 percentage points.
- Maximum domains served by Cloudflare in April 2021:
  - Jurík (OLD): 811 264
  - Bednar (NEW1): 809 753
    → Difference: 1 511 domains (approximately 0.2 %).
- NEL domains compared to total domains in April 2021:
  - Jurík (OLD): 1 031 681 / 10 099 360
  - Bednar (NEW1): 1 030 323 / 10 080 175
    → Difference: again around 0,2 %.

Overall, the average deviation is about 0,2–3 %, depending on the metric.

The following tables 1, 2, 3, 4, 5, 6, 7 present the differences between the methods in more detail based on the analysis results. The differences between the OLD and NEW1 methods in particular were analyzed, but also the differences between the NEW1 and NEW2 methods.

In addition to the analysis results, the datasets themselves (parquet files) are also compared. These are sets, and therefore the set operations shown in the Figure 2 are used for comparison.

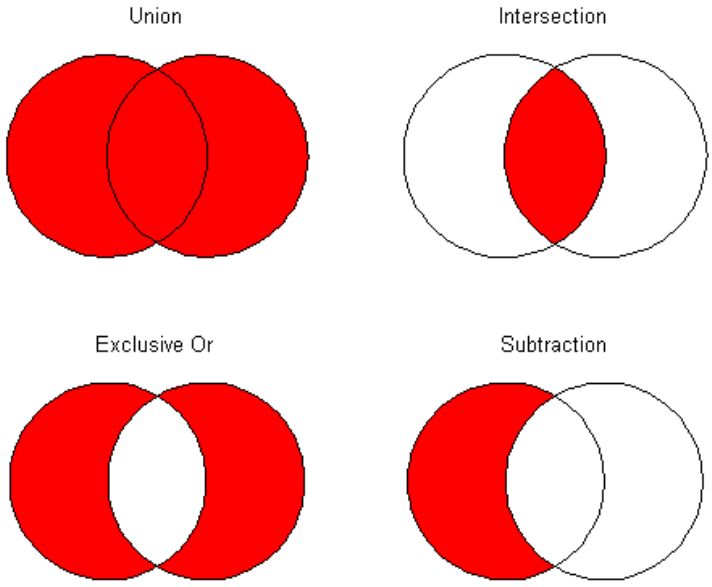


Figure 2: Set operations.

Total number of analyzed domains			
Year/Month	Jurik (OLD)	Bednar (NEW1)	Deviation
2019/02	5 707 189	5 809 506	1,76 %
2020/02	6 636 205	6 635 717	0,01 %
2021/02	10 147 089	10 131 527	0,15 %
2022/02	10 363 447	10 326 193	0,36 %
2023/02	19 159 612	20 350 639	5,85 %
2024/02	20 267 978	21 471 337	5,60 %

Table 1: Total number of analyzed domains.

Number of domains with correctly deployed NEL			
Year/Month	Jurik (OLD)	Bednar (NEW1)	Deviation
2019/02	370	469	21,11 %
2020/02	109 604	109 596	0,01 %
2021/02	1 008 105	1 007 001	0,11 %
2022/02	972 168	969 693	0,26 %
2023/02	2 258 979	2 305 689	2,03 %
2024/02	2 544 162	2 615 185	2,72 %

Table 2: Number of domains with correctly deployed NEL.

Comparing Bednar's method (NEW1) with Jurik's one (OLD) on results for Februaries 2019-2024							
Set	02/2019	02/2020	02/2021	04/2021	02/2022	02/2023	02/2024
LEFT: Total domains in JURIK's parquet	370	109 604	1 008 105	1 031 581	972 168	2 258 980	2 544 163
RIGHT: Total domains in BEDNAR's NEW1 (regex) parquet	469	109 601	1 006 970	1 030 333	969 686	2 305 671	2 615 262
INTERSECTION: Domains in both files	370	109 585	1 006 934	1 030 309	969 675	2 258 804	2 544 005
UNION: Domains in at least one file	469	109 620	1 008 141	1 031 605	972 179	2 305 847	2 615 420
INTERSECTION / UNION ratio	0,789	1,000	0,999	0,999	0,997	0,980	0,973
LEFT / INTERSECTION ratio	1,000	1,000	1,001	1,001	1,003	1,000	1,000
RIGHT / INTERSECTION ratio	1,268	1,000	1,000	1,000	1,000	1,021	1,028
SUBTRACTION LEFT: Domains only in JURIK's parquet	0	19	1 171	1 272	2 493	176	158
SUBTRACTION RIGHT: Domains only in BEDNAR's NEW1	99	16	36	24	11	46 867	71 257
SUBTRACTION LEFT / INTERSECTION	0,000	0,000	0,001	0,001	0,003	0,000	0,000
SUBTRACTION RIGHT / INTERSECTION	0,268	0,000	0,000	0,000	0,000	0,021	0,028

