

A Log Normalization and Feature Engineering Pipeline for Heterogeneous Healthcare IT Security Events

Obidinma Christian Alozie
Dept. of Mathematics/Computer Science
Clifford University
Owerrinta, Nigeria
(Corresponding author)

Chinagolum Ituma
Dept. of Computer Science
Ebonyi State University
Abakaliki, Nigeria

Nweso Emmanuel Nwogbaga
Dept. of Computer Science
Ebonyi State University
Abakaliki, Nigeria

Orih Nneka Bridget
Dept. of Planning, Research and
Statistics
David Umahi Federal University
Teaching Hospital
Uburu, Nigeria

Amadi Kelvin Emmanuel
Dept. of Computer Science
University of ICT Ofurekpe
Abakaliki, Nigeria

Nnona Jennifer Onyinyechi
Dept. of Computer Science
Ebonyi State University
Abakaliki, Nigeria

Abstract—Electronic Medical Record (EMR) systems in tertiary hospitals leave security evidence in three infrastructure layers: the database engine, the web and application server, and the operating system endpoints. Each layer records that evidence in its own syntax, at its own granularity and with its own idea of severity. In hospitals that cannot afford commercial security information and event management (SIEM) tooling, this heterogeneity, and not a shortage of data, is what stands in the way of machine-assisted incident detection. We built a log normalization and feature engineering pipeline that turns four operational years of live heterogeneous logs, collected between 2023 and 2026 at a Nigerian federal teaching hospital, into a single Common Event Schema. Four parsers share a three-pass architecture of raw parsing, rule-based classification and feature engineering. Two were written for this work: LIRA applies 25 deterministic rules to MySQL and MariaDB logs, and WELA applies 30 rules to Windows Event Log channels. Every label traces to a named rule and an observed line, so the corpus needed no manual annotation and stays auditable. The pipeline produced 187,446 label-free schema events and a 178,408-event unified corpus. On these, seven supervised classifiers and an Isolation Forest were trained with layer-specific feature sets of 8 to 33 features, and four algorithms were compared under one protocol. Macro F1 reached 1.0000 on the database downtime and endpoint code execution tasks and 0.9948, with perfect precision, on unauthorized database access. It fell to between 0.3584 and 0.5371 on the two web tasks. Explainability analysis showed that when a parser produces a deterministic feature geometry, all algorithms converge on the same boundary; when a parser-level confound merges two classes, no algorithm can separate them. Reported accuracy in log-based health informatics security is therefore, to a large degree, a property of the parser.

Keywords—common event schema; electronic medical records; feature engineering; healthcare cybersecurity; log normalization; machine learning

I. INTRODUCTION

Moving clinical documentation from paper to an Electronic Medical Record (EMR) system gathers patient histories, drug charts, laboratory results and administrative data into one electronic estate. The clinical gains are real and well documented. So is the exposure that comes with them. Every clinical workflow now depends on a handful of servers staying available and staying honest, and those servers make an obvious target [1], [2].

In a tertiary hospital the damage goes beyond lost data. An unplanned database outage takes the record away from every ward at once. A corruption event that nobody notices lets clinicians read and write against records that no longer match what was stored, sometimes for days. A credential compromise exposes protected health information and starts a statutory notification clock whether or not anyone is watching [3], [4].

Yet a hospital that wants to detect these events is rarely short of evidence. A conventional three-tier EMR already writes plenty of it. The database engine records start-ups, shutdowns, connection attempts, authentication failures and storage errors. The web server records requests and exceptions. The operating system records logons, privilege assignments, service installations, script execution and firewall changes. All of this piles up whether anyone reads it or not, and at the institution studied here it had piled up for four years.

The trouble is that the three streams have almost nothing in common. They share a timestamp. Beyond that, they differ in syntax, in what counts as a single event, in what they call an error, and in whether they name a user at all. A MySQL error line, an Apache access record and a Windows Event Log entry cannot simply be stacked into one feature matrix. Someone has

to decide what a severity of Warning in one vocabulary means in the other two, and that engineering choice carries through all the way to the reported metrics.

SIEM platforms exist to make exactly that decision, and the incident response literature tends to assume one is in place [5]–[7]. In the setting studied here, the assumption fails. A Nigerian federal teaching hospital running an EMR typically has no security operations centre, no commercial detection tooling and no dedicated analyst. Often it has one network administrator, who first hears of an incident when a clinician walks down to the server room to report it. Whatever detection capability such an institution gains has to be built from evidence it already owns, with software it can run on hardware it already has.

