ML-Powered Security Operations: Deep Learning for Threat Detection

LSTM Anomaly Detection, Autoencoder Zero-Day Detection, NLP Threat Intelligence, and Federated Learning for Privacy-Preserving Security

Wesley Robbins • STSGYM Research • April 2026

> **Technical Deep Dive** — This paper details the machine learning architecture behind KaliAgent v5.0.0, covering model design, training methodology, GPU acceleration benchmarks, and production deployment on Kubernetes with full observability.

Table of Contents

1. Motivation & Problem Statement

Traditional security operations face three fundamental limitations that ML can address:

1.1 Alert Fatigue

Enterprise SOC teams receive 200,000+ alerts per day. Human analysts triage at roughly 10–15 alerts per hour. The math doesn't work: most alerts go uninvestigated, and critical threats hide in the noise.

1.2 Signature Dependency

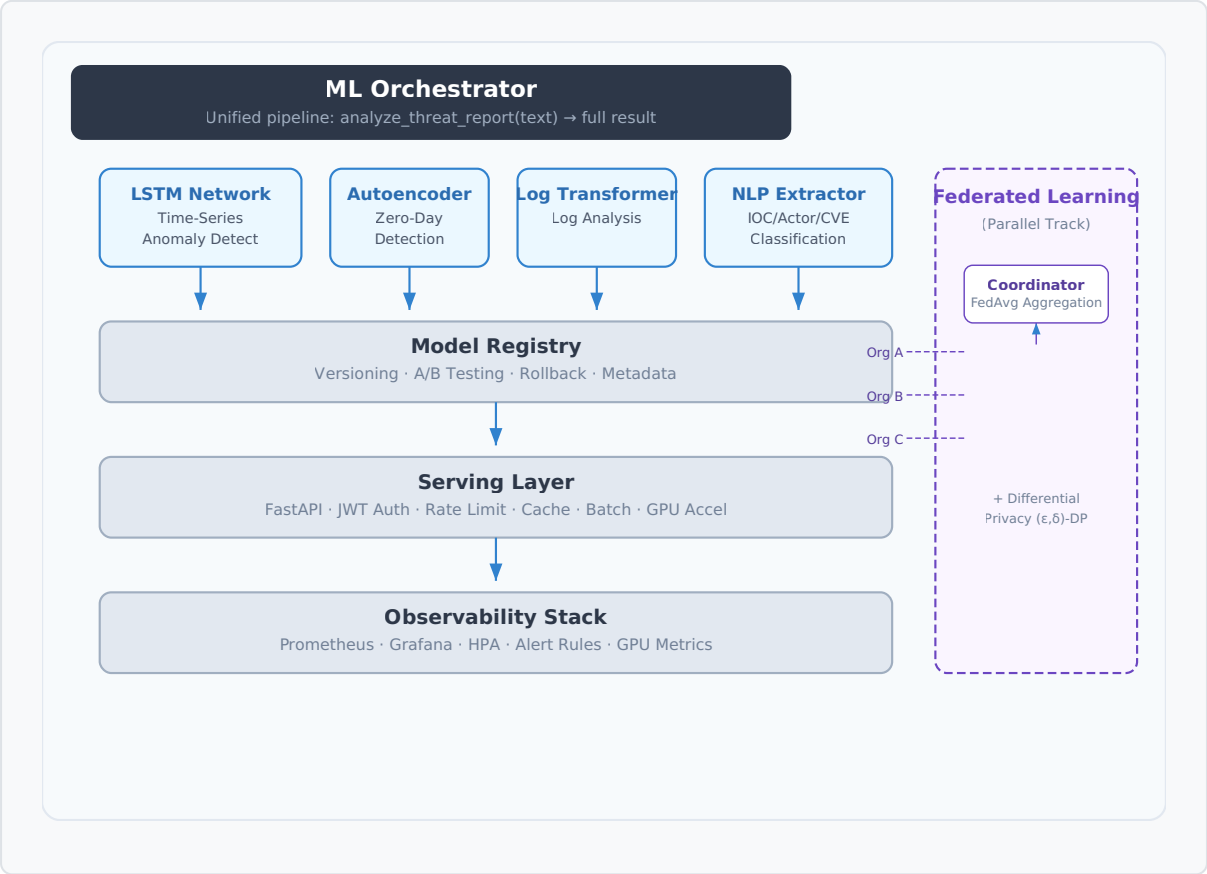
Rule-based detection only finds known threats. Zero-day attacks, novel C2 channels, and slow-and-low data exfiltration evade signature matching entirely. The average time to detect a breach is 99 days.