Table 3: Comparing Bednar's method (NEW1) with Jurik's one (OLD) on results for Februaries 2019-2024.

Comparing new methods on results for April 2021			
Set	Bednar (NEW1)	Bednar (NEW2)	Deviation
LEFT: Total domains in JURIK's parquet	1 031 581	1 031 581	0,00 %
RIGHT: Total domains in BEDNAR's parquet	1 030 333	1 030 320	0,00 %
INTERSECTION: Domains in both files	1 030 309	1 030 281	0,00 %
UNION: Domains in at least one file	1 031 605	1 031 620	0,00 %
INTERSECTION / UNION ratio	0,999	0,999	0,00 %
LEFT / INTERSECTION ratio	1,001	1,001	0,00 %
RIGHT / INTERSECTION ratio	1,000	1,000	0,00 %
SUBTRACTION LEFT: Domains only in JURIK's parquet	1 272	1 300	2,15 %
SUBTRACTION RIGHT: Domains only in BEDNAR's parquet	24	39	38,46 %
SUBTRACTION LEFT / INTERSECTION	0,001	0,001	2,16 %
SUBTRACTION RIGHT / INTERSECTION	2,329E-05	3,785E-05	38,46 %

Table 4: Comparing new methods on results for April 2021.

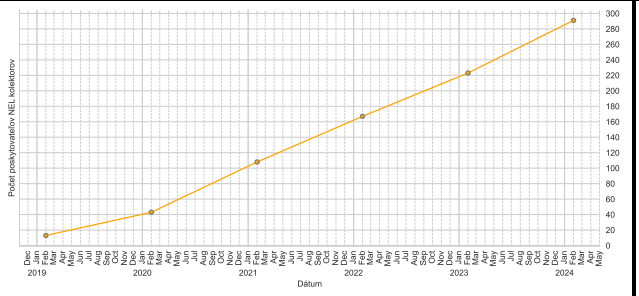
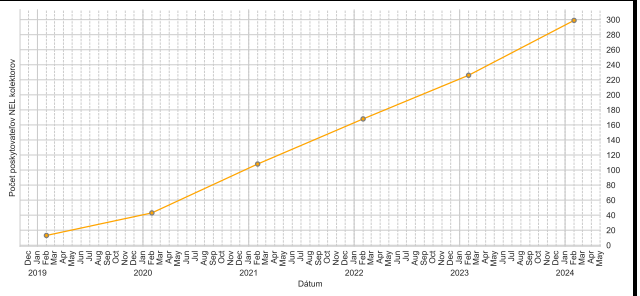
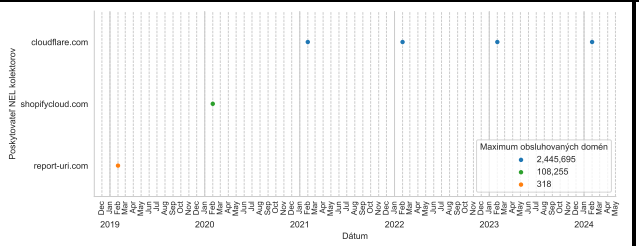