This paper reports the normalization and feature engineering layer of a larger design science project, which developed an enhanced incident response plan for the EMR at David Umahi Federal University Teaching Hospital (DUFUTH), Uburu, Ebonyi State. We report this layer on its own, and narrowly, because it is the part most likely to transfer. The response playbooks are tied to Nigerian regulation and to one hospital's clinical calendar. The parsers and the schema are not.

Three questions shaped the work. Can a deterministic parser produce labels good enough to train supervised detectors without anyone hand-annotating four years of logs? How should features be engineered at the database, web and endpoint layers, given that the signal at each has a different structure? And, most awkward for how results in this area are usually read, how much of the final performance belongs to the learning algorithm, and how much was already fixed by the parser before training began?

The third question had the sharpest answer. When the parser hands the model a deterministic mapping from one feature to one class, every tree-based algorithm finds the same boundary, and comparing them tells the reader nothing. When the parser leaves two classes overlapping in feature space, no algorithm separates them. On that evidence, a headline accuracy figure in this literature describes the parser at least as much as the model.

A. Literature Revi

Structured approaches to incident response have matured over the past decade. Ahmad *et al.* [8] studied how organisations build situation awareness during response and concluded that the binding constraint is the analytical processing of security events, not their availability. Naseer *et al.* [9] reached much the same place from a data analytics direction: response capability rests on an organisation's ability to turn raw telemetry into something a decision maker can act on. Both findings bear directly on this paper, since the pipeline described below tries to supply that processing where no analytical infrastructure exists at all.

Automation has attracted steady attention. Alam [10] reported a system that detected incidents in roughly three seconds and executed predefined responses in under five, with 95 per cent accuracy and no human in the loop. Aljahdali and

Alsulami [11] paired security orchestration tooling with a SIEM in a simulated corporate network and automated responses to phishing, port scanning and brute force activity. Automated vulnerability remediation defeated them, though, because the cloud-hosted orchestrator could not reach local scanners. Schlette *et al.* [12] surveyed threat intelligence from the response perspective and documented the fragmented representations that make cross-source correlation hard in the first place.

Work on formal playbooks runs alongside. Onwubiko and Ouazzane [13] proposed SOTER as a structured playbook for incident management. Shaked *et al.* [14] developed playbooks that bind response steps to operational constraints, and Akbari Gurabi *et al.* [15] set out requirements for playbook-assisted response and reporting. Staves *et al.* [16] addressed industrial control systems and critical national infrastructure, a domain that resembles healthcare in one respect: the security action itself can damage the process it protects.

Within healthcare, He *et al.* [17] extended the NIST life cycle by folding threat intelligence into a six-stage proactive procedure, using the National Health Service WannaCry incident as their case, and closed by recommending integration with SIEM, orchestration platforms and a security operations centre. Fysarakis *et al.* [18] reported PHOENIX, a European resilience framework that uses AI-assisted orchestration for continuity and recovery. Thompson [19] set out how to design a HIPAA-compliant security operations centre.

A pattern runs through all of this work. The capability is shown in a corporate network, in simulation, or in a well-funded health system in a developed country, and it assumes staff and infrastructure that the institutions this study addresses do not have. Seen from such an institution, the recommendations that close these papers simply restate the problem. O'Neil *et al.* [20] catalogued socio-technical barriers to organisational response, including false-positive fatigue, poor tool usability and siloed analyst teams, and Ojo *et al.* [21] derived a response team model from ten security operations centres. Neither considers an institution with no security operations centre to begin with.

The healthcare incident reporting literature, for its part, is almost entirely about clinical patient safety rather than cybersecurity. That tradition has produced careful work on under-reporting and feedback failure, but it does not extend to detecting security events against the record system itself. Studies of EMR adoption in Nigerian and West African settings confirm both the pace of digitisation and the absence of any matching security capability [22], [23], and analyses of the Nigerian regulatory environment set out the duties any response capability has to meet [24].

The gap this paper addresses sits underneath all of that work. Detection models, correlation rules and playbooks all take a normalized event stream as given. How that stream gets built from real hospital logs is usually treated as an implementation detail: mentioned in a sentence, or handled by adopting a vendor schema wholesale. We think that is a mistake. The normalization step sets the ceiling for everything

built on top of it, and the results reported here suggest it shapes the published numbers more directly than the choice of classifier does. Data quality frameworks in clinical decision support make a related argument about how downstream inference depends on upstream representation [25]. This study therefore treats normalization as the object of investigation.