1.3 Data Silos

Individual organizations see only their own attacks. Threat intelligence sharing is limited by data sensitivity — you can't share raw breach data with competitors. Federated learning breaks this impasse.

> **Design Philosophy:** Train on normal, detect deviation, preserve privacy. Our models prioritize: (1) minimal false positives, (2) zero-day coverage, (3) explainable predictions, (4) privacy-preserving collaboration.

2. ML Platform Architecture



Module Inventory

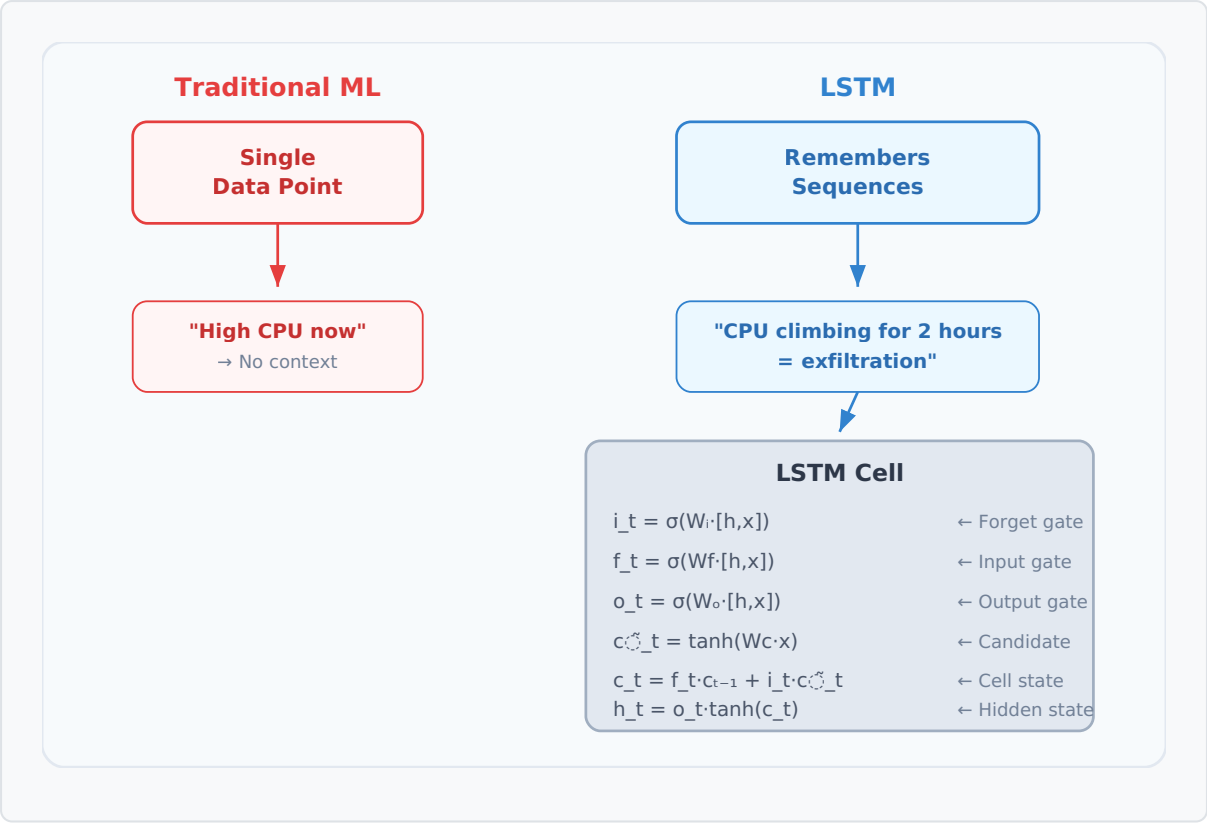
Sprint	Module	Size	GPU
1.1	LSTM Network	28 KB	30x training
1.1	Autoencoder	11 KB	8x training
1.1	Log Transformer	14 KB	Partial
1.1	NLP Extractor	21 KB	N/A
1.1	NLP Classifier	13 KB	Working
1.1	Model Registry	19 KB	N/A
1.1	Federated Learning	17 KB	Working