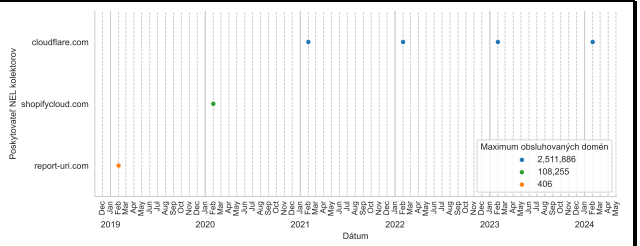
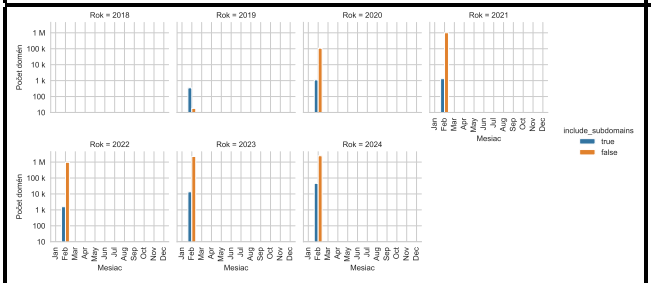
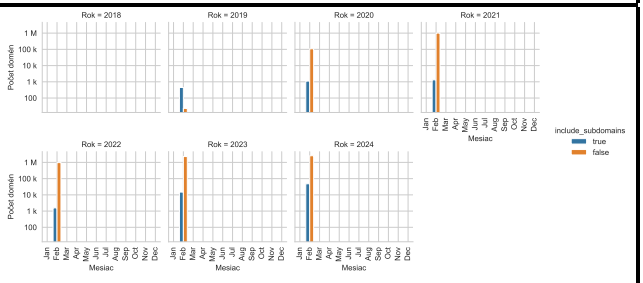
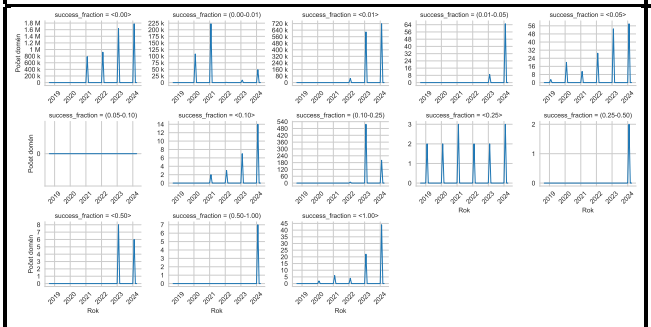
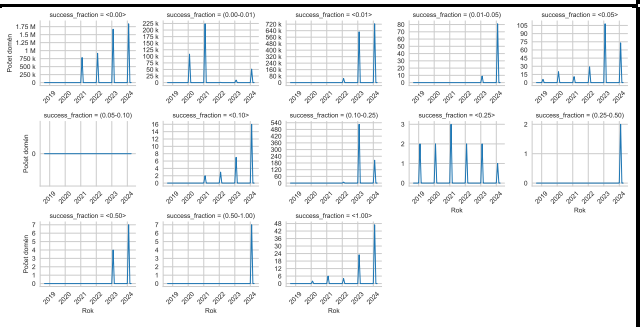
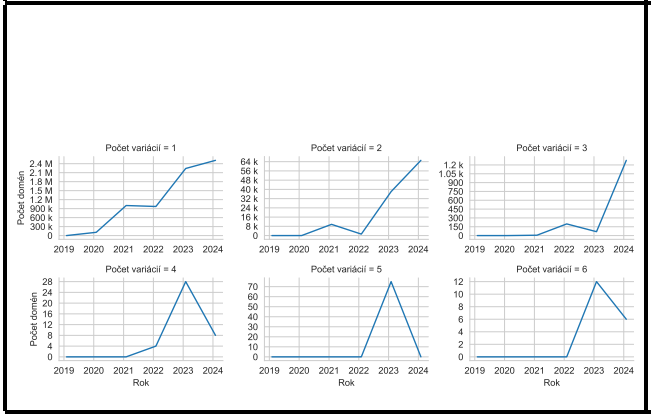
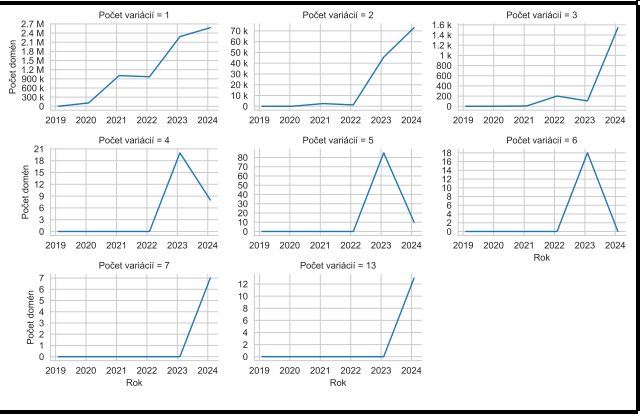
Comparing new methods on results for February 2022			
Set	Bednar (NEW1)	Bednar (NEW2)	Deviation
LEFT: Total domains in JURIK's parquet	972168	972168	0,00 %
RIGHT: Total domains in BEDNAR's parquet	969686	969711	0,00 %
INTERSECTION: Domains in both files	969675	969681	0,00 %
UNION: Domains in at least one file	972179	972198	0,00 %
INTERSECTION / UNION ratio	0,997	0,997	0,00 %
LEFT / INTERSECTION ratio	1,003	1,003	0,00 %
RIGHT / INTERSECTION ratio	1,000	1,000	0,00 %
SUBTRACTION LEFT: Domains only in JURIK's parquet	2493	2487	-0,24 %
SUBTRACTION RIGHT: Domains only in BEDNAR's parquet	11	30	63,33 %
SUBTRACTION LEFT / INTERSECTION	0,003	0,003	-0,24 %
SUBTRACTION RIGHT / INTERSECTION	1,134E-05	3,094E-05	63,33 %

