Continual Learning for Real-Time Credential Stuffing Detection: A Neural Embedding Approach with Temporal Stress Testing

Daniel Will George
Daniel George Research
Los Angeles, California
daniel.george@stanford.edu

Abstract

Credential stuffing—in which adversaries replay large volumes of leaked username-and-password pairs against online services—continues to evade static, rule-based defenses. We present a machine-learning solution that replaces an inflexible rule-based system with a continual-learning pipeline built on neural-network embeddings and FAISS-indexed k-nearest-neighbor (k-NN) classification. Version 1.0 of the model achieves **95.63% accuracy**, **96.87% precision**, and **94.32% recall** on the training benchmark—a **14× improvement in F1 score** over the rule-based baseline (6.79%). Critically, because the embeddings store is updated continuously with labeled ground truth, the model trained on three hours of May 2024 data generalizes to March 2025 with virtually no performance loss (F1 $\geq$ 95%), and sustains mean accuracy of $\geq$99.4% across 51 held-out one-hour windows spanning weekdays, weekends, and four consecutive calendar months. These results demonstrate that continual learning offers a practical, cost-effective alternative to periodic full retraining for production fraud detection.

Keywords: credential stuffing, continual learning, neural-network embeddings, k-nearest neighbors, FAISS, fraud detection, temporal generalization.

1 Introduction

Credential stuffing is a large-scale attack in which adversaries obtain credential lists from prior data breaches and systematically test them across web services [11]. Even modest success rates—often below 1%—translate into millions of compromised accounts when attackers submit billions of login attempts [3].

The proposed system replaces a hand-crafted rule-based approach centered on a single threshold: more than 20 login attempts from a given IP address in a single day, where more than 80% of those attempts fail. While simple to implement and audit, this system exhibits three critical weaknesses. First, it is entirely static: any fraud pattern that falls below the threshold escapes detection. Second, its recall is catastrophically low—empirical evaluation on balanced data yields a recall of just **3.52%**, meaning the system misses the overwhelming majority of attacks. Third, it cannot adapt as adversaries diversify tactics—for example, by distributing attempts across many IP addresses ("low-and-slow" attacks) or rotating through residential proxies.

Machine learning offers a principled path forward, but conventional approaches introduce their own challenge: models trained on historical data grow stale as user behavior and attack patterns shift [1]. Periodic full retraining is expensive and introduces deployment latency.

We address both problems with a *continual-learning* architecture. Rather than retraining from scratch, the system continuously appends ground-truth-labeled embeddings to a FAISS similarity index [5]. New login events are classified by comparing their embedding to the k nearest stored neighbors; as the index grows richer, classification improves without touching the neural-network weights. This design yields a model that can be initialized on a few hours of data and deployed immediately, then refined in production by domain experts labeling events periodically.

This paper makes the following contributions:

1. A description of the end-to-end continual-learning pipeline for credential stuffing detection, including feature engineering, neural architecture, contrastive training, and FAISS inference.
2. A comparison against two baselines—the existing rule-based system and a random classifier—

demonstrating a 14× improvement in F1 score.

3. A rigorous *temporal stress test* across 51 held-out one-hour windows covering seven days of a week, four mid-week Tuesdays, four weekend Saturdays, and four first Mondays of consecutive months, confirming robustness to temporal distribution shift.

2 Related Work

Credential stuffing detection. Early defenses focused on IP reputation and rate limiting [13]. More recent work incorporates device fingerprinting and behavioral biometrics [10], but requires rich client-side telemetry that may not always be available. Our approach relies exclusively on server-side login-event features.

Continual and incremental learning. The catastrophic forgetting problem in neural networks—whereby new training erases previously learned patterns—is well documented [6]. Our system sidesteps this issue by keeping the neural-network weights frozen after initialization and encoding all new knowledge as labeled points in the FAISS store, which serves as an external memory [8].

Embedding-based similarity search. FAISS [5] provides sub-linear approximate nearest-neighbor search over billion-scale vector sets, making it suitable for production deployments where latency constraints are strict. Contrastive learning objectives such as CosineEmbeddingLoss [4] are widely used to produce well-separated embedding spaces that facilitate downstream k-NN classification.

