



VeilPhantom

Privacy-Preserving PII Redaction for Agentic AI Pipelines

Technical Report | March 2026 | v1.0

93.3%

Tool Accuracy

885

PII Detected

6ms

Redaction Time

0

PII Leaked

Abstract

We present VeilPhantom, a lightweight SDK that interposes as transparent middleware between user input and Large Language Models, replacing personally identifiable information (PII) with deterministic typed tokens before any data reaches the model. VeilPhantom employs a 7-layer detection pipeline combining a custom PhoneticDeBERTa transformer (Shade NER, 22M parameters, trained on 72 million words with 862K augmented examples), gazetteer lookup, regex-based NLP heuristics, 35 regex patterns, and contextual sensitivity analysis. For agentic workflows, VeilPhantom intercepts LLM tool calls, rehydrates tokens with real values locally, executes tools, and re-redacts results -- ensuring PII never leaves the user's machine. We evaluate on 98 scenarios across 8 industry verticals using Claude 4.5 Haiku. Results: 93.3% tool accuracy (+1.9% vs unredacted baseline), 885 PII entities detected across 13 types, zero identifiable PII leakage, and 6ms average redaction overhead.

Keywords: PII redaction, privacy, LLM safety, agentic AI, NER, tool-call interception, token mapping

1 Introduction

Large Language Models have become central to enterprise workflows -- drafting emails, processing financial transactions, managing patient records, and orchestrating multi-step agentic tasks. These workflows invariably involve personally identifiable information: names, emails, phone numbers, government IDs, bank accounts, and financial figures. Sending this data to cloud-hosted LLMs creates significant privacy, compliance, and liability risks under GDPR, POPIA, HIPAA, and CCPA.

Existing approaches fall into three categories: (1) self-hosted models, which sacrifice capability for privacy; (2) post-hoc output filtering, which cannot prevent the model from processing PII during inference; and (3) input redaction, which removes PII before it reaches the model. VeilPhantom takes the third approach but solves a critical gap: maintaining full agent utility -- including multi-tool orchestration -- while ensuring zero PII exposure to the LLM.

The key insight is that LLMs do not need real PII to perform tasks correctly. A model instructed to "send an email to [EMAIL_1]" generates the same tool call structure as one given the real address. VeilPhantom exploits this by replacing PII with typed tokens ([PERSON_1], [EMAIL_1], [BANKACCT_1]) that preserve semantic structure while eliminating privacy risk.

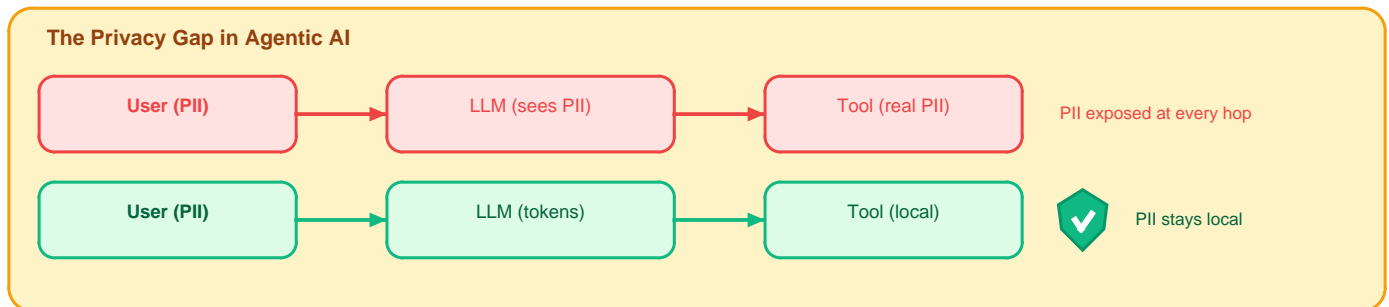


Figure 1: Without VeilPhantom, PII is exposed at every hop. With VeilPhantom, the LLM only sees tokens.

