LogGhostbuster: A Hybrid Machine Learning System for Bot Detection and Download Pattern Classification in Scientific Repository Logs

Yasset Perez-Riverol[1,*]
[1]European Molecular Biology Laboratory, European Bioinformatics Institute (EMBL-EBI), Wellcome Genome Campus, Hinxton, Cambridge CB10 1SD, UK
[*]Corresponding author: yperez@ebi.ac.uk

Abstract

Motivation: Scientific data repositories face increasing challenges from automated bot traffic that inflates download statistics, affects resource allocation decisions, and obscures genuine usage patterns. Distinguishing between legitimate automated access (mirrors, CI/CD pipelines) and malicious bots (scrapers, crawlers) requires sophisticated analysis of behavioral patterns that simple rule-based systems cannot capture.

Results: We present LogGhostbuster, a hybrid machine learning system that combines unsupervised anomaly detection with hierarchical classification to identify bot behavior in download logs. The system implements a three-level taxonomy distinguishing organic (human) from automated behavior, and further classifying automated access as either malicious bots or legitimate automation. Using Isolation Forest for anomaly detection and an optional Transformer-based deep architecture for temporal pattern recognition, LogGhostbuster achieves robust classification with configurable rules and provider-agnostic design. Applied to the PRIDE database at EMBL-EBI, the system identified 20,903 bot locations (29.4% of 71,133 analyzed) responsible for 113.1 million downloads (71.0% of total), while correctly distinguishing 667 legitimate automation sources including institutional mirrors.

Availability: LogGhostbuster is freely available at `https://github.com/bigbio/logghostbuster` under the Apache 2.0 license.

Keywords: bot detection, machine learning, anomaly detection, download statistics, scientific repositories

1 Introduction

Scientific data repositories have become essential infrastructure for research, hosting petabytes of data accessed by millions of users worldwide [Perez-Riverol et al., 2019]. Repositories such as the PRIDE database [Perez-Riverol et al., 2022], the European Nucleotide Archive (ENA) [Leinonen et al., 2011], and UniProt [The UniProt Consortium, 2023] provide critical resources for the scientific community. Understanding how these resources are used is essential for resource allocation, impact assessment, and service improvement.

However, download statistics from scientific repositories are increasingly contaminated by automated bot traffic. Studies have shown that bot traffic can account for 30-70% of web requests [Imperva, 2023], with scientific repositories being particularly attractive targets due to their open access policies and valuable data content. This contamination creates several problems:

1. **Inflated metrics**: Download counts used for impact assessment become unreliable when dominated by automated access.

2. **Resource misallocation**: Infrastructure decisions based on apparent demand may not reflect actual user needs.

3. **Obscured patterns**: Genuine usage trends become difficult to identify amid bot noise.

4. **Service degradation**: High-volume automated access can degrade service quality for human users.

Distinguishing bot traffic from human access is challenging because not all automated access is problematic. Legitimate automation includes institutional mirrors that improve global access, continuous integration pipelines that test software against repository data, and data aggregation services that provide valuable meta-analyses. A successful bot detection system must therefore implement a nuanced classification that distinguishes malicious from legitimate automation.

Existing approaches to bot detection typically fall into two categories: signature-based methods that identify known bot user agents and IP ranges, and behavioral analysis that identifies bot-like access patterns [Jonker et al., 2019]. Signature-based methods suffer from easy circumvention through user agent spoofing, while simple behavioral rules miss sophisticated bots that mimic human patterns. Machine learning approaches have shown promise [Rovetta et al., 2020], but most focus on web security rather than scientific repository contexts where different patterns prevail.

In this paper, we present LogGhostbuster, a hybrid machine learning system specifically designed for bot detection in scientific repository download logs. Our contributions include:

- A **hierarchical classification taxonomy** that distinguishes organic (human-like) from automated behavior, and further classifies automated access as malicious bots or legitimate automation.

- A **hybrid detection architecture** combining Isolation Forest anomaly detection with rule-based and optional deep learning classification.

- A **provider-agnostic design** that supports different log formats through configurable schema mappings and extensible feature extraction.