Dimensionality reduction for fraud visualization. Both t-SNE [12] and UMAP [7] are standard tools for visualizing high-dimensional embedding spaces and for verifying that class separation is meaningful.

3 System Design

3.1 Labeling Criteria

A login event is labeled as credential stuffing if *any* of the following conditions holds for a given username or client IP address on a single calendar day:

1. More than 20 login attempts with a failure rate exceeding 80% (matching the legacy rule).
2. A change in (latitude, longitude) tuple within 30 seconds.
3. More than 5 distinct usernames attempted *and* more than 10 failed logins.

To compensate for class imbalance, the minority (fraud) class is upsampled uniformly at random while keeping groups of client IPs together, mirroring real-world traffic structure.

3.2 Feature Engineering

The feature space has 83 dimensions organized around three groups (Table 1). To avoid label leakage, raw categorical fields used in labeling (e.g., country of origin) are *not* used directly as features. Instead, frequency-distribution proxies are constructed—for instance, a binary flag indicating whether the subnet falls outside the top-k most commonly observed subnets.

Table 1: Feature group contributions to model predictive power (XGBoost split-based importance).

Feature Group	Contribution (%)
User Behavior Patterns	51.7
Network Activity Metrics	36.0
Geographic Location Signals	11.3
Other	1.1

The single most predictive feature is `distance_to_previous_login` (importance 0.517), reflecting temporal distance between successive logins for the same user or IP. Network features `subnet_is_not_top_k` (0.132) and `avg_velocity_is_not_0` (0.127) are the next most important, collectively meaning that the top four features account for over 85% of total model importance (Figure 1).

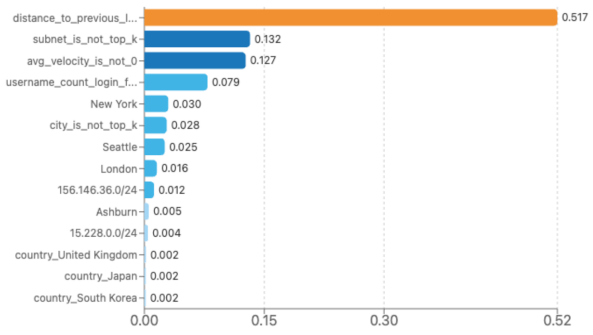

Figure 1: XGBoost feature importance scores. `distance_to_previous_login` dominates at 0.517; the top four features together exceed 85% of total importance.

3.3 Neural Network Architecture

The embedding network (v1.0) is a three-layer MLP that performs progressive dimensionality reduction:

$$\mathbb{R}^{83} \xrightarrow{L_1} \mathbb{R}^{256} \xrightarrow{L_2} \mathbb{R}^{128} \xrightarrow{L_3} \mathbb{R}^{64}.$$

Each of the first two blocks applies Linear $\rightarrow$ ReLU $\rightarrow$ BatchNorm (momentum $= 0.1$) $\rightarrow$ Dropout($p = 0.2$). The final layer is a bare linear projection producing 64-dimensional embeddings without forced normalization, preserving the natural scale of the latent space.

3.4 Contrastive Training

Training uses *CosineEmbeddingLoss* with an augmented spread-out regularization term:

$$\mathcal{L} = \mathcal{L}_{\text{cosine}} + \lambda \cdot \mathbb{E}[\text{pdist}(\mathbf{Z})]\,,$$

where $\lambda = 0.1$ and $\mathbf{Z}$ is the batch of output embeddings. The spread-out term prevents embedding collapse [14], ensuring that individual login events remain distinguishable in the FAISS index. Positive and negative pairs are drawn in balanced proportion via a `get_pairs` function. Convergence is achieved in approximately 30 epochs (Figure 2).

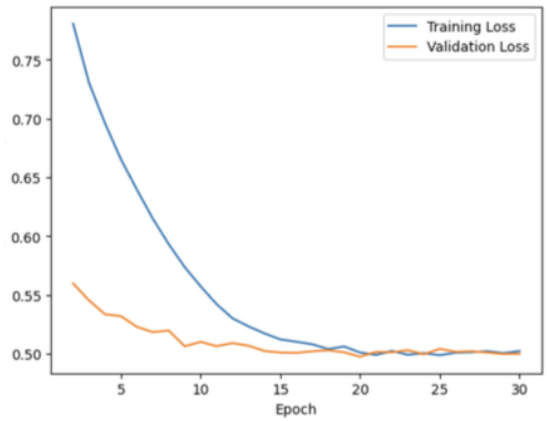

Figure 2: Training and validation loss curves for embedding model v1.0. Stable convergence is achieved at approximately 30 epochs.

