Data-driven digital forensics: anomaly detection in Mozilla Firefox

Zuzana Henelová[1], Pavol Sokol[1,*], Tomáš Bajtoš[1] and Sophia Petra Krišáková[1]

[1]Institute of Computer Science, Faculty of Science, Pavol Jozef Šafárik University in Košice, Jesenná 5, 040 01 Košice, Slovakia

Abstract

The use of web browsers generates a large amount of data that can serve as valuable forensic artifacts during the investigation of cybersecurity incidents. These artifacts contain information about visited websites, downloaded files, and timestamps of user activity, which can help identify the attack vector. Existing tools, such as Autopsy, allow the visualization of these data; however, their evaluation is time-consuming and relies heavily on the analyst's expertise. In this paper, we employ a data-driven digital forensics methodology to analyze these forensic artifacts. The goal is to automate the identification of suspicious events using machine learning algorithms. Data from the Mozilla Firefox web browser is processed into a unified timeline, on which anomalies are detected. The results show that the ECOD and COPOD methods offer a suitable compromise between accuracy and the volume of output data. The proposed approach enables faster data orientation and more efficient forensic analysis.

Keywords

digital forensics, anomaly detection, web browser, forensic artifact

1. Introduction

For several years, we have been observing a growing trend in cyber threats. New threats are emerging, and existing ones are becoming more sophisticated. This development is closely linked to the use of modern technologies. A standard application that nearly every individual uses today is a web browser. It enables communication with various applications and services, and it is hard to imagine the use of information technologies without it.

While browsers bring many advantages, they also introduce significant risks. The web browser represents a primary attack vector used by cybercriminals to penetrate systems [1], or to spread malicious code [2]. A typical example is the drive-by-compromise attack, which exploits zero-day vulnerabilities. Using exploit kits, attackers infect a large number of user devices. A significant portion of these attacks is delivered to users through the web browser, which underscores the importance and necessity of browser forensics during cybersecurity incidents [3, 2].

In handling cybersecurity incidents, it is essential to identify the cause of the incident, the course of the attacker's actions, and their objective. To do this, it is necessary to analyze the available digital traces. However, many traces acquired from browsers are irrelevant to the investigation. Therefore, it is crucial to distinguish between relevant and irrelevant traces. Most current techniques rely on manual review [4]. On the other hand, machine learning and AI methods can accelerate forensic analysis, particularly through pattern recognition and the detection of abnormal behavior [5].

This paper addresses this problem, focusing on the automatic identification of unusual digital trace occurrences in web browsers. The study is limited to the Mozilla Firefox browser and the Windows operating system. Based on this, we propose a model that identifies digital traces relevant to a specific case.

Identifying possible digital trace attributes and analyzing relationships between them is a key research challenge in cybersecurity incident response and digital forensics [6].

ITAT'25: Information Technologies – Applications and Theory, September 26–30, 2025, Telgárt, Slovakia

*Corresponding author.

†These authors contributed equally.

zuzana.henelova@upjs.sk (Z. Henelová); pavol.sokol@upjs.sk (P. Sokol); tomas.bajtos@upjs.sk (T. Bajtoš); sophia.petra.krisakova@upjs.sk (S. P. Krišáková)

0000-0002-1967-8802 (P. Sokol)

To summarize the above issues, we pose the following research questions:

- Is the time window an appropriate aggregation method for digital traces when analyzing user activity in web browsers?
- Which anomaly detection methods are suitable for identifying time windows that represent digital traces relevant to forensic investigation?

To answer these questions, we focused on unsupervised outlier detection methods. Outliers can be defined as observations in a dataset that appear inconsistent with the rest of the data [7]. In our research, we consider relevant digital evidence (i.e., browser activity time windows) as outliers relative to other digital evidence. We tested several outlier detection methods representing different approaches. The individual methods are described in Section 3, and their comparative results are discussed in Section 4. For artifact verification and testing, a device running Windows 11 equipped with Mozilla Firefox version 124.0.2 is used.