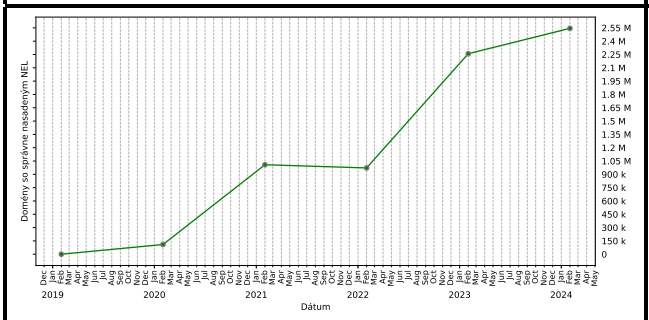
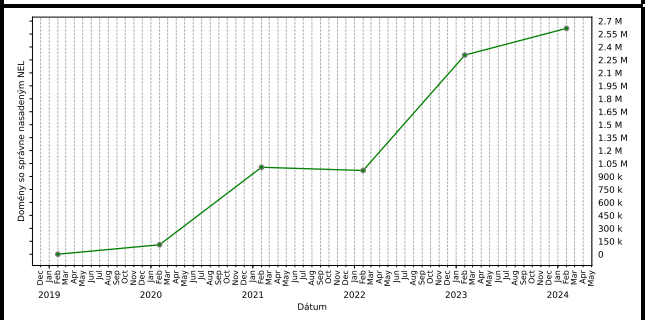
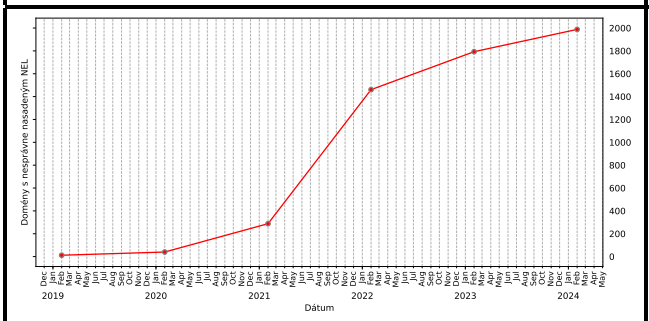
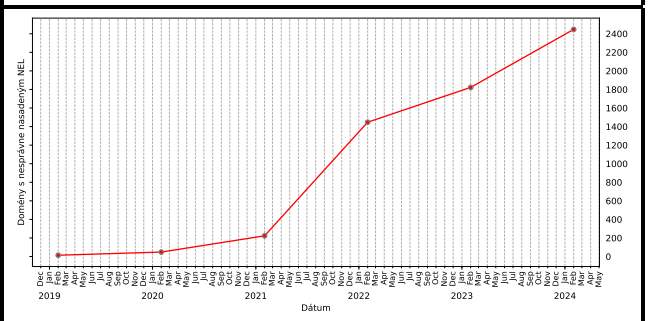
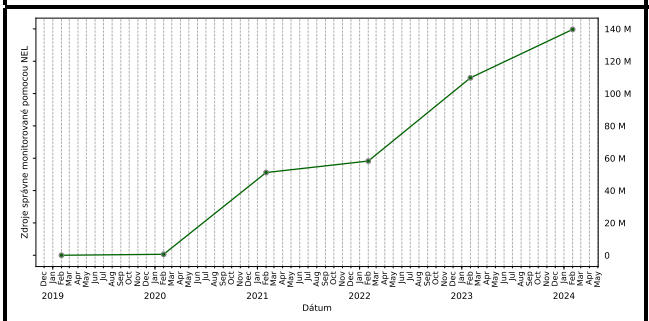
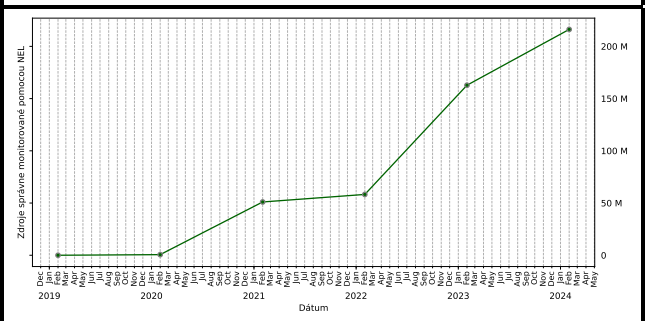