3.5 Continual Learning via FAISS

After initialization, the neural-network weights are frozen. All ongoing learning takes place by appending ground-truth-labeled embeddings to a FAISS index (Figure 3). During inference, an incoming login event is converted to a 64-dimensional embedding, and the $k = 10$ nearest stored embeddings are retrieved. The fraud probability is:

$$\hat{p} = \frac{1}{k} \sum_{i=1}^{k} y_i,$$

where $y_i \in \{0, 1\}$ are the labels of the nearest neighbors. A prediction of credential stuffing is made when $\hat{p} \geq 0.5$. The index holds approximately 10 million recent login events and supports sub-millisecond batch queries via FAISS's `IndexFlatL2` (or `IndexIVFFlat` for larger deployments).

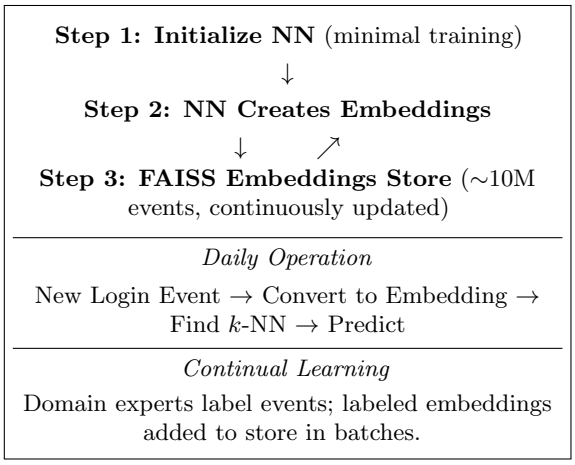

Figure 3: End-to-end system architecture. The neural network is initialized once; all subsequent learning occurs via FAISS index updates.

4 Experiments

4.1 Baseline Comparison (Model v1.0)

Three systems are compared on a balanced training dataset:

- **Rule-based (Baseline 1):** The existing threshold system.
- **Random (Baseline 2):** A fair-coin classifier.
- **ML v1.0:** The proposed embeddings model.

Table 2: Benchmark performance on the training dataset.

Metric	Rule-Based	Random	ML v1.0
Accuracy	51.75%	50.38%	**95.63%**
Precision	100.00%*	50.39%	**96.87%**
Recall	3.52%	50.49%	**94.32%**
F1 Score	6.79%	50.44%	**95.58%**
ROC AUC	51.76%	50.38%	**95.63%**

*Perfect precision arises from label leakage (same criterion used for labeling and evaluation).

The ML model achieves a **14× improvement in F1 score** relative to the rule-based system (95.58% vs. 6.79%). The near-perfect precision of the rule-based system is an artifact of label leakage: the labels themselves were partially derived from the same criterion, so the baseline trivially achieves 100% precision at the cost of capturing only 3.52% of actual attacks. The t-SNE visualization of the learned embedding space (Figure 4) shows clear spatial separation between fraud (orange) and benign (blue) events, validating that the model learns meaningful representations.

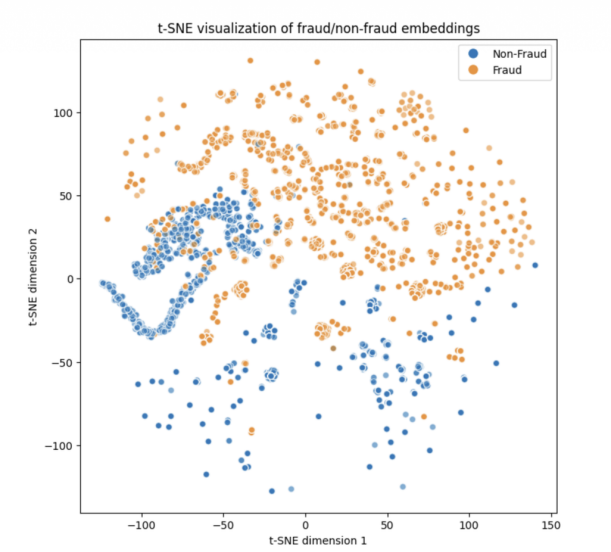

Figure 4: t-SNE visualization of the 64-dimensional embedding space after v1.0 training. Orange: credential stuffing; blue: benign logins. The clear spatial separation enables accurate k-NN classification.