The contribution of this paper lies in proposing a method for representing groups of digital traces suitable for automated processing (i.e., browser activity time windows). Additionally, the paper compares machine learning methods for anomaly detection in the context of identifying groups of digital traces (time frames) relevant to digital forensic investigations.

This paper builds on the results of our previous research presented in [8, 9]. The first study [8] focused on identifying relevant digital evidence in the Windows OS and the NTFS file system. The second study [9] analyzed digital records from the file system and explored the relationships between individual pieces of digital evidence.

This paper is organized into five sections. After the introduction, we present the related research and papers in Section 2. Section 3 briefly describes the dataset and presents methods and evaluation metrics for this research. Section 4 provides results and discusses the outlier detection method and the time window size. Finally, Section 5 contains conclusions and focuses on the challenges and directions for future research.

2. Related Works

The research presented in this paper falls within the domain of digital forensic analysis of web browsers. The most closely related work is that of Kim et al., who proposed the application of machine learning to browser forensics through the development of AIBFT (Artificial Intelligence Browser Forensic Toolkit). This tool enables automatic detection of malicious websites using AI, attack probability estimation, and timeline visualization [2]. Aside from this study, there is no other research focusing on the use of machine learning or artificial intelligence for analyzing digital traces from web browsers. In browser forensics, several studies focus on identifying and collecting artifacts related to browser activity, such as URLs, timestamps, browser versions, downloads, and search queries [10]. Others analyze browser artifacts in Firefox, Chrome, and Edge on Windows 11 [11]. An interesting extension is the focus on portable web browsers. Hariharan et al. examined Brave, TOR, Vivaldi, and Maxthon, in conjunction with various memory acquisition tools, to determine the quantity and quality of data that can be recovered from memory dumps under two conditions – with open and closed tabs [12]. Several works also analyze different browsing modes. Chand et al. focused on three browsing scenarios (regular, private, portable), critically evaluating browser artifacts using forensic tools [13]. Another example is Nelson et al., who simulated identical browsing behavior in Google Chrome, Mozilla Firefox, and the TOR Browser and analyzed the retrievable forensic artifacts in both regular and private modes [14]. In this context, the TOR browser stands out. Javed et al. analyzed artifacts generated by the TOR browser on Windows systems using a methodology based on NIST SP 800-86 [15]. Since our research employs anomaly detection to identify interesting or relevant digital traces, it is also important to mention a second group of relevant articles focused on automating forensic analysis using anomaly detection. The research group led by Studiawan proposed a graph-based anomaly detection method for forensic log data [16]. This research was extended by evaluating the performance of seven anomaly detectors on system log data [17] and by using deep autoencoders for anomaly detection [4]. Deep autoencoders are also used in other studies, such as that of Yuan et al., who analyzed time-series-based user behavior anomalies [18]. Another example is anomaly detection in file systems, where digital evidence is most often stored [19]. Within Windows OS anomaly analysis, the registry is a key area of focus. Chouhan et al. explored anomalies in registry access, DLL libraries, and file access to detect suspicious process behavior [20].

3. Methodology

3.1. Dataset

The research was conducted based on real usage data collected from Mozilla Firefox browser version 124.0.2 on the Windows 11 operating system. The dataset under consideration spans a period from July 26, 2023, to February 11, 2025. In terms of its internal structure, the dataset comprises 568,471 rows. Initially, 68 attributes were extracted from the four central SQLite databases, specifically the places.sqlite, cookies.sqlite, formhistory.sqlite, and favicons.sqlite.

The database places.sqlite holds the main browsing history. Information regarding the times and URLs visited, as well as the number of times URLs were visited and the number of files downloaded, was extracted. The database also stores information on how the URL was accessed, such as whether it was typed, accessed from a bookmark, or reloaded. There are 10 visit types, which were converted to 10 binary attributes.

From the cookies.sqlite database, attributes regarding cookie names and values, their expiration dates, and last access times, along with the domain names of websites to which the cookies apply and the specific path on the website, were extracted. After aggregation, we added cookie count, mean, and maximum values for path length, value, and name length, among other metrics.