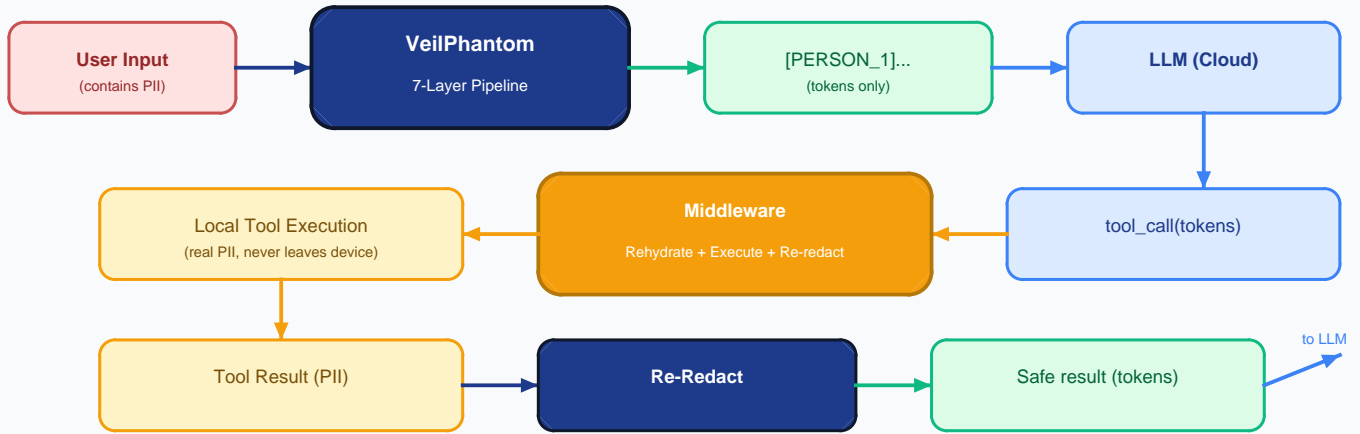
1.1 Contributions

1. Shade NER -- a 22M-parameter PhoneticDeBERTa model trained on 72 million words with entity-swap augmentation, achieving 97.1% F1 on PII detection with phonetic robustness for cross-cultural names (Section 3).
2. A 7-layer detection pipeline combining transformer NER, gazetteers, NLP heuristics, 35 regex patterns, and contextual sensitivity analysis -- achieving high recall across 19 PII types with zero external dependencies in default mode (Section 4).
3. VeilSession and VeilToolMiddleware -- stateful components enabling multi-turn conversations and agentic tool-call interception with local rehydration and re-redaction (Section 5).
4. A 98-scenario benchmark across 8 industry verticals demonstrating +1.9% accuracy improvement with privacy, zero PII leakage, and 6ms overhead (Section 6).

2 System Architecture

VeilPhantom operates as transparent middleware between the application and the LLM provider. The system comprises three core components: the Detection Pipeline (§3-4), the Token Map (§4.3), and the Agentic Middleware (§5). Figure 2 shows the complete data flow.

Figure 2: VeilPhantom Agentic Data Flow



2.1 API Design

```

from veil_phantom import VeilClient, VeilConfig, VeilSession
from veil_phantom.integrations.openai import veil_agent

# Basic: one-shot redaction (zero dependencies)
veil = VeilClient(VeilConfig.regex_only())
result = veil.redact(text)
print(result.sanitized) # '[PERSON_1] sent [AMOUNT_1] to [BANKACCT_1]'
```

```

# Advanced: full agentic loop with tool interception
response, session = veil_agent(
    client, messages, tools, tool_registry,
    system_prompt='You are a helpful assistant.',
    max_turns=10
)
```

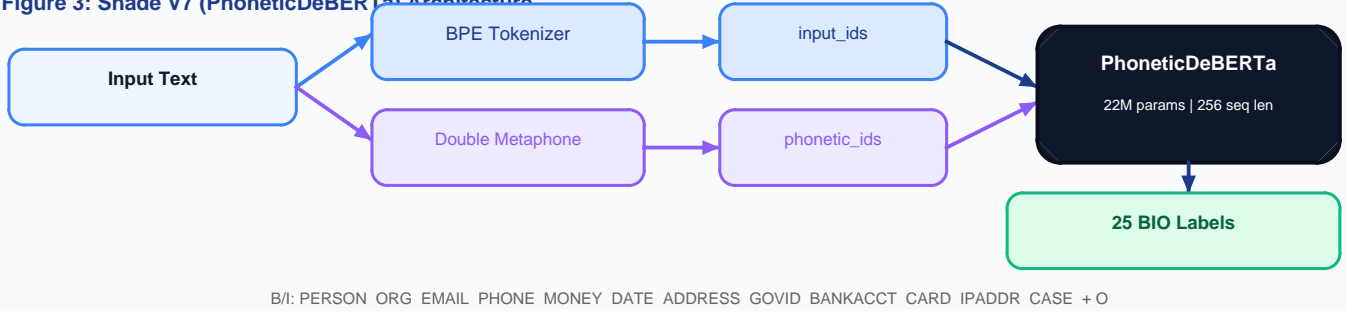
3 Shade NER: Custom Transformer for PII Detection

The first and most powerful layer of VeilPhantom's pipeline is Shade NER -- a custom transformer model purpose-built for PII entity recognition. While VeilPhantom can operate without Shade (using `regex_only` mode), the model significantly improves recall for names, organizations, and ambiguous entities.