- **Comprehensive evaluation** on real-world data from the PRIDE proteomics database, demonstrating the system's effectiveness in a production environment.

2 Background

2.1 Bot Detection Challenges

Bot detection in web traffic has been extensively studied in the security community [Iliou et al., 2021]. Traditional approaches include:

Signature-based detection relies on matching known bot signatures such as user agent strings, IP address ranges, or access patterns. While effective against unsophisticated bots, these methods fail against bots that spoof legitimate user agents or use residential proxies [Jonker et al., 2019].

Behavioral analysis examines access patterns to identify non-human behavior. Metrics such as request rate, session duration, and navigation patterns can distinguish bots from humans [Rovetta et al., 2020]. However, sophisticated bots increasingly mimic human behavior, reducing the effectiveness of simple rules.

Machine learning approaches have shown promise in capturing complex patterns that simple rules miss. Both supervised [Cabri et al., 2021] and unsupervised [Habibi Lashkari et al., 2020] methods have been applied, with ensemble approaches often achieving the best results.

2.2 Scientific Repository Context

Bot detection in scientific repositories presents unique challenges:

1. **Legitimate automation**: Unlike e-commerce sites where automation is generally unwanted, scientific repositories benefit from mirrors, aggregators, and automated pipelines.
2. **Download-centric patterns**: Scientific repositories primarily serve file downloads rather than page views, requiring different feature engineering.
3. **Long-term patterns**: Bot campaigns against repositories may span months or years, requiring temporal analysis across extended periods.
4. **Geographic distribution**: Global scientific collaboration means legitimate access comes from diverse locations, making geographic filtering problematic.

2.3 Isolation Forest

Isolation Forest [Liu et al., 2008] is an unsupervised anomaly detection algorithm particularly suited to high-dimensional data. Unlike distance-based methods, Isolation Forest identifies anomalies by their susceptibility to isolation through random partitioning.

The algorithm constructs an ensemble of isolation trees, each built by recursively partitioning the data using random feature selections and split points. Anomalies, being rare and different, require fewer partitions to isolate and thus have shorter average path lengths in the trees.

For a data point x, the anomaly score is computed as:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{1}$$

where $E(h(x))$ is the expected path length of x across all trees, n is the sample size, and $c(n)$ is a normalization factor.

Isolation Forest has several advantages for our application:

- No assumption about the distribution of normal data
- Linear time complexity $\mathrm{O}(n \log n)$
- Effective in high-dimensional spaces
- Robust to irrelevant features

3 Methods

3.1 System Architecture

LogGhostbuster implements a multi-stage pipeline (Figure 1) comprising:

1. **Data Ingestion**: Memory-efficient loading and optional sampling of Parquet files via DuckDB.
2. **Feature Extraction**: Computation of 60+ behavioral features at the location (IP/geo) level.
3. **Anomaly Detection**: Isolation Forest identifies statistically unusual access patterns.
4. **Classification**: Hierarchical classification using rules or deep learning.
5. **Reporting**: Generation of annotated data and comprehensive reports.