The favicons.sqlite file holds information about which icon to display for each website. It contains mostly forensically irrelevant data, such as size and color. However, we decided to extract the presence of favicons, as their presence without actual visits to that site can indicate potential use of anti-forensic techniques.

From formhistory.sqlite, we extracted attributes regarding the first and last use of the saved form data (names, emails, phone numbers, etc.), including their name and value, as well as the source. We prepared a blacklist of potentially suspicious words, such as ’hack’, ’curl’, and ’<script>’, to mark the saved value as blacklisted.

The evaluation of used machine learning methods is contingent upon the availability of a labeled dataset. Due to the absence of a suitable dataset, the available dataset was modified by inserting synthetic anomalies, crafted to reflect techniques that can be identified in the browser data. This facilitates the evaluation of the algorithms’ efficacy in detecting synthetic anomalies. The selection of modified rows is conducted at random on binary data. A separate dataset containing various anomalies is created for drive-by compromise (16 new anomalies), phishing (11 additional anomalies), data exfiltration (5 anomalies), detection evasion (12 anomalies), and the download of executables (10 anomalies). We selected these techniques based on the MITRE ATT&CK framework. We looked up techniques that leverage web browsers and selected those with the potential to manifest in our data. For each attack technique, we selected a set of binary attributes that reflect key behavioral indicators. We then generated all possible combinations of 0s and 1s for these attributes, representing different manifestations of the attack. However, not all combinations were plausible in practice. To ensure realism, we manually filtered out combinations that were logically inconsistent or semantically invalid. The remaining valid combinations, including those that already existed in the original dataset, were marked as anomalous.

3.2. Method Description

The available Mozilla Firefox browser data from SQLite databases is combined into a single dataframe through a series of steps. Initially, the tables within each database are joined. Then, the joint data from the databases is merged. In that phase, there is typically no attribute by which the data can be straightforwardly merged—for instance, the places.sqlite and formhistory.sqlite databases exhibit no shared attributes. The form history, however, includes the *firstUsed* and *lastUsed* attributes, which indicate the temporal context of utilization. These timestamps are then correlated with the *visit_date* timestamp. The process of adding data from the cookies.sqlite database requires identifying a match between the cookie domain and the URL of the visited page. It is important to note that a single page may have multiple cookies stored.

Following this, the data is aggregated by time of website visit into 30-minute time windows. We selected this time window based on Table 1, where the forensic analyst can see how the choice of time window affects the number of aggregated data (data points) and, ultimately, the resulting number of anomalies. By selecting the contamination value (ranging from 0.001 to 0.1), the respective percentage of the data will be marked as anomalous. The selection of aggregation functions varies according to the nature of each attribute. We use different aggregation functions on different attributes. For example, for the URL, an entropy is calculated and the maximum value is selected. For the visit ID's in each timeframe a length of the list of ID's and set of ID's is used to get new attributes, capturing number of total visited pages and distinct visited pages. Then, the attributes of the dataframe are converted to categorical and binary attributes. The conversion to binary data is based on the computed quartiles, which are used to define unique value ranges for each attribute. This approach allows the script to adapt the attributes to the specific characteristics of the data, providing a more accurate representation of user behavior. The number of final binary attributes depends on the range of values for each attribute. In this case, this resulted in 165 binary attributes.

Table 1
Data Points and Contamination Counts by Time Delta

Time Window	Data Points	0.001	0.002	0.005	0.01	0.02	0.05	0.1
15min	8766	9	18	44	88	176	439	877
30min	6230	7	13	32	63	125	312	623
1H	4204	5	9	22	43	85	211	421
2H	2680	3	6	14	27	54	134	268
6H	1397	2	3	7	14	28	70	140
1D	506	1	2	3	6	11	26	51

The subsequent step in the process involves implementing multiple anomaly detection methods. We can divide these methods into four categories [21]:

- Linear models for outlier detection;
- Proximity-based outlier detection models;
- Probabilistic models for outlier detection data;
- Outlier ensembles and combination frameworks.