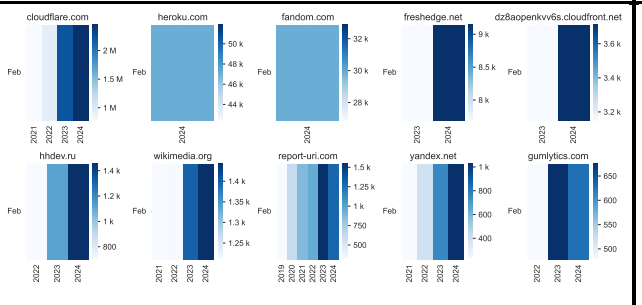
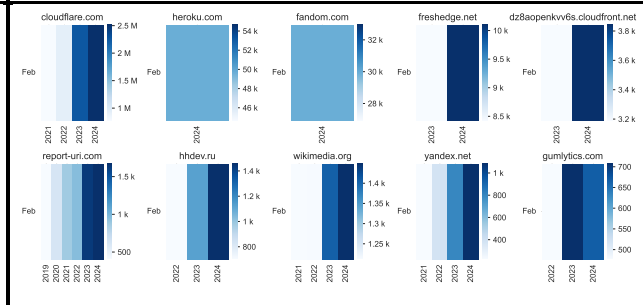
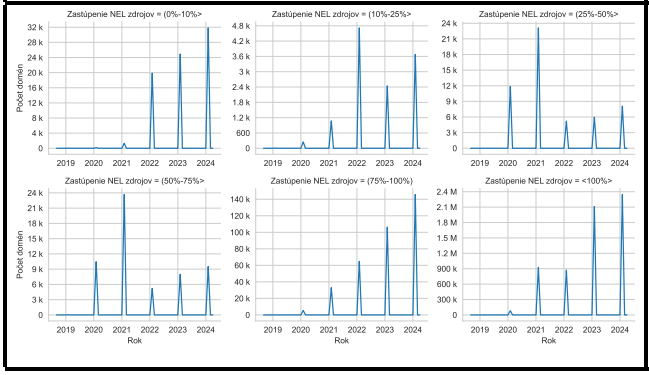
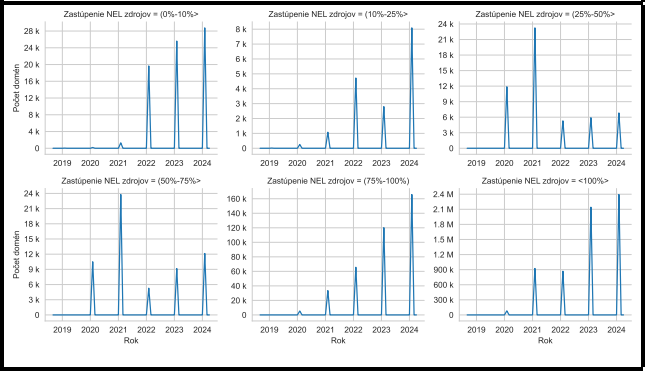
Table 5: Comparing new methods on results for February 2022.