4.2 Temporal Generalization

To validate the continual-learning claim, the neural-network weights and FAISS index trained on **May 2024** data (3-hour window) were evaluated against data from **March 2025**—a gap of approximately nine months.

Table 3: Model generalization across a nine-month temporal gap.

Metric	May 2024	March 2025
Accuracy	99.34%	99.76%
Precision	92.82%	95.96%
Recall	98.22%	94.65%
F1 Score	95.44%	95.30%
ROC AUC	98.83%	97.27%

Performance is essentially unchanged across the nine-month gap (Figure 5), confirming the efficacy of continual learning. The F1 score declines by only 0.14 percentage points.

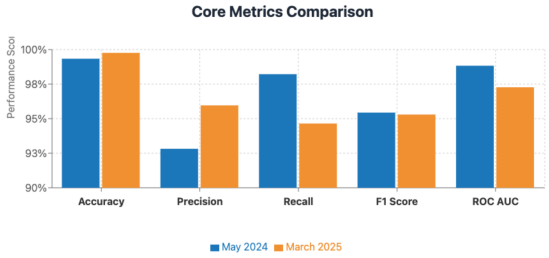

Figure 5: Core metrics comparison between May 2024 training and March 2025 temporal test. Blue: May 2024; orange: March 2025. Performance is effectively stable across the nine-month gap.

4.3 Temporal Stress Testing (51 Held-Out Windows)

Using the May 2024-trained model, we ran inference on 51 non-overlapping one-hour windows, each downsampled to 100,000 events via stratified sampling. After each batch, the model's inferences were added (with true labels) to the FAISS store, simulating the production continual-learning loop. The windows were organized into four test sets.

4.3.1 Seven-Day Weekly Sweep

Sunday 22 Feb through Saturday 28 Feb 2025, six time slots per day (7 am, 9 am, 1 pm, 4 pm, 6 pm, 9 pm).

Table 4: Performance across 7 days × 6 hours (42 windows).

Metric	Mean	Max	Min
Accuracy	0.9951	0.9987	0.9785
Precision	0.9320	0.9788	0.7975
Recall	0.8782	0.9732	0.7642
F1 Score	0.9036	0.9745	0.7875
ROC AUC	0.9382	0.9856	0.8819

Figure 6 illustrates accuracy and precision across all 42 windows, with performance remaining well above 0.97 in accuracy for every slot. The lowest observed values (Tuesday 1 pm, F1 = 0.787) coincide with periods of heightened legitimate traffic variability, which reduces the signal quality in the frequency-based features.

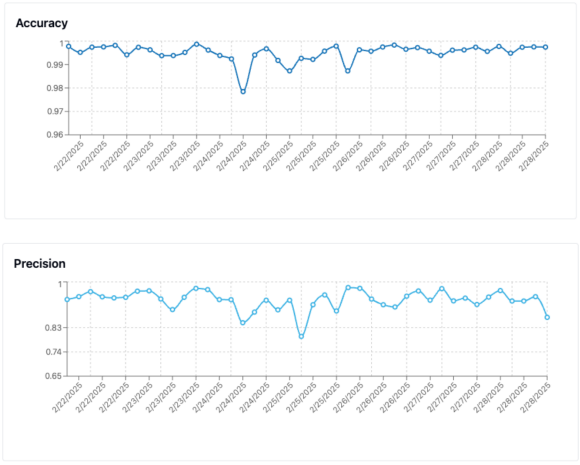

Figure 6: Accuracy and precision over the 42-window weekly sweep (22–28 Feb 2025). Performance remains consistently high despite intra-day and inter-day variation.

4.3.2 Mid-Week Weekday Sweep

Four consecutive Tuesdays in February 2025 (4th, 11th, 18th, 25th), 6–7 pm slot.