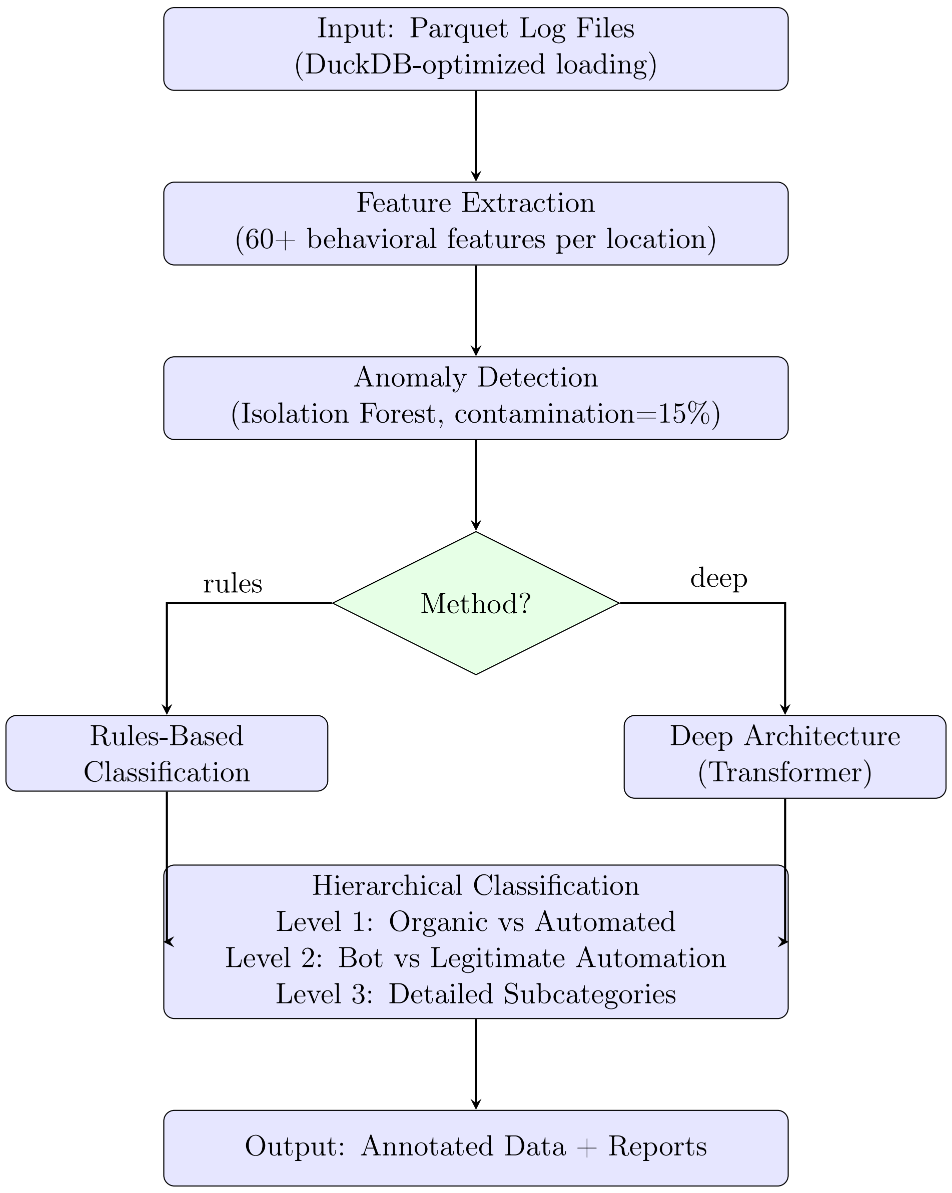
Input: Parquet Log Files
(DuckDB-optimized loading)
Feature Extraction
(60+ behavioral features per location)
Anomaly Detection
(Isolation Forest, contamination=15%)
Method?
rules
deep
Rules-Based Classification
Deep Architecture (Transformer)
Hierarchical Classification
Level 1: Organic vs Automated
Level 2: Bot vs Legitimate Automation
Level 3: Detailed Subcategories
Output: Annotated Data + Reports

Figure 1: LogGhostbuster system architecture showing the multi-stage pipeline from data ingestion through classification and reporting.

3.2 Feature Extraction

Features are computed at the *location* level, where a location represents a unique geographic origin (typically derived from IP geolocation). This aggregation reduces millions of individual download events to thousands of location profiles, making analysis tractable while preserving behavioral patterns.

3.2.1 Basic Activity Features

- `unique_users`: Number of distinct user identifiers from this location
- `downloads_per_user`: Ratio of total downloads to unique users
- `total_downloads`: Aggregate download count
- `projects_per_user`: Diversity of accessed resources

3.2.2 Temporal Features

Temporal patterns are critical for bot detection, as automated access often exhibits characteristic timing signatures:

- `hourly_entropy`: Shannon entropy of hourly download distribution. Bots often show low entropy (concentrated activity) while humans show higher entropy (distributed activity).
- `working_hours_ratio`: Fraction of downloads during business hours (9 AM - 6 PM local time). Human users typically show higher ratios.
- `night_activity_ratio`: Fraction of downloads during night hours. Elevated night activity suggests automated scheduling.
- `yearly_entropy`: Distribution of activity across years, detecting sudden spikes.
- `spike_ratio`: Ratio of latest year downloads to historical average.