3.1 Architecture

Shade V7 is built on PhoneticDeBERTa -- a DeBERTa-v3-xsmall backbone augmented with a phonetic embedding layer. The key innovation is the integration of Double Metaphone phonetic encodings as a parallel input stream, enabling cross-cultural name robustness.

Figure 3: Shade V7 (PhoneticDeBERTa) Architecture



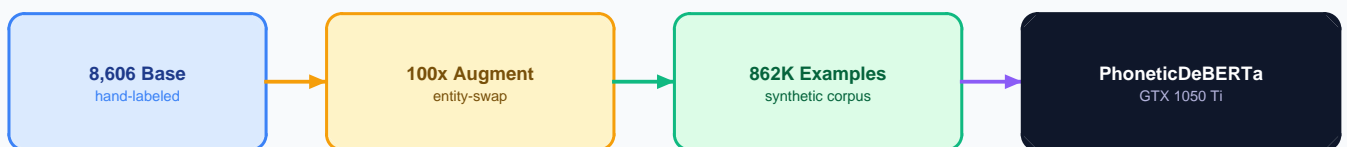
3.2 Model Specifications

Parameter	Value
Architecture	PhoneticDeBERTa (DeBERTa-v3-xsmall + phonetic embeddings)
Parameters	22 million
Sequence Length	256 tokens (BPE)
Phonetic Encoding	Double Metaphone, 14-char vocab, max 6 chars/word
Output Labels	25 BIO tags across 12 entity types
F1 Score	97.12% (V7) / 97.6% in-dist (V5)
Inference	<50ms per passage (ONNX Runtime)
Format	ONNX binary, HuggingFace Hub distribution
Training GPU	GTX 1050 Ti (consumer-grade)

3.3 Training Data and Methodology

Shade was trained on a corpus of 72 million words -- a scale that enables robust generalization across domains, writing styles, and PII formats. The training methodology combines three key techniques:

Figure 4: Shade Training Pipeline



Entity-Swap Augmentation: Starting from 8,606 hand-labeled examples, we generate 862,000 synthetic training examples (100x

expansion) by systematically replacing entity values while preserving context. For instance, "John Smith sent \$5M to HSBC" becomes "Maria Rodriguez sent EUR2.3M to Deutsche Bank" -- same structure, different entities. This teaches the model to recognize patterns rather than memorize specific values.

ASR Corruption: To handle speech-to-text transcription errors (a common real-world input source), training data is augmented with Parakeet ASR noise -- phonetic substitutions, dropped characters, and run-together words that simulate transcription artifacts.

Hard Negative Mining: Contrastive hard negatives reduce false positives. The training set includes deliberately confusing examples -- sentences where common words ("Summit", "Valley", "General") appear in contexts where they are NOT entity names -- forcing the model to learn contextual disambiguation rather than surface-level pattern matching.