Linear models for outlier detection - e.g., Principal Component Analysis (PCA), One-Class Support Vector Machines using Stochastic Gradient Descent (SGDOCSVM). **PCA** is a matrix decomposition method. It extracts important information and represents it as principal components [22]. Anomalies are identified as points that lie significantly far from the principal components in the transformed space. **SGDOCSVM** is a variant of OCSVM that optimizes hyperparameters using stochastic gradient descent [23].

Proximity-based outlier detection models - e.g., Local Outlier Factor (LOF). **LOF** is a method that identifies points appearing to be anomalous by using nearest neighbor search. LOF differs in that it identifies points that are outliers relative to a local cluster of points, rather than concerning the entire dataset [24].

Probabilistic models for outlier detection - e.g., Empirical-Cumulative-distribution-based Outlier Detection (ECOD), and Copula Based Outlier Detection (COPOD). **ECOD** uses empirical cumulative distribution functions to detect anomalies. This approach is based on the idea that anomalies will

Table 2
Comparison of eight unsupervised methods.

Method	Parameters	Time complexity
COPOD	contamination	$O(n.d)$
ECOD	contamination	$O(n.d)$
IForest	contamination, no_estimators	$O(n)$
iNNE	contamination, no_estimators	$O(n)$
LODA	contamination, n_bins	$O(n \cdot k \cdot d^{-1/2})$
LOF	contamination, no_neighbors, metric	$O(n^2)$
SGDOCSVM	nu	$O(d.n.T)$
PCA	contamination	$O(d^2.n + d^3)$

exhibit significantly different distributional characteristics compared to normal data [25]. On the other hand, **COPOD** utilizes the mathematical concept of copulas to model the multivariate distribution of the data, and identifies points with low probability in this distribution as anomalies [26].

Outlier ensembles and combination frameworks - e.g., Isolation Forest, Isolation-based Anomaly Detection Using Nearest-Neighbor Ensembles (iNNE), and Lightweight Online Detector of Anomalies (LODA). **Isolation forest** is a method that isolates observations by randomly selecting an element and then selecting a split value between the maximum and minimum value of the selected feature (Liu, Ting and Zhou, 2008). **iNNE** detects anomalies by combining isolation principles with nearest-neighbor ensembles to improve detection accuracy and robustness [27]. **Loda** can operate on incomplete data, such as during sensor outages, and simultaneously identify the attributes in which the anomaly deviates the most [28].

We have utilized two Python libraries for implementing the methods: scikit-learn and pyod. Scikit-learn [29] is an open-source library used for data analysis and machine learning. It encompasses various methods for classification, regression, and clustering. Pyod [30] is a library for detecting outlying objects in multivariate data. It includes more than 40 detection algorithms, such as LOF and ECOD.

Table 2 presents a comparative analysis of various methods, where n denotes the number of data points, d represents the number of dimensions, T is the number of epochs, and k are the random sparse projections. The time complexities of all the methods are also presented, and it is anticipated that the LOF and PCA methods will have the longest execution time. We are also examining the effect of parameter contamination across all methods under consideration. Additionally, for the Isolation Forest (IForest) method, we are exploring the impact of the number of estimators. For the Local Outlier Factor (LOF) approach, we are investigating the effect of the number of neighbors and the chosen metric.

For each of these methods, we experiment with various combinations of input attributes, depending on the method's specific requirements. The contamination value, however, is left for the forensic analyst to decide, regarding how many outliers can be reviewed manually.

3.3. Evaluation Metrics

To evaluate the performance of anomaly detection models, we used several basic metrics, such as precision (1) and recall (2).

$$\texttt{Precision} = \frac{TP}{TP + FP} \tag{1}$$

$$\texttt{Recall} = \frac{TP}{TP + FN} \tag{2}$$

Given the fact that this is an unbalanced dataset, we do not use the accuracy and F1-score metrics. Accuracy is defined as the proportion of correct predictions made. In the context of an imbalanced

Table 3
Performance comparison of anomaly detection methods.