3.2.3 Behavioral Features (Deep Method)

Advanced features capture nuanced behavioral patterns:

- `burst_pattern_score`: Detects concentrated download bursts within short time windows.
- `circadian_rhythm_deviation`: Measures deviation from typical human circadian patterns.
- `user_coordination_score`: Identifies synchronized activity across multiple user IDs, suggesting bot farms.
- `regularity_score`: Measures temporal regularity, with high values indicating scheduled automation.

3.2.4 Discriminative Features

Features designed to distinguish malicious bots from legitimate automation:

- `file_exploration_score`: Breadth of file access (bots often scrape systematically)
- `file_mirroring_score`: Pattern consistency with mirroring behavior
- `bot_farm_score`: User homogeneity suggesting coordinated fake accounts
- `geographic_stability`: IP diversity from this location
- `legitimate_automation_score`: Composite score for legitimate automation indicators

3.3 Hierarchical Classification Taxonomy

LogGhostbuster implements a three-level classification taxonomy (Figure 2):

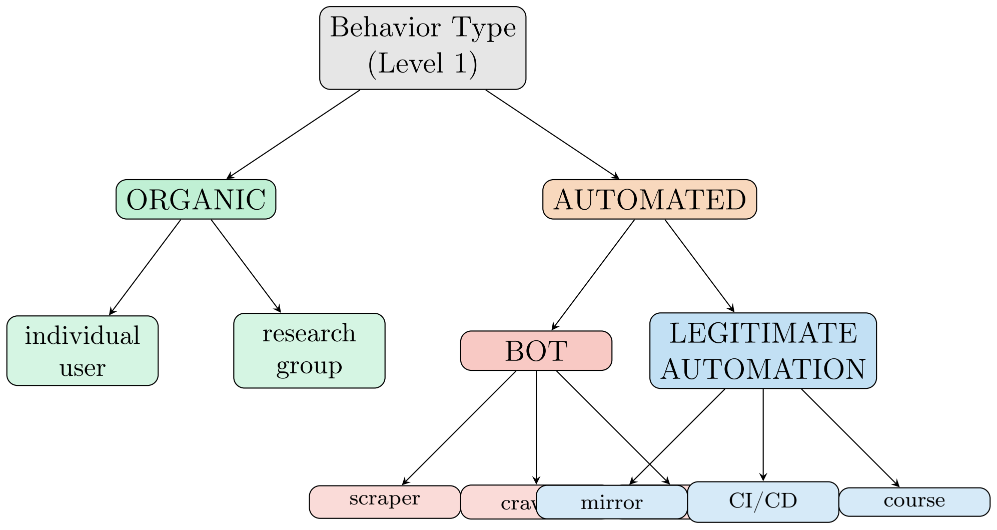

Figure 2: Three-level hierarchical classification taxonomy. Level 1 distinguishes organic (human-like) from automated behavior. Level 2 classifies automated behavior as malicious bot or legitimate automation. Level 3 provides detailed subcategories.

Table **??** presents the complete hierarchical classification rules organized by level, class, and the specific feature thresholds used for classification.

Table 1: Hierarchical classification rules. Level 1 distinguishes behavior type, Level 2 classifies automation intent, and Level 3 provides detailed subcategories.