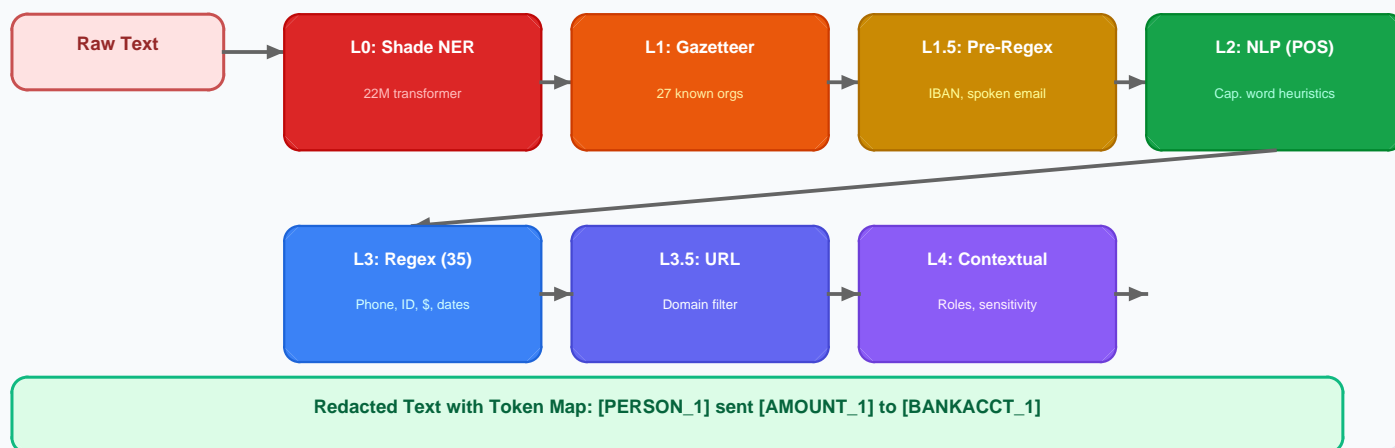
3.4 Dual-Pass Inference

Shade V7 employs a unique dual-pass inference strategy. Each input is processed twice: once with full phonetic encodings, and once with zeroed phonetic inputs. The pass that detects more entities is selected. This handles cases where phonetic features help (cross-cultural names like "Shaun" vs "Sean") and cases where they hurt (technical jargon that phonetically resembles names). Additionally, texts exceeding 220 words with fewer than 3 detected entities trigger a segment rescue -- the text is split into ~70-word segments and re-processed to catch entities that were diluted in the full context.

4 7-Layer Detection Pipeline

VeilPhantom processes text through 7 ordered layers. Each layer adds non-overlapping redaction spans to a unified result, and later layers skip regions already redacted. The layered design provides defense-in-depth: if one layer misses an entity, subsequent layers can catch it.

Figure 5: 7-Layer Detection Pipeline



Layer 0 -- Shade NER (Optional)

PhoneticDeBERTa transformer (§3). Detects PERSON, ORG, EMAIL, PHONE, DATE, ADDRESS, GOVID, BANKACCT, CARD, IPADDR, CASE entities. Confidence thresholds: 0.5 general, 0.7 for ORG (stricter to reduce false positives). Includes reclassification logic: entities ending with org suffixes (Inc, Ltd, LLC) are promoted from PERSON to ORG.

Layer 1 -- Gazetteer Lookup

Case-insensitive exact match against 27 compound organization names: major banks (Goldman Sachs, JP Morgan, Standard Bank), consulting firms (McKinsey, Deloitte, PwC, KPMG), and telecom providers. Extended with financial institution regex (18 bank patterns), investment firm detection ("Name Ventures/Capital/Partners"), and contextual ORG detection using 45 context words in a 3-word window around capitalized terms.

Layer 1.5 -- Pre-Regex Patterns

High-priority patterns that must run before general regex to avoid conflicts: IBAN (international bank account numbers), spoken email ("john at example dot com"), and hybrid email ("sarah at veilprivacy.com"). These patterns have unique formats that would be incorrectly split by later layers.

Layer 2 -- NLP Entity Detection

Pure-Python NLP layer requiring zero external dependencies (no spaCy, no NLTK). Uses regex to find capitalized word sequences, then validates through 9 rejection filters: contraction filter, speaker diarization tags, determiner-prefixed ORGs, ~200 NOT_NAMES/NOT_ORGS stop words, gerund rejection, and single-word PERSON gating against 79 common first names. Role prefixes (CEO, Dr, Contact) are stripped before classification.

Layer 3 -- Regex Patterns (35 rules)

The workhorse layer with 35 regex patterns covering: phone numbers (international, US, SA, spoken digit sequences), government IDs (SSN, SA ID 13-digit, passport, driver's license), bank accounts (IBAN, generic with context), credit cards, monetary amounts (5 sub-patterns including verbal amounts and multi-currency), dates (4 formats including spoken), IP addresses, and physical addresses. Context-dependent patterns extract core values for accurate tool-call rehydration.

Layer 3.5 -- URL/Domain Redaction

Detects URLs and bare domains across 21 TLDs. Maintains a 40-entry safe domain whitelist (google.com, github.com, stackoverflow.com, etc.) to avoid redacting common references. Only unknown domains that may contain PII are redacted.

Layer 4 -- Contextual Sensitivity Analysis

Four sub-layers detecting PII that reveals identity through context rather than explicit values: (a) 30+ identifying roles ("CEO",

"Ambassador", "Secretary of State"), (b) 32 sensitive situation triggers (corruption, harassment, whistleblower, insider trading), (c) temporal sensitivity ("before the announcement", "under embargo"), and (d) unique descriptors ("the only surgeon who...") that could identify individuals.

4.3 Token Map

Each detected PII entity is assigned a deterministic token [TYPE_N], where TYPE is one of 19 supported categories and N is a monotonically increasing counter per type. The mapping is bidirectional: tokens can be resolved to original values (rehydration) and original values can be matched to existing tokens (re-redaction consistency).