II. MATERIALS AND METHODS

A. Setting and Data Collection

The case institution runs a conventional three-tier EMR. Tier one is a MySQL 5.7 and MariaDB 10.x database server holding all structured patient and administrative data; when it crashes, electronic clinical documentation stops across the hospital and wards fall back to paper. Tier two is an Apache HTTP server that delivers the PHP application layer and handles authentication, sessions and the link between clinical browsers and the database. Tier three is the Windows estate of workstations and domain controllers, whose event logs capture logon activity, PowerShell execution, SMB file access and firewall changes.

Raw log files covering 2023 to 2026 were exported by the hospital system administrator under controlled access and moved to the research workstation on encrypted removable media. Written institutional consent was obtained through the hospital Research and Ethics Unit. One point matters for the ethics of the study: the MySQL error and general logs contain no patient-identifiable data. They record server-level events, connection attempts, engine state changes and query errors, and never the content of a record. Access controls and audit trails of the kind Aldosari [26] describes governed the transfer throughout.

B. Pipeline Architecture

All four parsers share a three-pass architecture. We chose it so that the origin of every field in the output could be stated exactly, which matters more here than elegance does.

1) *Pass one, raw parsing*: Each line is matched against format-specific regular expressions that extract only what the format actually guarantees. For MySQL that is the timestamp, the severity level (Note, Warning or Error) and the full message text. For the Windows Event Log the equivalents come from the canonical tuple of channel, event identifier, keywords and event data. Nothing is interpreted in this pass. A line that matches no expression is kept and marked unparsed instead of being dropped, since silent loss at this stage cannot be seen downstream.

2) *Pass two, rule-based classification*: A deterministic rule engine assigns an incident label. The rules come from vendor documentation for each format and from the hospital's clinical operating procedures, and each carries a match confidence between 0 and 1. LIRA, the database parser, holds 25 rules. WELA, the endpoint parser, holds 30, grouped into authentication, privilege and policy, and network and service categories. Both were written to prefer a false negative to a false positive. Whatever they emit becomes ground truth for the models, and a wrong label costs more than a missing one.

3) *Pass three, feature engineering*: The last pass computes derived features that carry operational context the raw line lacks: temporal decomposition, ordinal severity encoding, a composite severity score, boolean state indicators and categorical context codes. Section II-F describes these layer by layer. Keeping the passes separate makes the pipeline auditable. Every label in the final dataset traces back to a named rule and a specific line, and that is what lets the artefact survive a regulatory inspection as well as score well. Table I summarises the four parsers and Table II the schema they write to.

TABLE I.
THE FOUR NORMALIZATION PARSERS AND THEIR RULE ENGINES

Source log	Tool	Rule engine	Key detections	Output
MySQL/MariaDB error and general log	LIRA v2.2	25 deterministic rules	Crash recovery sessions, data corruption, failed authentication, after-hours database access, unknown hosts, privilege misuse	8 CSV datasets and a text report
Apache HTTP access log	log_parser.py	Composite anomaly score	Brute force (401 floods), credential exposure in request path, suspicious paths, backup and restore metadata, bot detection	access_log_dataset.csv
Apache HTTP error log	error_parser.py	AH code lookup and severity scoring	PHP database authentication failure, DNS failure, TLS certificate mismatch, unclean shutdown, process life-cycle events	error_log_dataset.csv
Windows Event Log (EVTX)	WELA v1.0	30 deterministic rules	Failed logon (4625), brute force episodes, SMB storms, privilege escalation, service installation, account management, process creation	Per-channel CSVs, combined dataset and text report

TABLE II.
THE COMMON EVENT SCHEMA FIELD SET

Field	Type	Role in the normalized record
ces_hour	Integer 0 to 23	Hour of the event, decomposed from the source timestamp
ces_day_of_week	Integer 0 to 6	Monday is 0; captures weekday and weekend operating patterns

Field	Type	Role in the normalized record
ces_is_weekend	Binary	Weekend indicator
ces_is_after_hours	Binary	Event outside clinical operating hours (before 07:00, or at or after 21:00)
ces_severity_norm	Float 0 to 1	Source severity level, normalized across the four severity vocabularies
ces_confidence_norm	Float 0 to 1	Confidence of the parser rule that assigned the label
ces_auth_fail_flag	Binary	Authentication failure indicator, harmonised across MySQL, Apache and Windows semantics
ces_suspicious_flag	Binary	Suspicious pattern indicator raised by the source rule engine
ces_req_rate_norm	Float 0 to 1	Per-address request rate for web sources; 0 for non-web sources
ces_source_lira	Binary	One-hot source identity: database layer
ces_source_access	Binary	One-hot source identity: Apache access log
ces_source_error	Binary	One-hot source identity: Apache error log
ces_source_evtx	Binary	One-hot source identity: Windows endpoint

Source identity is one-hot encoded rather than label encoded so that the Isolation Forest treats each source equally at every random split. Free-text fields are excluded from the schema by design.