Level	Class	Rules	Description
Level 1	ORGANIC	`working_hours_ratio` ≥ 0.4 `regularity_score` ≤ 0.6 `interval_cv` ≥ 0.7 `unique_users` < 50	Human-like download patterns with activity during business hours, irregular timing, and limited user counts
	AUTOMATED	`regularity_score` ≥ 0.7 `night_activity_ratio` ≥ 0.35 `user_coordination_score` ≥ 0.6 `downloads_per_user` ≥ 500	Programmatic access with high temporal regularity, non-working hours activity, or coordinated multi-user patterns
Level 2	BOT	`unique_users` ≥ 1000 `downloads_per_user` ≤ 50 `user_coordination_score` ≥ 0.7 `user_authenticity_score` ≤ 0.3	Malicious automation: bot farms with many fake users, coordinated activity, and suspicious behavioral patterns
	LEGITIMATE_ AUTOMATION	`downloads_per_user` ≥ 500 `unique_users` ≤ 100 `geographic_stability` ≥ 0.7 `regularity_score` ≥ 0.8	Benign automation: mirrors, CI/CD pipelines with few users, high per-user activity, and geographic stability
Level 3	individual_user	`unique_users` ≤ 5, `DL/user` ≤ 20	Single researchers or casual users
	research_group	`users` 5–50, `working_hours` ≥ 0.5	Small academic teams
	scraper_bot	`users` ≥ 5000, `DL/user` ≤ 20	High-frequency automated scrapers
	crawler_bot	High `file_exploration_score`	Systematic crawlers
	coordinated_bot	`coordination_score` ≥ 0.8	Bot farms with synchronized activity
	mirror	`DL/user` ≥ 500, `users` ≤ 100	Institutional mirrors
	ci_cd_pipeline	`users` ≤ 10, `regularity` $\geq$ 0.8	Automated testing/builds
	institutional_hub	`downloads` $\geq$ 100K	Research infrastructure hubs
	course_workshop	`users` 50–500, `DL/user` 5–30	Educational events

3.4 Classification Methods

LogGhostbuster provides two classification methods:

3.4.1 Rule-Based Classification

The rule-based method uses YAML-configurable pattern matching. For each classification level, patterns are defined as conjunctions of feature constraints:

Listing 1: Example rule pattern

```
behavior_type:
  automated:
    patterns:
      - id: high_regularity
        regularity_score:
          min: 0.7
      - id: many_users_pattern
        unique_users:
          min: 1000
        downloads_per_user:
          max: 50
```

Patterns are evaluated in order, with the first matching pattern determining classification. This approach provides:

- Interpretability: Each classification has a clear rule chain
- Configurability: Rules can be adjusted without code changes
- Speed: Pattern matching is computationally efficient

3.4.2 Deep Architecture Classification

The deep method augments rule-based classification with temporal pattern learning:

Time-series Feature Extraction: Download patterns are aggregated into fixed-length sequences (default: 12 monthly windows) capturing temporal evolution.

Transformer Encoder: A Transformer architecture processes time-series features:

$$\mathbf{h} = \text{Transformer}(\mathbf{X}_{ts}) \oplus \mathbf{x}_{fixed} \tag{2}$$

where $\mathbf{X}_{ts} \in \mathbb{R}^{T \times d}$ represents T time windows with d features, and $\mathbf{x}_{fixed}$ are static location features.

Multi-task Learning: The model jointly predicts:

- User category (bot, hub, independent, normal, other)
- Binary bot indicator (via auxiliary head)

Confidence-based Classification: Predictions with low confidence are flagged for human review:

$$\text{confidence} = 1 - H(p)/\log(K) \tag{3}$$

where $H(p)$ is the entropy of the softmax distribution and K is the number of classes.

3.5 Implementation

LogGhostbuster is implemented in Python 3.8+ with the following key dependencies:

- **DuckDB**: Memory-efficient Parquet processing with SQL interface
- **scikit-learn**: Isolation Forest implementation
- **PyTorch**: Deep learning components (optional)
- **pandas/numpy**: Data manipulation
- **matplotlib/seaborn**: Visualization

The system supports:

- **Sampling**: RESERVOIR sampling for large datasets
- **Memory management**: Configurable limits (default 16GB)
- **Parallel processing**: Multi-core utilization
- **Provider plugins**: Extensible log format support

4 Results

4.1 Dataset: PRIDE Database Logs

We evaluated LogGhostbuster on download logs from the PRIDE proteomics database [Perez-Riverol et al., 2022], one of the world's largest mass spectrometry data repositories. The dataset comprises:

- **Total downloads**: 159.3 million
- **Unique locations**: 71,133 (after geographic aggregation)
- **Time period**: Multi-year (exact dates anonymized)
- **Features**: 60+ computed per location

4.2 Classification Results

Table 2 summarizes the hierarchical classification results:

Table 2: Classification results on PRIDE database logs

Classification	Locations	%	Downloads (%)
Level 1: Behavior Type			
ORGANIC	49,563	69.7%	913K (0.6%)
AUTOMATED	21,570	30.3%	158.4M (99.4%)
Level 2: Automation Category			
BOT	20,903	96.9%*	113.1M (71.0%)
LEGITIMATE_AUTOMATION	667	3.1%*	45.3M (28.4%)
Level 3: Subcategories			
individual_user	29,303	41.2%	—
research_group	20,311	28.5%	—
generic_bot	12,375	17.4%	—
scraper_bot	8,477	11.9%	—
mirror	667	0.9%	—

*Percentage of automated locations

4.3 Key Findings

4.3.1 Bot Traffic Dominance

Despite representing only 29.4% of locations, bot traffic accounts for 71.0% of total downloads. This dramatic asymmetry highlights the importance of bot detection for accurate usage statistics.

4.3.2 Download Hub Identification

The system correctly identified 667 legitimate automation sources responsible for 45.3 million downloads (28.4%). These include:

- Institutional mirrors providing regional access
- CI/CD pipelines for bioinformatics software
- Data aggregation services

Distinguishing these from malicious bots prevents incorrect blocking of valuable services.

4.3.3 Human vs Automated Patterns

The Level 1 classification reveals a clear separation:

- **Organic locations** (69.7%): Many locations with minimal download activity, typical of individual researchers.
- **Automated locations** (30.3%): Fewer locations with massive download volumes.

4.4 Feature Importance

Analysis of feature importance reveals the most discriminative features for bot detection (Table 3):

Table 3: Top features for bot classification (correlation with bot label)

Feature	Correlation
unique_users	0.45
downloads_per_user	-0.38
user_coordination_score	0.32
night_activity_ratio	0.28
working_hours_ratio	-0.25
regularity_score	0.24
hourly_entropy	-0.21
burst_pattern_score	0.19

The inverse relationship between `unique_users` and `downloads_per_user` is the strongest signal: bots typically show many users with low individual activity (cycling fake accounts), while mirrors show few users with very high activity.

4.5 Performance Characteristics

Table 4: Computational performance

Component	Rules Method	Deep Method
Feature extraction	3.2 min	3.2 min
Anomaly detection	0.5 min	0.5 min
Classification	0.3 min	15.8 min
Report generation	0.8 min	0.8 min
Total	**4.8 min**	**20.3 min**

Tested on 71K locations, 16GB RAM, 8-core CPU

5 Discussion

5.1 Strengths

Hybrid approach: Combining unsupervised anomaly detection with supervised/rule-based classification provides robustness. Isolation Forest captures unusual patterns without labeled training data, while the classification layer provides interpretable categories.

Hierarchical taxonomy: The three-level classification provides both high-level insights (organic vs automated) and actionable detail (specific subcategories). This supports different use cases from quick triage to detailed investigation.

Provider agnosticism: The schema mapping and extensible feature extraction support different log formats without code changes. This enables deployment across diverse repositories.

Configurability: YAML-based rule configuration allows domain experts to tune thresholds without programming knowledge, adapting to evolving bot patterns.

5.2 Limitations

Ground truth uncertainty: Without definitive labels, evaluation relies on pattern consistency and expert review. Some edge cases may be misclassified.

Temporal lag: The system analyzes historical data and may not detect new bot campaigns in real-time. Integration with streaming analysis would address this.

Feature engineering dependency: Performance depends on the quality of engineered features. Novel bot behaviors not captured by current features may evade detection.

Computational cost: The deep method significantly increases processing time. For very large datasets or real-time needs, the rules method may be preferable.