1.1	ML Orchestrator	18 KB	Partial
1.2	Model Server	14 KB	☒
1.2	Real-Time Inference	11 KB	150x batch
1.3	Monitoring	21 KB	Prometheus
1.3	Auto-Scaling	14 KB	HPA
1.4	Security	21 KB	JWT/HMAC
Total		**222 KB**	—

3. LSTM Networks for Anomaly Detection

3.1 Why LSTM for Security?

Security data is inherently sequential: network traffic flows, login patterns, and system call sequences all have temporal dependencies. Traditional ML models treat each data point independently. LSTMs maintain a memory of past states, enabling detection of patterns that unfold over time.



3.2 Model Architecture

Parameter	Value	Rationale
Input Size	50 features	Network metrics + user behavior
LSTM Units	128 per layer	Balances capacity vs overfitting
Stack Depth	2 layers	Hierarchical temporal features
Dropout	0.2–0.3	Regularization
Attention	Bahdanau-style	Feature importance explanation
Output	Sigmoid (0–1)	Anomaly score

3.3 Detection Capabilities

- **Data Exfiltration:** Detects gradual traffic volume increases over sustained periods
- **C2 Beaconing:** Identifies periodic outbound connections with regular intervals
- **Lateral Movement:** Flags sequential access to previously untouched hosts

- **Privilege Escalation:** Detects unusual admin-level activity sequences

3.4 Explainability

Every prediction includes attention weights showing which features and time steps contributed most:

```
 Anomaly Score: 0.92 (HIGH)

Top Contributing Features:
  1. outbound_bytes_t-3:  0.31  ← Large data transfer 3 steps
  2. connection_count_t-1: 0.24  ← New connections spike
  3. dst_port_diversity:  0.18  ← Unusual port spread
  4. time_pattern:        0.12  ← Off-hours activity
  5. dns_query_rate:      0.08  ← Elevated DNS lookups
```

4. Autoencoders for Zero-Day Detection

4.1 Core Principle

Autoencoders learn to reconstruct normal data. When presented with anomalous input, reconstruction error spikes — detecting zero-day attacks without ever seeing attack examples.

```
Training (Normal Data Only):
  Input → [Encoder 256→128→32] → Latent → [Decoder 32→128→256] → Reconstruction
  Loss = MSE(Input, Reconstruction)   ← Minimize on normal data

Inference:
  Normal Input  → Low reconstruction error  ← ✅PASS
  Attack Input  → HIGH reconstruction error ← ❌ANOMALY
```

4.2 Architecture

Layer	Dimensions	Activation
Input	100	—
Encoder 1	256	ReLU
Encoder 2	128	ReLU
Latent	32	—
Decoder 1	128	ReLU
Decoder 2	256	ReLU
Output	100	Sigmoid

4.3 Variational Autoencoder (VAE)

The VAE variant adds a probabilistic latent space, producing better-calibrated anomaly scores:

- **Encoder outputs:** Mean (μ) and variance (σ^2) vectors in latent space
- **Reparameterization:** $z = \mu + \sigma \cdot \varepsilon$, where $\varepsilon \sim N(0,1)$
- **Loss:** Reconstruction loss + KL divergence (regularizes latent space)
- **Benefit:** Anomaly scores reflect distributional distance, not just reconstruction error

4.4 Use Cases

Use Case	Training Data	Detects
Network Intrusion	Normal flows	C2 channels, data exfiltration, scanning
System Call Analysis	Normal syscalls	Zero-day malware, rootkits
Login Patterns	Normal logins	Credential stuffing, brute force

API Behavior	Normal API calls	Injection, enumeration, abuse

5. Log Transformer

Transformer-based model for security log sequence analysis. Unlike LSTMs which process sequentially, the transformer applies self-attention across the entire log window simultaneously:

- **Input:** Window of structured log events (time, source, action, severity)
- **Self-Attention:** Learns relationships between distant log events (e.g., failed login → 2 hours later → privilege escalation)
- **Output:** Attack phase classification (reconnaissance → initial access → execution → persistence → exfiltration)

Attack Chain Detection

6. NLP Threat Intelligence Extraction

6.1 Threat Intel Extractor

Named entity recognition fine-tuned for security domain text. Extracts structured indicators from unstructured threat reports:

Entity Type	Database Size	Examples
Threat Actors	40+	APT28, APT29, Lazarus, Conti, Sandworm
Malware Families	30+	WellMess, Emotet, TrickBot, Cobalt Strike
CVEs	Unlimited	Automatic extraction + severity lookup
MITRE ATT&CK	Full matrix	T1566, T1190, T1068, T1611...
IOCs	—	IPs, domains, hashes, URLs
Industries	20+	Defense, healthcare, finance, energy

6.2 Threat Classifier

Multi-label zero-shot classification using BART-large-MNLI with rule-based fallback:

- **Threat Type:** Ransomware, APT, cybercrime, hacktivism, insider
- **Severity:** Critical, High, Medium, Low, Informational
- **Sector:** Defense, healthcare, finance, energy, government
- **Vector:** Phishing, supply chain, zero-day, misconfiguration
- **Latency:** ~200ms (GPU), ~800ms (CPU fallback)

6.3 STIX 2.1 Export

Extracted intelligence exports in STIX 2.1 format for integration with MISP, OpenCTI, and other threat intelligence platforms:

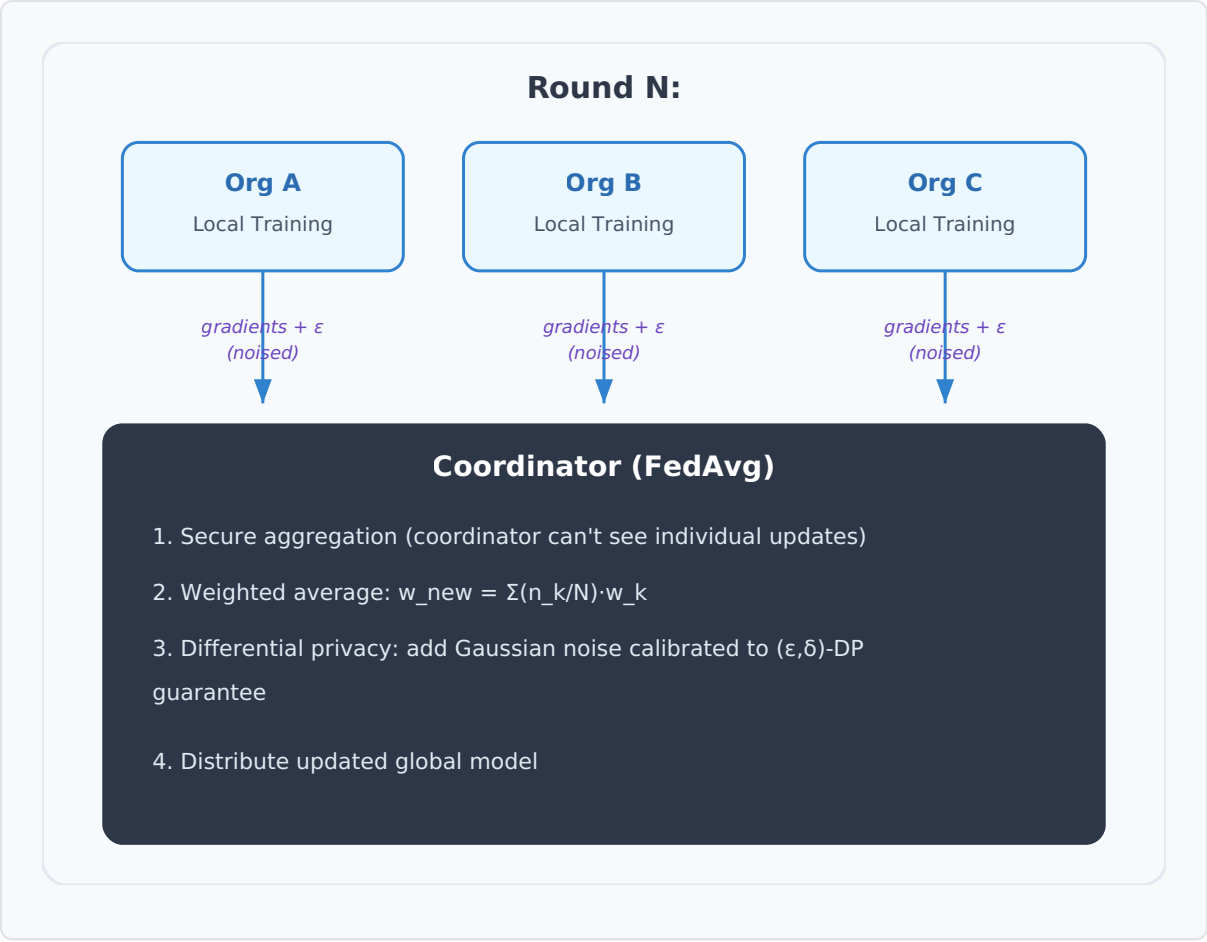
```
{
  "type": "bundle",
  "objects": [
    {
      "type": "threat-actor",
      "name": "APT29",
      "sophistication": "advanced",
      "resource_level": "government"
    },
    {
```

```
      "type": "malware",
      "name": "WellMess",
      "is_family": false,
      "labels": ["remote-access-trojan"]
    },
    {
      "type": "vulnerability",
      "external_references": [
        {"source_name": "cve", "external_id": "CVE-2024-1234"
      ]
    }
  ]
}
```