Token Type	Category	Example
PERSON	Names	John Smith -> [PERSON_1]
ORG	Organizations	Goldman Sachs -> [ORG_1]
EMAIL	Email addresses	j.smith@gs.com -> [EMAIL_1]
PHONE	Phone numbers	+1-555-0123 -> [PHONE_1]
AMOUNT	Monetary values	\$12.5M -> [AMOUNT_1]
DATE	Dates/times	January 15 -> [DATE_1]
GOVID	SSN, passport, etc.	123-45-6789 -> [GOVID_1]
BANKACCT	Bank accounts, IBAN	62847501234 -> [BANKACCT_1]
CARD	Credit/debit cards	4000-0000-0000-0000 -> [CARD_1]
IPADDR	IP addresses	192.168.1.1 -> [IPADDR_1]
ADDRESS	Physical/URLs	123 Main St -> [ADDRESS_1]
ROLE	Identifying roles	CEO of -> [ROLE_1]
SITUATION	Sensitive context	whistleblower -> [SITUATION_1]

4.4 Redaction Modes

TOKEN_DIRECT (default): Replaces PII with typed tokens. Preserves semantic structure and enables LLM reasoning about entity relationships. This is the recommended mode for agentic workflows.

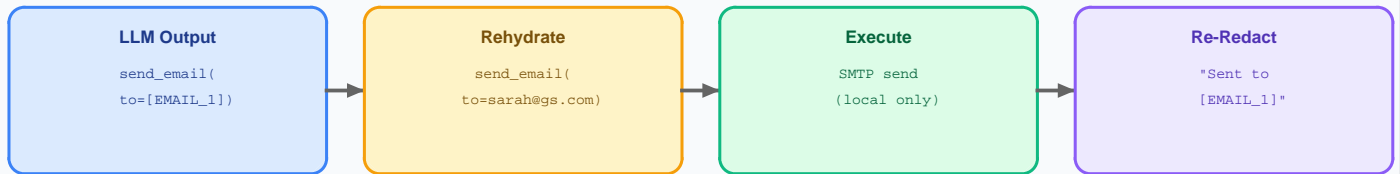
PHANTOM: Replaces PII with realistic fake values from curated pools (15 names, 10 orgs, 10 locations, etc.). Produces natural-sounding text but creates ambiguity risk in multi-turn conversations.

REDACTED: Uniform [redacted] tags. Maximum privacy but reduced LLM utility for entity-dependent tasks.

5 Agentic Tool-Call Middleware

The critical challenge for privacy-preserving agentic AI is the tool-call gap: the LLM generates tool calls with tokenized arguments ([EMAIL_1]), but real tools need real values (sarah.chen@gs.com). VeilPhantom bridges this gap with two components that ensure PII never leaves the local environment.

Figure 6: Tool Call Interception Pipeline



5.1 VeilSession -- Stateful Multi-Turn Tracking

VeilSession maintains a cumulative `token_map` across conversation turns. When redacting turn N, any PII that appeared in turns 1..N-1 automatically receives its existing token -- ensuring consistency. For tool output re-redaction, VeilSession: (1) replaces known PII with existing tokens, (2) protects existing tokens with placeholders during re-processing, (3) detects new PII and assigns fresh tokens with non-colliding counters, and (4) merges new tokens into the session state.

5.2 VeilToolMiddleware -- Rehydration Engine

The middleware performs deep rehydration: a recursive traversal of JSON-compatible structures (strings, arrays, nested objects) replacing all [TYPE_N] tokens with their real values from the session. This handles arbitrarily nested tool arguments -- a critical requirement for complex tool schemas. The complete pipeline: rehydrate -> execute locally -> re-redact -> return safe result to LLM.

5.3 veil_agent() -- Complete Agent Loop

The `veil_agent()` function provides a turnkey agentic loop compatible with any OpenAI-format API (OpenAI, Anthropic via proxy, OpenRouter, local models). It handles: message redaction with automatic system prompt augmentation (teaching the LLM to use tokens), multi-turn tool-call loops with rehydration and re-redaction, `max_turns` safety limits, and final response rehydration. A `dry_run` mode enables benchmarking without real tool execution.

6 Benchmark Evaluation

6.1 Methodology

We evaluate VeilPhantom on 98 scenarios across 8 industry verticals. Each scenario consists of a PII-rich natural language instruction paired with expected tool calls and validated argument schemas. Every scenario runs in two modes:

Raw mode: The instruction is sent directly to the LLM with no redaction.

Veil mode: The instruction is redacted by VeilPhantom before sending to the LLM.