Comparison of Jurik (OLD), Bednar (NEW1) and Bednar (NEW2) analysis results for April 2024			
Metrics	Jurik (OLD)	Bednar (NEW1)	Bednar (NEW2)
Number of domains served by `cloudflare.com`	811 264	809 753	811 040
Total number of analyzed domains	10 099 360	10 080 175	10 080 175
Number of domains with correctly deployed NEL	1 031 581	1 030 323	1 030 334

Table 6: Comparison of Jurik (OLD), Bednar (NEW1) and Bednar (NEW2) analysis results for April 2024.

Comparison of Jurik (OLD) and Bednar (NEW1) analysis results for Februaries 2019-2024

Jurik (OLD)	Bednar (NEW1)
	
	
	
	
	

Jurik (OLD)	Bednar (NEW1)
	
	
	
	
	

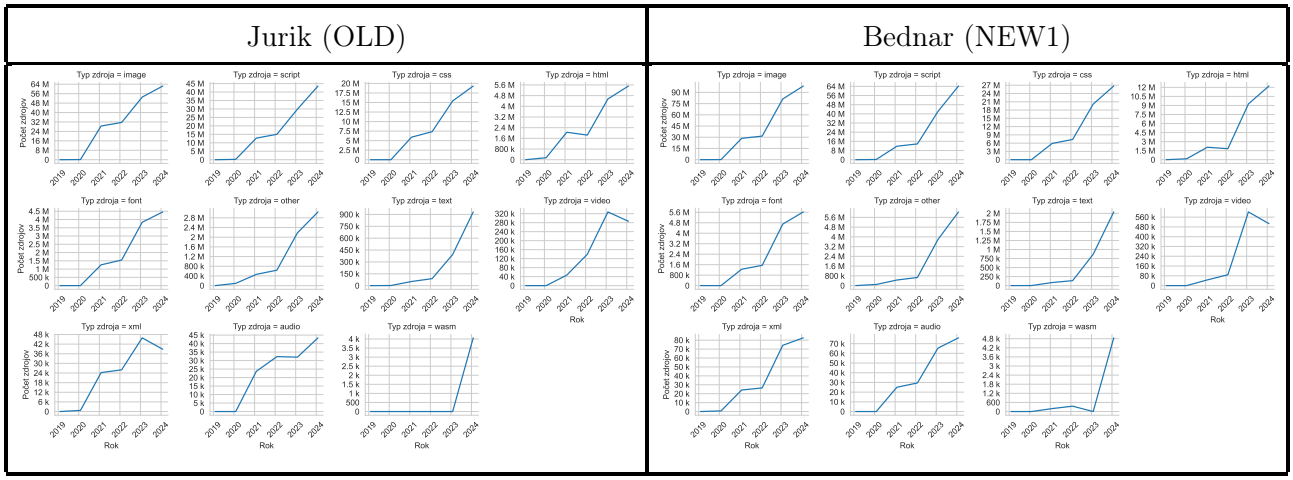


Table 7: Comparison of Jurik (OLD) and Bednar (NEW1) analysis results for Februaries 2019-2024.

