Threat Model Information

Application Name: Adversarial Stylometry for Codeship Author Attribution
Description: An ML pipeline that predicts the likely author of a code snippet using stylometric features. Training uses a labeled repository of code samples per author; outputs include predicted author, confidence, and warnings.
Document Owners: Andrew Varkey, Elang Sisson
Reviewer: Tashi Stirewalt

Scope

Concise Objective:
Primary - Evade attribution to the true author
Secondary - Induce misattribution to a specific target author
Out of scope objectives - DoS, Model or data theft, privacy attacks

External Dependencies (Black/Grey Box)

ID	Description
ED1	System Under Test: Third party Stylometry Attribution Service - Hosted by Client (Tashi), we have query-only access. No current access to the source, training data, or model internals.
ED2	Supported Languages & Limits -TBD: Programming languages accepted, max upload size.
ED3	Output Contract - expected fields: predicted author label, confidence score, warnings/notes.
ED4	Sample data - Open source corpora for testing.
ED5	Usage Constraints - Follow clients acceptable-use, submit code we are only permitted to transform.
ED6	Change Control - Service may change, we will record any version changes if necessary for each experiment run.

Entry Points

ID	Name	Description	Trust Levels
EP1	HTTPS Prediction API	Primary Interface to submit a single code sample and receive an authorship prediction.	TL2
EP2	Bulk Submission	UI for submitting many files at once (CSV). This is dependent on the size limit and if the model even allows multiple submissions.	TL2
EP3	Training Data Ingestion	The pipeline for adding training data	TL3

Exit points

These are the observable outputs and provider-side emissions we assume exist. We plan to confirm and alter these during testing as we go.

ID	Name	Description
XP1	Downloadable Report	Batch run summary, CSV/PDF download
XP2	Client-Visible timing	Latency in our measurement of request round-trip time.

Assets

Since this is a black/gray box attack, we include the provider assets that we are impacting and then our red team assets.

Provider Owned Assets

ID	Name	Description	Access (TL)	Sensitivity
A1	Training Corpus	Labeled code per author used to	TL3 - TL4	High

		train the system	(Provider)	
A2	Preprocess/Feature Code	Parsers, feature extraction rules	TL4	Med-High
A3	Trained Model	Model weights, feature config, thresholds	TL4	High
A4	Attribution Service	Web UI returning predictions	TL2 (us)	Med
A5	Logs and Telemetry	Request logs, model/Version ID's, metrics	TL4	Med-High
A6	Policy/Allow List	Languages, Size Limits,	TL4	Low

Red Team Assets

ID	Name	Description	Access (TL)	Sensitivity
RT1	Seed Corpera	Public code that we will use as source inputs	TL2 Team	Low
RT2	Transformed Variants	Adversarial edits (White space, rename, refactor, mimicry) with labels of which transformations were applied.	TL2	Med
RT3	Experiment Harness	Notebooks to generate variants	TL2	Low
RT4	Results Store	Captured responses: status, latency, JSON	TL2	Med
A5	Provenance and Audit Logs	Run ID's, tool versions, System observed version	TL2	Low

Trust Levels

This defines who can do what. In the system test setup, TL2 is where we operate, while TL3-4 are the providers internal roles.

ID	Name	Who	Description
TL1	Anonymous Submitter	This would be for anyone who has access to the public UI (Assuming it exists)	Can post a single code snippet for attribution with no other historical

			access
TL2	Authenticated Client	Our team using the API	Query predictions, view our own results.
TL3	Data Engineer	Providers data team	Ingest training data, relabel or remove samples
TL4	Admin	Providers Ops	Deploy or rollback models, change major features

Determine Threats

STRIDE Threat Lists:

S- Spoofing:

- Attacker impersonates the author by crafting code to match their style
- Usage of models to mimic target authors
- Addition of author names in comments to mislead labeling

Goal is to misattribute output to a specific false author

T - Tampering

- Modify dataset by mislabeling code samples

Goal is to corrupt the integrity of model or dataset so attribution accuracy is degraded

R - Repudiation:

- Author denies authorship of code after evading detection
- No signatures or metadata to confirm original submission source

Prevent reliable verification of the true author identity

I - Information Disclosure

- Unauthorized access to datasets or author metadata
- Results are public to attacker which exposes the models behavior

Gain insight into how the model works to improve attacks

D - Denial of service:

- Force model retraining cycle by poisoning or corrupting dataset.
- Cause crashes by sending non binary inputs
- Flood model with large or malformed code files to exhaust resources

Disrupt availability of the system

E - Elevation Privilege:

- Unauthorized user escalation to modify training data or model weights
- Somehow exploit misconfigured access controls in dataset or computer

Goal is to gain access and persistent control

Threat Analysis:

S - Spoofing:

Target Components: Code submission input and author identity metadata

Possible Exploitation Path:

1. Attacker uses an LLM to generate variants matching the author's style
2. Submit these variants through the model's input channel
3. Model misattributes to the impersonated author

Root Cause:

- Reliance on stylistic features without semantic verification
- Lack of digital signatures

Likelihood: High

Impact: Medium-High

T - Tampering:

Target Components: Training dataset, model weights

Possible Exploitation Path

1. Attacker adds mislabeled sample into training corpus
2. Model learns incorrect author style mappings
3. Attribution accuracy degrades

Root Cause:

- Insecure data ingestion
- Absence of dataset validation or integrity checks

Likelihood: Medium

Impact: High

R - Repudiation:

Target Components: Logging, storage, and how it audits

Possible:

1. Author delete or overwrites submission records
2. No digital signatures exists
3. Attribution record cannot be verified

Root Cause:

- Insufficient logging

Likelihood: Medium

Impact: Medium

I – Information Disclosure

Target Components: System logs, dataset storage

Possible Exploitation Path:

1. Attacker queries model repeatedly and observes scores

2. Determines feature sensitivity or model's biases.
3. Uses this information to craft better evasions

Root cause:

- Overly detailed model output and lack of access restrictions

Likelihood: High
Impact: Medium

D - Denial of Service:
Target Components: Data ingestion pipeline, compute cluster
Possible Exploitation Path:

1. Attacker floods the model with large or malformed submissions.
2. Resources become saturated
3. Defender forced to pause model

Root cause:

- No input size validation or rate limiting

Likelihood: High
Impact: Low-Medium

E - Elevation of Privilege:
Target Components: File permissions, container environments
Possible Exploitation Path:

1. Attacker gains unauthorized model access
2. Modifies datasets
3. Maintains persistence by creating hidden backdoors

Root Cause:

- Weak access control or insecure environment configuration

Likelihood: Low
Impact: High

Qualitative Risk Model

Threat	Likelihood	Impact	Overall Risk
Spoofing	High	Medium - High	High
Tampering	Medium	High	High
Repudiation	Medium	Medium	Medium
Information Disclosure	High	Medium	High
Denial of Service	High	Low-Medium	Medium
Elevation of Privilege	Low	High	Medium

Determine Countermeasures and Mitigation

S - Spoofing:
Threat: Attacker impersonates another author by crafting code to match their style
Mitigations:

- Use digital signatures or provenance tags to verify the author of submitted code.
- Add semantic verification such as behavioral fingerprints to validate real authorship
- Apply adversarial training with spoofed samples so the model learns to detect artificial style mimicry
- Perform style consistency checks across multiple code samples per author

T - Tampering:
Threat: Modification or poisoning of datasets, model files, or logs.
Mitigations:

- Enforce access control and authorization on all datasets or model directories
- Use hashes or checksums to verify dataset and model integrity
- Store data in version controlled repositories
- Add read only storage policies

R - Repudiation:
Threat: Author denies submission or manipulates audit records
Mitigations:

- Implement digital signatures on every submission and results
- Include timestamps and submission IDs

I - Information Disclosure:
Threat: Sensitive data or model behavior is exposed.
Mitigations:

- Restrict model outputs to the essential results only such as labels and a confidence score range
- Use role based authorization for access to dataset and logs
- Anonymize training data when sharing data results
- Sanitize logs to remove author identifiers

D - Denial of Service:
Threat: Overloading the model or data ingestion pipeline to make it unavailable
Mitigations:

- Add rate limiting and input size validation
- Filter or reject oversized inputs

- Queue or throttle long running jobs to preserve system
- Monitor for unusual traffic or resources spikes

E - Elevation of Privilege:
Threat: Attacker gains unauthorized control
Mitigations:

- Use separate accounts for training and evaluation
- Log and alert on all privilege escalations
- Apply least privilege principle to users and services