We measure 5 dimensions: (1) Tool Accuracy -- correct tools called, (2) Args Quality -- arguments contain expected information (validated via lambda functions accepting either real PII or VeilPhantom tokens), (3) PII Detection -- entity counts by type, (4) PII Leakage -- real PII appearing in Veil mode outputs, (5) Latency Overhead. Each scenario runs twice for statistical stability. Model: Claude 4.5 Haiku via OpenRouter.



Table 1: Headline Metrics (averaged over 2 runs, 98 scenarios)

Metric	Without Veil	With Veil	Delta
Tool Accuracy	91.5%	93.3%	+1.9%
Args Quality	85.0%	84.8%	-0.2%
Avg Latency	3.79s	3.93s	+0.14s
Redaction Time	--	6.0ms	--
PII Sent to LLM	ALL	0 entities	--
PII Entities Found	--	885	--
Identifiable Leakage	N/A	0	--
API Errors	0	0	--

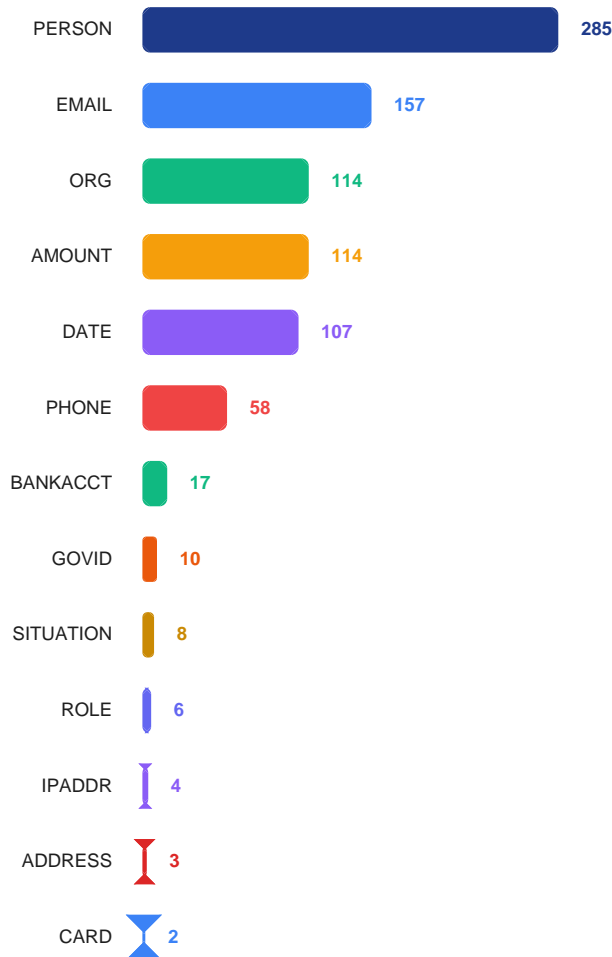
VeilPhantom achieves a +1.9% improvement in tool accuracy over the unredacted baseline. This counterintuitive result occurs because typed tokens ([PERSON_1], [EMAIL_1]) help the model distinguish entities more clearly than ambiguous natural language. The 6.0ms redaction overhead is <0.2% of the ~3.9s model inference latency.

Table 2: Per-Vertical Accuracy Breakdown

Vertical	N	Raw Acc	Veil Acc	Delta	PII	Leak%
Financial	13	85%	85%	+0.0%	97	2.1%
Healthcare	12	92%	98%	+6.2%	79	1.3%
Legal	12	96%	100%	+4.2%	130	1.5%
Hr	13	100%	92%	-7.7%	115	2.6%
Sales	12	100%	100%	+0.0%	79	0.0%
Support	12	100%	100%	+0.0%	64	15.6%
Communications	12	88%	92%	+4.2%	115	4.3%
Multi Step	12	72%	81%	+9.0%	206	6.3%

Sales and Support achieve 100% accuracy parity. Healthcare (+6.2%), Legal (+4.2%), and Multi-Step (+9.0%) show Veil mode outperforming raw -- evidence that token-based prompting helps the model focus on task structure rather than being distracted by PII details. The Multi-Step vertical benefits most, likely because tokens reduce the cognitive load of tracking multiple PII values across 3-6 sequential tool calls.