3.3 Interpretation of differences

The results of NEW2 are very close to NEW1 and also to the OLD method. Still, they are not identical.

Possible reasons include:

- Some data may not have been fully migrated during the schema change, leading to missing values.
- Alternatively, a small number of HTTP requests may not have been loaded in Bednar's queries, though the cause is unclear.

Each change in methodology introduces small deviations. It is unlikely that two different methods will ever produce exactly the same results. The goal is therefore to select the most accurate and reliable method.

The differences are most likely related to the schema change and the adjustments made in SQL queries. The comparison shows that both NEW1 and NEW2 methods are valid, with NEW2 being technically cleaner because it avoids regex.

4 Conclusion and future work

This work focused on analyzing the deployment of NEL (Network Error Logging) headers using data from the HTTP Archive. To make this possible, several steps were needed.

First, the Google Cloud environment was prepared. This included creating a project, service account, storage bucket, and downloading the necessary credentials. Without this setup, the analysis scripts could not run.

Second, we looked at the breaking changes in the HTTP Archive. At the end of 2024, the dataset structure changed. The old datasets stopped updating, and a new unified dataset called `httparchive.crawl` became the only source of data. Because of this, the original script `query_and_store.py` stopped working. We described how the script was updated to match the new schema and how header extraction logic was improved.

Third, the results of different methods were compared. Three versions of SQL queries were tested:

- OLD – the original query by Matěj Juríк (no longer functional).
- NEW1 – first updated query, still using concatenation and regex.
- NEW2 – the latest query, using direct SQL on key-value pairs without regex.

The comparison showed that results are not identical, but the differences are small (around 0,2–3 %). For example, Cloudflare's market share was 78 % in Juríк's data and 77.95 % in Bednar's one. These small deviations are acceptable and confirm that the updated methods work correctly. Both NEW1 and NEW2 produce very similar results, which shows that regex extraction was already quite reliable. However, NEW2 is technically cleaner because it avoids unnecessary concatenation and regex.

Finally, we discussed the meaning of these differences. Even small changes in methodology can introduce deviations, and it is unlikely that two methods will ever produce exactly the same results. The important point is that the analysis remains consistent and trustworthy.

4.1 Future Work

This work successfully adapted the analysis scripts to the new HTTP Archive dataset. The updated methods allow us to continue studying NEL deployment without interruption. The differences between old and new results are small and acceptable, which means the analysis can be trusted, but there are still open questions.

Why do small deviations (around 0,2 %) appear between methods?

Are these differences caused by the dataset migration, missing requests, or by the query logic itself?

Can further optimization reduce costs in BigQuery queries?

Future work should focus on investigating these questions, improving the accuracy of header extraction, and exploring new ways to analyze NEL deployment. This will help make the analysis more precise and provide deeper insights into how NEL is used across the web.

References

[1] Jeřábek, K. and Polčák, L. *Network Error Logging: HTTP Archive Analysis*. 2023. [Online; visited 04.11.2025]. Available at: https://arxiv.org/abs/2305.01249.

[2] Juřík, M. *Analysis of Network Error Logging Deployment*. Brno, CZ, 2024. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. [Online; visited 04.11.2025]. Available at: http://hdl.handle.net/11012/247835.

[3] Ostapenko, M. *Migrate queries to 'crawl' dataset*. 2024. [Online; visited 04.11.2025]. Available at: https://har.fyi/guides/migrating-to-crawl-dataset/.

[4] Ostapenko, M. *New Release: 'httparchive.crawl' Dataset*. 2024. [Online; visited 04.11.2025]. Available at: https://discuss.httparchive.org/t/new-release-httparchive-crawl-dataset/2814.

[5] Polčák, L. and Jeřábek, K. Data Protection and Security Issues with Network Error Logging. In: Vimercati, S. D. C. di and Samarati, P., ed. *Proceedings of the 20th International Conference on Security and Cryptography*. Řím: SciTePress - Science and Technology Publications, 2023, p. 683–690. DOI: 10.5220/0012078100003555. ISBN 978-989-758-666-8. [Online; visited 04.11.2025]. Available at: https://arxiv.org/abs/2305.05343.