Method	Precision	Recall	F-2 Score	Balanced Accuracy
COPOD	0.050000	0.623430	0.170906	0.787803
ECOD	0.050108	0.609323	0.171441	0.780830
IFOREST	0.013462	0.140447	0.045254	0.545423
INNE	0.000000	0.000000	0.000000	0.500000
LODA	0.015413	0.094489	0.042603	0.522726
LOF	0.009715	0.137344	0.034755	0.546402
ONECLASS	0.019991	0.060375	0.037608	0.520526
PCA	0.000000	0.000000	0.000000	0.500000

dataset, a model has the potential to attain a high degree of accuracy by consistently predicting the majority class. The F1-score penalizes both low precision and low recall, which does not reflect the priority of identifying the anomaly.

To identify suitable models, we supplemented the basic metrics with F-2 score (3), which prioritizes recall over precision, and balanced accuracy (4), which, in contrast to standard accuracy, assigns equal weight to both classes (regular and anomaly), even when one class is rare.

$$\texttt{F-2 score} = 5 \cdot \frac{\texttt{Precision} \cdot \texttt{Recall}}{4 \cdot \texttt{Precision} + \texttt{Recall}} \tag{3}$$

$$\texttt{Balanced Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{4}$$

In the domain of digital forensics, the primary objective of anomaly detection is to identify all instances of potentially malicious or unauthorized behavior for subsequent analysis. Within the specified context, recall (actual positive rate) emerges as the paramount performance metric. While false positives—indicated by lower precision—can be mitigated through manual validation by forensic analysts, false negatives represent a failure to surface relevant signals altogether.

4. Results and Discussion

A summary table 3 has been created by performing the mean operation on the results from all 5 dataset variations. It shows evaluation metrics for precision, recall (the most important), F-2 score with a focus on data we labeled as anomalous, and balanced accuracy. The results are not ordered by any metric.

All evaluation metrics indicate that both representatives of probabilistic models for outlier detection, namely `COPOD` and `ECOD`, achieve the best results. Within the performed analysis, they reach similar values, with `COPOD` performing slightly better in terms of *recall* and *balanced accuracy*. When focusing on *recall*, the third-best method is `iForest` (representing outlier ensembles and combination frameworks), closely followed by `LOF` (representing proximity-based outlier detection). The methods `LODA` and `OneClass` show overall unsatisfactory results.

As previously stated, both `COPOD` and `ECOD` are probabilistic methods for outlier detection, whereas `LOF` and `OneClass` rely on distance-based computations, which may be less suitable for binary data. The method `LODA` is also more appropriate for continuous data. The advantage of these models is also their linear time complexity ($O(n \cdot d)$), where n denotes the number of data points and d represents the number of dimensions. This allows for an increase in the number of attributes as well as the ability to handle a larger volume of data points (in our case, time windows). The third-ranking method (according to multiple evaluation metrics) is `iForest`. It may be encountering difficulties due to the imbalanced dataset, which contains only a limited number of anomalies. Since it also has a linear time complexity, it is feasible to consider increasing the number of attributes and adding additional outliers.

Figure 1 visualizes the *recall* values for the tested machine learning methods, while differentiating the techniques simulated in the data (drive-by compromise - labeled as *driveby*, downloading executables -

execution, detection evasion - *evasion*, phishing for information - *phishing*, and data exfiltration - labeled as *exfiltration*). Furthermore, it illustrates the variation in *recall* values in cases where the method had multiple configurations.

It is evident that the methods COPOD and ECOD are the most effective. Despite a significant drop in performance on the *defense evasion* dataset, they still outperform other machine learning methods. The methods INNE and PCA failed to detect any anomalies in our data, and modifying the input data would be necessary to obtain any results. The method OneClass shows relatively low *recall* in all variants. The methods LOF, iForest, and LODA had multiple settings, with some of them showing higher *recall* values — particularly LODA in the *detection evasion* variant — but the average performance remains low.