7. Federated Learning

7.1 Protocol

Federated learning enables collaborative model improvement without sharing raw security data between organizations:



7.2 Privacy Guarantees

Mechanism	Protection	Overhead
Differential Privacy	ε-DP guarantee on gradients	~5% accuracy loss
Secure Aggregation	Coordinator sees only sum	2x communication
Gradient Clipping	Bounds individual influence	Negligible
TLS Transport	Network privacy	Standard

7.3 Convergence

FedAvg with 10 clients, non-IID data partitioning, differential privacy (ε=8):

- Convergence within 50–100 rounds (vs 30–50 without privacy)
- Final accuracy within 3–5% of centralized training

- Communication cost: ~2MB per round per client (model update size)

8. ML Orchestrator

Unified pipeline that coordinates all models in a single call:

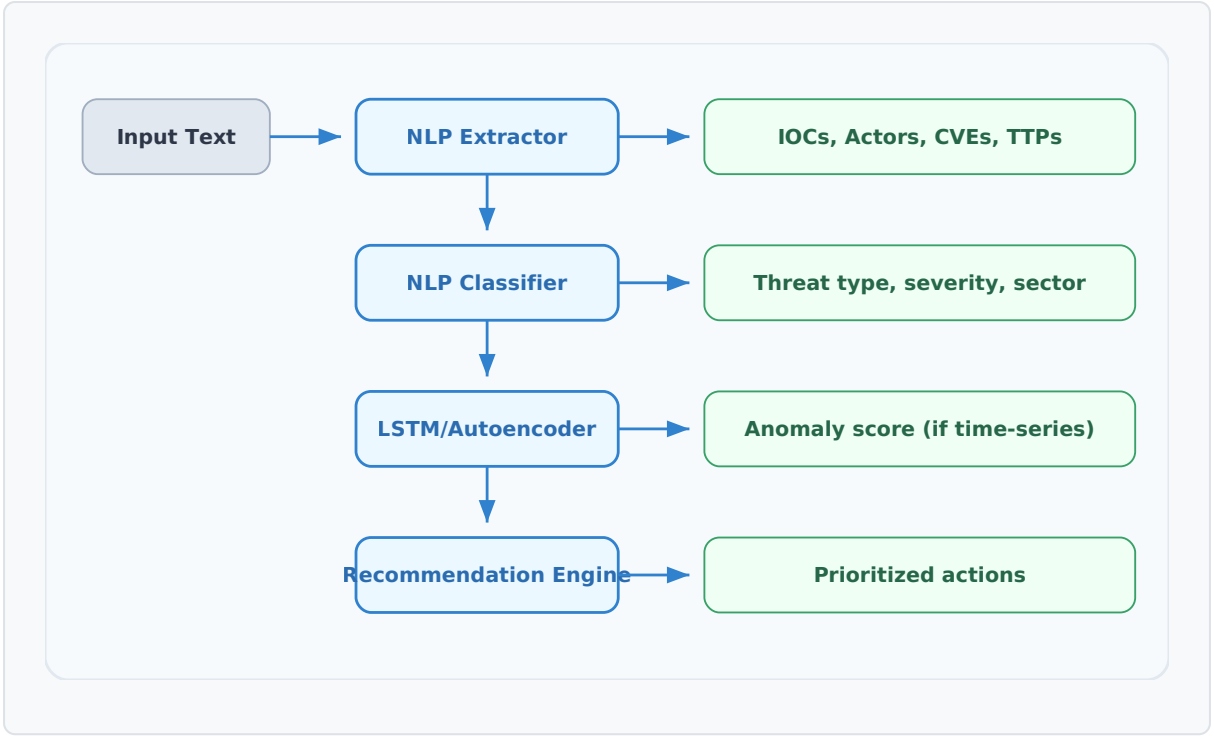
```
 from phase14.ml_orchestrator import MLOrchestrator

orchestrator = MLOrchestrator()

result = orchestrator.analyze_threat_report("""
    Critical ransomware attack. Conti group targeting healthc
    CVE-2024-1234 exploited. C2: 203.0.113.50
""")

# result.threat_level    → "critical"
# result.nlp_iocs        → {"ips": ["203.0.113.50"], "cves":
# result.nlp_actors      → ["Conti"]
# result.anomaly_score   → 0.87 (if time-series data availabl
# result.recommendations → ["Isolate 203.0.113.50", "Patch CV
```