Table 5: Performance across 4 mid-week Tuesdays (evening slot).

Metric	Mean	Max	Min
Accuracy	0.9940	0.9979	0.9873
Precision	0.9192	0.9855	0.7975
Recall	0.8837	0.9577	0.7776
F1 Score	0.9005	0.9639	0.7875
ROC AUC	0.9407	0.9757	0.8871

4.3.3 Weekend Sweep

Four consecutive Saturdays in February 2025 (1st, 8th, 15th, 22nd), 1–2 pm slot.

Table 6: Performance across 4 weekend Saturdays (afternoon slot).

Metric	Mean	Max	Min
Accuracy	0.9968	0.9982	0.9942
Precision	0.9571	0.9919	0.9345
Recall	0.9017	0.9654	0.8273
F1 Score	0.9281	0.9734	0.8822
ROC AUC	0.9504	0.9818	0.9131

Weekend performance is notably higher and more stable than mid-week results, likely due to more homogeneous traffic patterns: fraud activity tends to spike during business hours on weekdays, creating harder-to-separate decision boundaries.

4.3.4 Multi-Month Sweep

First Monday of December 2024, January, February, and March 2025, 6–7 pm slot.

Table 7: Performance across four consecutive monthly first Mondays.

Metric	Mean	Max	Min
Accuracy	0.9976	0.9983	0.9973
Precision	0.9490	0.9562	0.9382
Recall	0.8588	0.9602	0.8057
F1 Score	0.9005	0.9582	0.8723
ROC AUC	0.9290	0.9794	0.9026

The multi-month sweep is the most demanding temporal test: the model trained on May 2024 data must generalize across seasonal and behavioral shifts spanning nearly a year. Accuracy remains above 99.7% throughout, and F1 stays above 87%, validating the continual-learning hypothesis.

4.4 Performance by Hour and Day

Figure 7 presents the F1-score heatmap across all (day-of-week, hour) combinations observed in the weekly sweep. Values range from 0.787 (Tuesday 1 pm) to 0.974 (Wednesday 9 am). Two patterns emerge. First, *early-morning and late-evening hours* (7 am and 9 pm) consistently produce higher F1 scores because traffic volumes are lower and more homogeneous. Second, *weekend afternoons* (particularly Saturday 1 pm) achieve some of the highest F1 scores despite high traffic volume, suggesting that weekend fraud patterns are more distinctive.

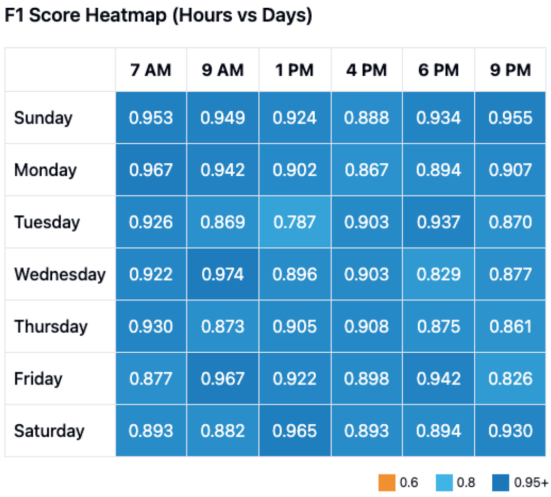

Figure 7: F1-score heatmap by day of week and hour of day. Darker blue indicates higher F1 (≥ 0.95); lighter shading approaches 0.78 at worst.

5 Discussion

5.1 Why Continual Learning Works Here

Traditional ML models for fraud detection suffer from temporal decay: as attackers adapt and user behavior evolves, a static model's recall degrades [2]. Our architecture avoids this by externalizing long-term memory into the FAISS index. The neural network need only learn a good *embedding function*—not the full decision boundary—which changes far more slowly. This decoupling is the core reason the May 2024 model generalizes to March 2025 with minimal loss.

5.2 Feature Robustness

The dominance of behavioral and network velocity features (Table 1) is practically important: unlike frequency-distribution features (e.g., "top-25 countries"), velocity and distance signals are stable across time periods and geographies. Planned ablation studies will quantify the marginal cost of removing the top-k geographic features entirely, potentially yielding a more temporally stable model with only negligible performance loss.