The *detection evasion* dataset and its anomalies are the most difficult to detect — they yield the lowest *recall* values for 3 out of 6 methods (excluding INNE and PCA). To improve detection, more attributes that reflect evasion behavior during detection should be included. The *drive-by compromise* dataset achieves the best *recall* for 4 out of 6 methods.

A critical component of enhancing the effectiveness of anomaly detection in digital forensic investigations is the optimization of the time window size, the *contamination* value, and the workload assigned to the analyst. Increasing the number of time windows enables finer-grained resolution of events but also imposes greater demands on manual validation. Similarly, setting an appropriate *contamination* value directly influences the sensitivity of the detection models. A lower value results in fewer flagged anomalies, thereby focusing the digital forensic analyst's attention, but risks missing events. A higher value increases *recall*, but at the cost of requiring more manual validation. The analyst must be aware of this trade-off and select a *contamination* value appropriate to the specific situation.

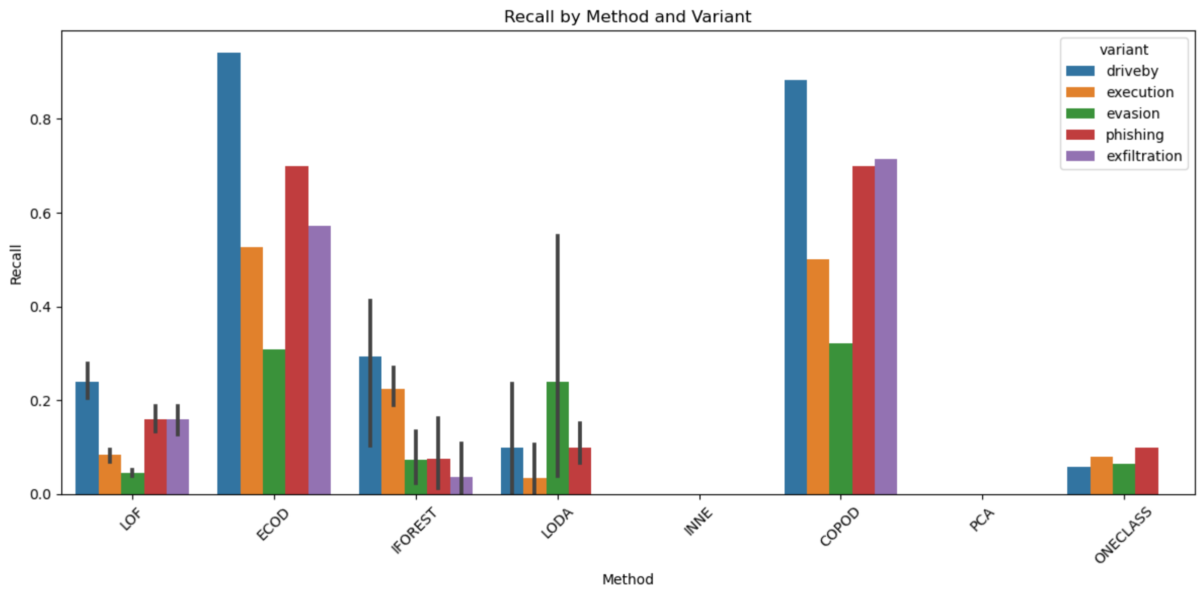

Figure 1: Recall by Method and Technique Variant

5. Conclusions

This paper presents a systematic approach to detecting anomalies in user activity behavior in web browsers by evaluating several unsupervised machine learning techniques. It focuses on four SQLite databases of the Mozilla Firefox browser, from which attributes were extracted, merged into a single dataframe, then aggregated and binarized. Synthetic anomalies were inserted into the binary data for evaluation purposes.

The paper demonstrated that the time window is an appropriate method for aggregating digital traces when analyzing user activities in web browsers. It is crucial to select an appropriate time frame, which depends on the amount of digital evidence and the number of selected time windows.