6 Conclusion

LogGhostbuster provides an effective solution for bot detection in scientific repository logs. By combining Isolation Forest anomaly detection with hierarchical classification, the system accurately distinguishes malicious bots from legitimate automation while maintaining interpretability.

Applied to the PRIDE database, LogGhostbuster identified that 71% of downloads originated from automated bot traffic, while correctly preserving 667 legitimate automation sources. This finding has significant implications for download statistics reporting and resource allocation in scientific repositories.

The open-source release enables other repositories to benefit from bot detection, improving the accuracy of scientific usage metrics across the research data ecosystem.

Funding

This work was supported by EMBL core funding.

Acknowledgements

We thank the PRIDE team for providing access to download logs and feedback on classification results.

References

Alberto Cabri, Grażyna Suchacka, Stefano Rovetta, and Francesco Masulli. Detecting web bot activity on social networks: from social media to web environments. In *2021 IEEE International Conference on Web Services (ICWS)*, pages 622–627. IEEE, 2021. doi: 10.1109/ICWS53863.2021.00085.

Arash Habibi Lashkari, Andi Fitriah Abdul Kadir, Hugo Gonzalez, Kenneth F Mbah, and Ali A Ghorbani. Bot detection using User Agent-based fingerprinting. *Computers & Security*, 95: 101869, 2020. doi: 10.1016/j.cose.2020.101869.

Christos Iliou, Theodoros Kostoulas, Theodora Tsikrika, Vasileios Katos, Stefanos Vrochidis, and Ioannis Kompatsiaris. Towards a framework for detecting advanced web bots. *arXiv preprint arXiv:2101.11468*, 2021.

Imperva. Bad bot report 2023: The account takeover edition. Technical report, Imperva Inc., 2023. URL `https://www.imperva.com/resources/reports/2023-bad-bot-report/`. Annual analysis of automated bot traffic patterns across the internet.

Mattijs Jonker, Brett Stone-Gross, David Plonka, and Alistair Boehme. Fingerprinting tooling used for SSH dictionary attack. *Digital Investigation*, 31:S138–S146, 2019. doi: 10.1016/j.diin.2019.06.012.

Rasko Leinonen, Ruth Akhtar, Ewan Birney, Lawrence Bower, Ana Cerdeno-Tárraga, Yuan Cheng, Iain Cleland, Nadeem Faruque, Neil Goodgame, Richard Gibson, et al. The European Nucleotide Archive. *Nucleic Acids Research*, 39(suppl_1):D28–D31, 2011. doi: 10.1093/nar/gkq967.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008. doi: 10.1109/ICDM.2008.17.

Yasset Perez-Riverol, Attila Csordas, Jingwen Bai, Manuel Bernal-Llinares, Suresh Hewapathirana, Deepti J Kundu, Avinash Inuganti, Johannes Griss, Gerhard Mayer, Martin Eisenacher, et al. Making proteomics data accessible and reusable: current state of proteomics databases and repositories. *Proteomics*, 19(16):1800371, 2019. doi: 10.1002/pmic.201800371.

Yasset Perez-Riverol, Jingwen Bai, Chakradhar Bandla, David García-Seisdedos, Suresh Hewapathirana, Selvakumar Kamatchinathan, Deepti J Kundu, Ananth Prakash, Anika Frericks-Zipper, Martin Eisenacher, et al. The PRIDE database resources in 2022: a hub for mass spectrometry-based proteomics evidences. *Nucleic Acids Research*, 50(D1):D483–D490, 2022. doi: 10.1093/nar/gkab1038.

Stefano Rovetta, Grażyna Suchacka, and Francesco Masulli. Bot and gender identification of Twitter users based on writing style analysis. In *International Conference on Computational Collective Intelligence*, pages 459–470. Springer, 2020. doi: 10.1007/978-3-030-63119-2_38.

The UniProt Consortium. UniProt: the universal protein knowledgebase in 2023. *Nucleic Acids Research*, 51(D1):D483–D489, 2023. doi: 10.1093/nar/gkac1052.