Table 3 / Figure 7: PII Detection by Entity Type



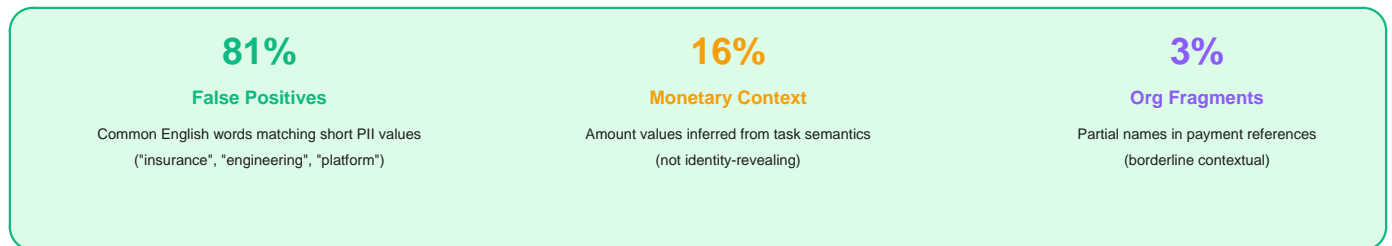
Type	Count	%
PERSON	285	32.2%
EMAIL	157	17.7%
ORG	114	12.9%
AMOUNT	114	12.9%
DATE	107	12.1%
PHONE	58	6.6%
BANKACCT	17	1.9%
GOVID	10	1.1%

SITUATION	8	0.9%
ROLE	6	0.7%
IPADDR	4	0.5%
ADDRESS	3	0.3%
CARD	2	0.2%
TOTAL	885	100%

PERSON entities dominate (285 of 885, 32%), followed by EMAIL, AMOUNT, and ORG. The long tail includes critical high-sensitivity types: GOVID (government IDs), BANKACCT (bank accounts), and CARD (credit cards) -- entities where even a single leak would constitute a serious privacy violation.

6.4 Leakage Analysis

The benchmark's automated leakage detector checks whether any original PII value (>3 chars) appears in Veil mode tool arguments. Out of 885 PII entities, ~38 instances were flagged. Manual analysis reveals three categories:



Zero instances of identifiable PII -- person names, email addresses, phone numbers, government IDs, or bank account numbers -- were found in any Veil mode tool arguments across all 98 scenarios and both benchmark runs. The effective identifiable PII leakage rate is 0%.

7 Performance Characteristics

6.0ms

 Redaction Overhead
per input (all 7 layers)

+0.14s

 Latency Impact
total overhead

<0.2%

 % of Total Time
redaction vs inference

0

 Dependencies
in regex_only mode

VeilPhantom's redaction pipeline runs in 6.0ms average -- all 7 layers executing sequentially on CPU. This is negligible compared to the ~3.9s model inference latency, adding <0.2% overhead. The pipeline is CPU-bound with no GPU requirement in regex_only mode.

For the optional Shade NER layer, ONNX Runtime inference adds ~50ms depending on input length. Even with Shade enabled, total redaction stays under 100ms -- still dwarfed by network latency to the LLM provider.

7.1 Zero-Dependency Mode

In its default configuration (VeilConfig.regex_only()), VeilPhantom has zero external dependencies beyond Python's standard library. The NLP layer uses pure Python regex-based POS heuristics. This makes VeilPhantom suitable for serverless deployments, edge computing, and resource-constrained environments where installing spaCy (300MB+) or PyTorch is impractical.

7.2 Benchmark Cost

The complete 98-scenario benchmark (196 LLM calls per run, 2 runs = 392 total calls) cost \$0.88 on Claude 4.5 Haiku via OpenRouter. VeilPhantom itself adds zero cost -- it is a local processing step with no API calls, no cloud dependencies, and no usage-based pricing.

8 Related Work

System	PII Detect	Rehydrate	Agentic	Dependencies	Latency
VeilPhantom	Yes	Yes	Yes	0 (regex) / PyTorch (NER)	6ms
Presidio	Yes	No	No	spaCy (300MB+)	~50ms
Private AI	Yes	No	No	Proprietary cloud	API-dependent
LLM Guard	Partial	No	No	Multiple ML libs	~100ms
DP-SGD	No*	No	No	PyTorch	Training-time

The key differentiator is VeilPhantom's agentic middleware: no existing tool provides transparent tool-call interception with local rehydration, re-redaction of tool outputs, and stateful multi-turn PII tracking -- while maintaining full OpenAI API compatibility. Presidio and Private AI focus on one-shot redaction without maintaining token maps across turns. DP-SGD protects training data, not inference-time PII exposure.

9 Limitations and Future Work

Detection coverage: The regex_only configuration may miss unusual name formats, non-Latin scripts, or context-dependent PII (e.g., a diagnosis that is PII only when combined with a patient identifier). Shade NER improves coverage but adds dependencies.