Furthermore, the paper analyzed anomaly detection methods and identified those that are suitable for detecting time windows representing digital evidence relevant to forensic investigation. The research showed that COPOD and ECOD methods are the most efficacious. It is essential to emphasize that the primary evaluation metric is recall, which measures the ability to capture true positive cases, as the objective is to identify as many true anomalies as possible. A higher number of false positives is not considered problematic, as a forensic analyst will manually review the anomalies.

In the future, this research should be extended to include other web browsers, focusing not only on standard browsing modes but also on private or incognito modes. In this context, it would be valuable to analyze browsers such as Tor and Brave. Additionally, this research could be expanded to include artifacts stored in volatile memory, as well as other artifacts saved by the Windows operating system (e.g., prefetch files and the MFT table).

Acknowledgments

This research was carried out within the project" Automatization of Digital Forensics and Incident Response (ADFIR)" (project code 09-I05-03-V02-00079), funded under the Recovery and Resilience Plan of the Slovak Republic K9 scheme:" Effective management and support of funding for science, research and innovation" approved by the Council of the European Union. The project, implemented at Pavol Jozef Šafárik University in Košice in collaboration with IstroSec s.r.o. and the European Information Society Institute, aims to develop an automated framework for the collection, normalization, and evaluation of digital traces, while ensuring their integrity and legal admissibility, to empower cybersecurity teams in responding to incidents and reducing the impact of cyber-attacks.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] D. Dissanayake, S. Rajakaruna, D. Ranasinghe, A. Wijesooriya, A. Jayakody, S. Rajapaksha, Platform independent browser forensic tool for advanced analysis of artifacts and case management, in: 2021 3rd International Conference on Advancements in Computing (ICAC), IEEE, 2021, pp. 383–388.

[2] H. Kim, I. Kim, K. Kim, AIBFT: Artificial intelligence browser forensic toolkit, Forensic Science International: Digital Investigation 36 (2021) 301091.

[3] P. Reedy, Interpol review of digital evidence for 2019–2022, Forensic Science International: Synergy 6 (2023) 100313.

[4] H. Studiawan, Y. Y. Rahadian, A. Riadi, Anomaly detection in log data using deep autoencoder, International Journal of Intelligent Engineering and Systems 14 (2021) 185–196.

[5] F. Shahzad, et al., Cyber anomaly detection using machine learning: A comprehensive review, ACM Computing Surveys (2022).

[6] F. Oladipo, et al., A state-of-the-art survey of digital forensic investigation frameworks, Forensic Science International: Digital Investigation (2020).

[7] R. Johnson, D. Wichern, Applied multivariate statistical analysis, Prentice Hall, 2002.

[8] A. Marková, et al., Detection of relevant digital evidence in Windows OS, in: Proceedings of the International Conference on Digital Forensics, 2022.

[9] P. Sokol, et al., Analysis of digital traces in NTFS: Relationship discovery and relevance for forensic investigation, in: Proceedings of the International Conference on Cybersecurity and Forensics, 2022.

[10] N. N. Joshi, S. L. Bajeja, Enhanced web browser forensics: Innovative methodologies for evidence collection and analysis, in: International Conference on Advancements in Smart Computing and Information Security, Springer Nature Switzerland, Cham, 2024, pp. 139–164.

[11] A. Raza, M. Hussain, H. Tahir, M. Zeeshan, M. A. Raja, K. H. Jung, Forensic analysis of web browsers lifecycle: a case study, Journal of Information Security and Applications 85 (2024) 103839.

[12] M. Hariharan, A. Thakar, P. Sharma, Forensic analysis of private mode browsing artifacts in portable web browsers using memory forensics, in: 2022 International Conference on Computing, Communication, Security and Intelligent Systems (IC3SIS), IEEE, 2022, pp. 1–5.

[13] R. R. Chand, N. A. Sharma, M. A. Kabir, Advancing web browser forensics: Critical evaluation of emerging tools and techniques, SN Computer Science 6 (2025) 355.

[14] R. Nelson, A. Shukla, C. Smith, Web browser forensics in Google Chrome, Mozilla Firefox, and the Tor Browser Bundle, in: Digital forensic education: An experiential learning approach, Springer International Publishing, Cham, 2019, pp. 219–241.