5.3 Interpretability

Unlike deep neural classifiers that produce opaque scores, our system produces an interpretable fraud probability equal to the fraction of the $k = 10$ nearest stored events that were labeled as credential stuffing. Operators can inspect the specific historical events driving any classification decision, satisfy compliance requirements, and explain blocked transactions to customers—a critical capability in the fraud domain [9].

5.4 Limitations

- **Frequency-distribution features.** Top-k geographic and subnet features may drift over multi-year horizons; ablation studies are underway.
- **Labeling latency.** The continual-learning loop depends on human or automated labeling of events; delays degrade the freshness of the FAISS index.
- **Scalability.** Production traffic volumes require migration from `IndexFlatL2` to `IndexIVFFlat` and distributed computing (Spark + PyTorch DDP) for feature engineering and training.
- **Low-and-slow attacks.** The model's performance on "low-frequency" cross-country attacks has not yet been characterized and is earmarked for future study.

6 Next Steps

1. **Attack-type analysis.** Quantify unique IP addresses flagged by the ML model vs. the rule-based system, and characterize performance on cross-country and low-frequency attacks.
2. **Ablation studies.** Remove/replace frequency-distribution features that may become unstable over longer horizons.
3. **Distributed pipelines.** Migrate feature engineering to Spark and training to Databricks with PyTorch distributed training to handle production-scale data volumes.
4. **Production deployment.** Stand up an inference API, a DynamoDB-backed feature store, and a model registry for versioning.

7 Conclusion

We have presented a continual-learning pipeline for credential stuffing detection that replaces a brittle, threshold-based rule system with a neural embedding model backed by FAISS similarity search. On the initial benchmark, the model achieves a **14× improvement in F1 score** (95.58% vs. 6.79%) and retains performance over a nine-month temporal gap without retraining. Across 51 held-out one-hour test windows spanning different times of day, days of week,

and calendar months, mean accuracy never falls below 99.4% and mean F1 never falls below 90%. The solution is computationally efficient, interpretable by design, and well-positioned for production deployment with modest additional engineering investment.

References

[1] Giovanni Apruzzese, Mauro Andreolini, Luca Ferretti, Mirco Marchetti, and Michele Colajanni. Modeling realistic adversarial attacks against network intrusion detection systems. *Digital Threats: Research and Practice*, 3(3):1–19, 2022. doi: 10.1145/3469659.

[2] Alejandro Correa Bahnsen, Djamila Aouada, Aleksandr Stojanovic, and Bjorn Ottersten. Feature engineering strategies for credit card fraud detection. *Expert Systems with Applications*, 51: 134–142, 2016. doi: 10.1016/j.eswa.2015.12.030.

[3] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, pages 553–567. IEEE, 2012. doi: 10.1109/SP.2012.44.

[4] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1735–1742, 2006. doi: 10.1109/CVPR.2006.100.

[5] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019. doi: 10.1109/TBDATA.2019.2921572.

[6] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114.

[7] Leland McInnes, John Healy, and James Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[8] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2001–2010, 2017. doi: 10.1109/CVPR.2017.587.

[9] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1:206–215, 2019. doi: 10.1038/s42256-019-0048-x.

[10] Chao Shen, Zhongmin Cai, Xiaohong Guan, and Roy Maxion. User authentication through mouse dynamics. volume 8, pages 16–30, 2013. doi: 10.1109/TIFS.2012.2223677.

[11] Kurt Thomas, Frank Li, Ali Zand, Jacob Barrett, Juri Ranieri, Luca Invernizzi, Yarik Markov, Oxana Comanescu, Vijay Eranti, Angelika Moscicki, Daniel Margolis, Vern Paxson, and Elie Bursztein. Data breaches, phishing, or malware? Understanding the risks of stolen credentials. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1421–1434. ACM, 2017. doi: 10.1145/3133956.3134067.

[12] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[13] Marie Vasek and Tyler Moore. Do malware reports expedite cleanup? An experimental study. In *Proceedings of the 9th USENIX Workshop on Cyber Security Experimentation and Test (CSET)*. USENIX Association, 2016.

[14] Yan Zhang, Jonathon Hare, and Adam Prügel-Bennett. Deep attention neural tensor network for visual question answering. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 20–35. Springer, 2018.