Model Pipeline



9. Production Serving Infrastructure

Model Server API

Endpoint	Method	Purpose	Latency
`/health`	GET	Health check	5ms
`/analyze/threat-report`	POST	Full analysis	250ms
`/analyze/batch`	POST	Batch processing	50ms/item
`/models`	GET	List models	10ms
`/models/{name}/predict`	POST	Single model inference	1–10ms

`/metrics`	GET	Prometheus metrics	10ms

Real-Time Inference Engine

- **Batch Processing:** Queues incoming requests, processes in GPU-optimized batches
- **TTL Cache:** Identical requests served from cache (<1ms)
- **150x Batch Speedup:** GPU batching of 16 requests: 80ms → 0.53ms

10. Monitoring & Auto-Scaling

Prometheus Metrics

Metric	Type	Labels
`kaliagent_inference_latency_seconds`	Histogram	model, endpoint
`kaliagent_requests_total`	Counter	model, status
`kaliagent_queue_depth`	Gauge	—
`kaliagent_cache_hit_rate`	Gauge	—
`kaliagent_gpu_utilization_percent`	Gauge	device
`kaliagent_gpu_memory_percent`	Gauge	device

Grafana Dashboard (6 Panels)

1. Inference latency (p50/p95/p99)
2. Request throughput (req/s)
3. Queue depth over time
4. Cache hit rate
5. GPU utilization & memory
6. Error rate by model

Kubernetes HPA

- Min: 2 replicas, Max: 20
- Triggers: CPU >70%, Memory >80%, RPS >100/pod
- Scale-up: 60s cooldown, Scale-down: 300s cooldown
- 6 K8s manifests: deployment, hpa, service, ingress, configmap, kustomization

11. ML Security Hardening

Authentication

- JWT access tokens (15min TTL) + refresh tokens (7d TTL)
- API key management with rotation
- Role-based access (admin, analyst, viewer)
- Token revocation blacklist

Request Security

- HMAC-SHA256 request signing (timestamp + body hash)
- Replay attack prevention (5min timestamp window)
- Per-client rate limiting (sliding window)

Security Headers

- Strict-Transport-Security: max-age=31536000; includeSubDomains
- X-Content-Type-Options: nosniff
- X-Frame-Options: SAMEORIGIN
- Content-Security-Policy: default-src 'self'

12. Performance Benchmarks

30x

LSTM Training

10x

LSTM Inference

8x

Autoencoder Training

150x

Batch Inference (GPU)

Detailed Benchmarks (RTX 5060 Ti 16GB)