[15] M. S. Javed, S. M. Sajjad, D. Mehmood, K. Mansoor, Z. Iqbal, M. Kazim, Z. Muhammad, Analyzing Tor browser artifacts for enhanced web forensics, anonymity, cybersecurity, and privacy in Windows-Based systems, Information 15 (2024) 495.

[16] H. Studiawan, Y. Y. Rahadian, A. Riadi, Graph-based anomaly detection on forensic log data, in: 2017 3rd International Conference on Science in Information Technology (ICSITech), IEEE, 2017, pp. 307–312.

[17] H. Studiawan, A. Riadi, Y. Y. Rahadian, Performance evaluation of anomaly detection methods on log data, TELKOMNIKA 18 (2020) 1489–1498.

[18] J. Yuan, C. Hu, J. Pan, C. Zhao, Time-series anomaly detection for user behavior using deep learning, Journal of Information Security and Applications 58 (2021) 102781.

[19] M. Du, F. Li, G. Zheng, V. Srikumar, Automated anomaly detection for system logs with deep learning, Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (2020) 1285–1298.

[20] S. S. Chouhan, A. Dubey, R. Gupta, S. Jain, Ensemble model for anomaly detection in Windows registry for digital forensics, Forensic Science International: Digital Investigation 36 (2021) 301088.

[21] C. C. Aggarwal, An introduction to outlier analysis, in: Outlier analysis, Springer, 2017, pp. 1–34. doi:10.1007/978-3-319-47578-3_1.

[22] H. Abdi, L. J. Williams, Principal component analysis, Wiley Interdisciplinary Reviews: Computational Statistics 2 (2010) 433–459. URL: https://onlinelibrary.wiley.com/doi/full/10.1002/wics.101https://onlinelibrary.wiley.com/doi/abs/10.1002/wics.101https://wires.onlinelibrary.wiley.com/doi/10.1002/wics.101. doi:10.1002/WICS.101.

[23] G. Mutlu, C. Aci, SVM-SMO-SGD: A hybrid-parallel support vector machine algorithm using sequential minimal optimization with stochastic gradient descent, Parallel Computing 113 (2022) 102955. doi:10.1016/j.parco.2022.102955.

[24] H. Belyadi, A. Haghighat, Chapter 4 - Unsupervised machine learning: clustering algorithms, Gulf Professional Publishing, 2021, pp. 125–168. URL: https://www.sciencedirect.com/science/article/pii/B9780128219294000020. doi:https://doi.org/10.1016/B978-0-12-821929-4.00002-0.

[25] Z. Li, Y. Zhao, X. Hu, N. Botta, C. Ionescu, G. H. Chen, ECOD: Unsupervised outlier detection using empirical cumulative distribution functions, IEEE Transactions on Knowledge and Data Engineering 35 (2023) 12181–12193. doi:10.1109/TKDE.2022.3159580.

[26] Z. Li, Y. Zhao, N. Botta, C. Ionescu, X. Hu, COPOD: Copula-based outlier detection, Proceedings - IEEE International Conference on Data Mining, ICDM 2020-November (2020) 1118–1123. doi:10.1109/ICDM50108.2020.00135.

[27] T. R. Bandaragoda, K. M. Ting, D. Albrecht, F. T. Liu, Y. Zhu, J. R. Wells, Isolation-based anomaly detection using nearest-neighbor ensembles, Computational Intelligence 34 (2018) 968–998. doi:10.1111/coin.12168.

[28] T. Pevný, LODA: Lightweight On-line Detector of Anomalies, Machine Learning 102 (2016) 275–304. doi:10.1007/s10994-015-5521-0.

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer,

R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[30] Y. Zhao, Z. Nasrullah, Z. Li, PyOD: A python toolbox for scalable outlier detection, Journal of Machine Learning Research 20 (2019) 1–7. URL: http://jmlr.org/papers/v20/19-011.html.