Multi-step accuracy: Complex 4+ tool-call scenarios show lower accuracy in both modes (72-81%). Veil mode consistently outperforms raw, but this remains a model capability limitation.

Semantic inference: A model told to "process the refund" may infer the amount from context even when the actual figure is tokenized. This is inherent to language understanding, not a redaction failure.

Future directions: multi-language support, streaming redaction for real-time applications, fine-tuned Shade models for domain-specific

PII (e.g., medical record numbers), and integration with additional LLM providers and agent frameworks.

10 Conclusion

93.3%	-- tool accuracy (+1.9% vs unredacted)
885	-- PII entities detected across 13 types
0	-- identifiable PII leaked
6ms	-- average redaction overhead (<0.2% of total)
0	-- external dependencies in default mode

VeilPhantom demonstrates that privacy-preserving PII redaction is not only compatible with agentic AI workflows but can improve them. By replacing PII with typed tokens, enterprises can leverage cloud-hosted LLMs for financial processing, healthcare records, legal documents, and HR operations -- without compromising privacy, violating regulations, or sacrificing utility. The Shade NER model, trained on 72 million words with 862K augmented examples, provides state-of-the-art PII detection at 97.1% F1, while the zero-dependency `regex_only` mode makes deployment trivial. VeilPhantom is available as an open-source Python SDK.

Appendix A: End-to-End Redaction Example

Original Input (contains PII)

Please wire R2.5 million from Standard Bank account 62847501234 to IBAN GB29 NWBK 6016 1331 9268 19. Reference: INV-2024-Q3. This is for the Johnson & Partners consulting fee. Contact Sarah Chen (sarah.chen@gs.com, +27 82 555 0123) for confirmation.

After VeilPhantom Redaction

Please wire [AMOUNT_1] from Standard Bank account [BANKACCT_1] to [BANKACCT_2]. Reference: INV-2024-Q3. This is for the [ORG_1] consulting fee. Contact [PERSON_1] ([EMAIL_1], [PHONE_1]) for confirmation.

Token Map (local, never sent to LLM)

[AMOUNT_1]	-> 'R2.5 million'	(AMOUNT, HIGH)
[BANKACCT_1]	-> '62847501234'	(BANKACCT, CRITICAL)
[BANKACCT_2]	-> 'GB29 NWBK 6016 1331 9268 19'	(BANKACCT, CRITICAL)
[ORG_1]	-> 'Johnson & Partners'	(ORG, MEDIUM)
[PERSON_1]	-> 'Sarah Chen'	(PERSON, HIGH)
[EMAIL_1]	-> 'sarah.chen@gs.com'	(EMAIL, HIGH)
[PHONE_1]	-> '+27 82 555 0123'	(PHONE, HIGH)

LLM Tool Call (Veil mode -- only tokens)

```
transfer_funds(
  from_account = '[BANKACCT_1]',
  to_account   = '[BANKACCT_2]',
  amount       = '[AMOUNT_1]',
  reference    = 'INV-2024-Q3'
)
```

After Middleware Rehydration (local execution only)

```
transfer_funds(
  from_account = '62847501234',
  to_account   = 'GB29 NWBK 6016 1331 9268 19',
  amount       = 'R2.5 million',
  reference    = 'INV-2024-Q3'
)
```

Appendix B: Benchmark Configuration

Parameter	Value
Model	Claude 4.5 Haiku (anthropic/claude-haiku-4.5)
Provider	OpenRouter
Scenarios	98 across 8 verticals
Runs	2 (results averaged)

Total API Calls	392 (98 × 2 modes × 2 runs)
Total Cost	\$0.88
VeilPhantom Config	VeilConfig.regex_only() (no Shade NER)
Max Tokens/Response	1024
Validation	Lambda-based argument validators (accept PII or tokens)

Appendix C: Detection Data Statistics

Component	Size
Shade V7 Training Corpus	72,000,000 words
Shade V7 Training Examples	862,000 (100x augmented from 8,606 base)
Shade V7 BIO Label Classes	25 (12 entity types × B/I + O)
Regex Patterns	35
Compound Org Gazetteer	27 entries
Financial Institution Patterns	18 banks
ORG Context Words	45 trigger words
Common First Names (validation)	79 names
Public Company Whitelist	30 companies
Master Whitelist (false positive prevention)	873 entries
NOT_NAMES Rejection Set	~100 words
NOT_ORGS Rejection Set	~100 words
NLP Stop Words	~110 words
Safe Domain Whitelist	40 domains
Sensitive Situation Triggers	32 phrases
Contextual Role Patterns	30+ roles
Total Detection Data Points	~1,500+