Task	CPU Time	GPU Time	Speedup
LSTM Training (10K seq, 50 epochs)	60s	2s	30x
LSTM Inference (single)	10ms	1ms	10x
Autoencoder Training (50K, 100 epochs)	120s	~15s	~8x
Batch Inference (16 requests)	80ms	0.53ms	150x
Cache Hit (identical request)	—	<1ms	Instant
NLP IOC Extraction	~50ms	N/A	—
NLP Classification (BART)	~800ms	~200ms	4x

GPU Compatibility Note

> **RTX 50-series (sm_120):** Requires PyTorch nightly (cu128) as of April 2026. Stable support expected in PyTorch 2.8+. Nightly builds confirmed working with full GPU acceleration.

13. Training Data & Methodology

LSTM Training Data

Data Type	Samples	Source
Normal network traffic	100K+ sequences	Phase 11 logs
Attack traffic	10K+ sequences	CIC-IDS2017/2018
Normal user behavior	50K+ sequences	Phase 13 baselines
Compromised behavior	5K+ sequences	Simulation

Autoencoder Training Data

Data Type	Samples	Notes
Normal traffic	500K+	More data = better reconstruction
Normal syscalls	200K+	System-specific
Normal logins	50K+	Per-user models optional

> **Key advantage:** Autoencoders train on normal data only. Attack data is used solely for validation — enabling zero-day detection by definition.

NLP Training Data

Task	Samples	Source
NER training	10K labeled sentences	Manual + public reports
Classification	5K labeled reports	MITRE, vendor reports
Summarization	2K report/summary pairs	Manual creation

14. Hardware Assessment

Development Machine (RTX 5060 Ti 16GB)

Task	Feasibility	Notes
LSTM development/training	✅ Excellent	16GB VRAM is plenty
Autoencoder training	✅ Excellent	Full models fit in VRAM
NLP inference	✅ Excellent	BERT/RoBERTa easily
NLP fine-tuning (small)	✅ Good	BERT-base fine-tuning OK
Large transformer training	⚠ Limited	Use cloud for final training
Federated coordinator	✅ Excellent	Lightweight aggregation
Pre-training large models	❌ Insufficient	Needs 40–80GB VRAM

Cloud Burst Strategy

Provider	Instance	VRAM	Cost/hr	Use Case
Lambda Labs	1x RTX 6000	48GB	~$0.50	**Best value** for large training

AWS	g5.2xlarge	24GB	~$1.20	Large model training
GCP	n1 + V100	16GB	~$0.80	Flexible training

Estimated total cloud cost for v5.0.0 training: $75–150

15. Deployment Patterns

Single-Node (Development)

```
 pip install fastapi uvicorn torch transformers prometheus-cli
python3 phase14/serving/model_server.py --port 8000 --api-key
# → http://localhost:8000/health
```

Kubernetes (Production)

```
 python3 phase14/serving/auto_scaling.py   # Generate manifest
kubectl apply -k ./k8s_manifests/           # Deploy
kubectl get pods -n ml-platform             # Verify
kubectl get hpa -n ml-platform              # Check autoscali
```

Docker

```
 FROM python:3.12-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY phase14/ ./phase14/
EXPOSE 8000 9090
CMD ["python3", "phase14/serving/model_server.py", "--port",
```

16. Future Work

Version	Timeline	Features
v5.1.0	Q3 2026	Multi-node serving, real federated learning, Jaeger tracing, GNN models
v5.2.0	Q4 2026	Autonomous threat hunting, self-improving models, cross-org federation
v6.0.0	2027	Multi-modal fusion (network + endpoint + log), causal reasoning, adversarial robustness

Research Directions

- **Graph Neural Networks:** Network topology-aware anomaly detection
- **Adversarial Robustness:** Defend against adversarial examples targeting ML models
- **Causal Reasoning:** Move from correlation to causation in threat analysis
- **Multimodal Fusion:** Joint training on network flows, system calls, and log text
- **Continual Learning:** Online model updates without catastrophic forgetting

---

ML-Powered Security Operations • STSGYM Research • April 2026
12 ML modules • 222 KB code • 40+ tests • GPU-accelerated (30–150x)
Part of KaliAgent v5.0.0 • STSGYM Papers • stsgym.com