C. RA: Database Layer

LIRA, the Log Intelligence and Response Analyzer, is a Python command-line tool that parses, classifies and enriches MySQL and MariaDB events. Its 25 rules resolve each event into one of eight types: SYSTEM_DOWNTIME for crash-related sequences such as InnoDB crash recovery, Aria recovery, buffer pool rebuilds and unclean shutdown; PLANNED_MAINTENANCE for clean shutdowns started by an administrator or a scheduled job; DATA_CORRUPTION for table corruption, tablespace errors and binary log corruption; then ABORTED_CONNECTION, ACCESS_DENIED, DNS_FAILURE, SUSPICIOUS_REVIEW (for after-hours access and host or user blocking) and CLEAN_OPERATION for ordinary status messages.

The split between the first two types is the clearest case in the pipeline of engineering driven by clinical reasoning. At first glance a crash and a planned maintenance window look the same in the log: a service stops, and later it starts. What separates them is that a crash leaves a recovery sequence behind on the next start-up and a planned shutdown does not. An error in one direction is merely annoying. An error in the other is dangerous. If a genuine crash is labelled maintenance, nothing escalates, the on-call administrator is never paged, the business continuity protocol is never activated, and ward staff go on believing the outage was expected. Recall on SYSTEM_DOWNTIME is therefore the metric that matters at this layer, and the is_crash_recovery boolean exists only to carry that distinction.

LIRA also resolves unauthorized access through five signature classes: rogue device access, credential abuse, brute force, DNS hijacking and after-hours access. MySQL records host and user identity inconsistently, so both are folded into ordinal status codes instead of being kept as free text. The host_status code separates an unknown lookup from a registered baseline device, an unregistered IPv6 address, a missing reverse DNS resolution, a device present but not approved, and a direct address connection that bypasses

hostname resolution entirely. That last state is the most suspicious of the six. We ordered the code deliberately so that suspicion rises with the integer, which lets a tree split on the value meaningfully. The user_status code separates an undetermined context from a known authenticated user, an anonymous connection and a failed authentication.

D. eb Layer

The two Apache parsers face a different problem from LIRA, and from each other. The access log records every request and is dense and high in volume. The error log is sparse and fires only when something noteworthy happens, and its messages come from four abstraction layers at once: the operating system process model, the PHP runtime, the TLS stack and the server core. All application logic that touches the database runs through PHP, so database authentication failures and connection pool exhaustion flood the stream, making up roughly 80 per cent of entries across the four years.

The access parser emits 33 features in six families: four temporal, five HTTP request attributes, four traffic rate indicators, three user agent signals, ten security flags and seven ordinally encoded application context codes. The error parser emits only 17, and the smaller number is deliberate. Almost all the discriminative signal in an Apache error line sits in three categorical fields: the subsystem, the severity level and, above all, their interaction, which is kept as an explicit module_level feature. More features on that stream add noise.

E. WELA: Endpoint Lay

WELA, the Windows Event Log Analyser, is the endpoint counterpart of LIRA. Its 30 rules fire on structured matchers over the canonical tuple of channel, event identifier, keywords and event data, and its three passes are discovery, temporal context derivation and labelling. It emits eight features: the label-encoded event_id and channel_key, severity and severity_weight, and four temporal fields.

WELA splits the estate by channel into two disjoint problems. The Security, System and network service channels carry authentication, privilege and network activity, resolved

into SYSTEM_BASELINE, NORMAL_AUTH, BRUTE_FORCE, UNAUTHORIZED_LOGON, PRIVILEGE_ESCALATION, FIREWALL_EVENT and SMB_OPERATION. The PowerShell, PowerShell Classic, WMI Activity, Windows Defender and Code Integrity channels carry code execution and malware activity, resolved into BENIGN_SYSTEM_ACTIVITY, POWERSHELL_ABUSE, WMI_LATERAL_MOVEMENT, MALWARE_DETECTED and APPLICATION_FAULT.

One implementation detail in WELA deserves recording, because it decides a result reported later. For the code execution channels, the ground truth label is applied during parsing instead of in a separate pass, and this was necessary. The labelling rules have to match the event data payload in place, for instance to spot an encoded command argument or an anti-malware scan interface initialisation failure inside a PowerShell script block event, and the payload is dropped from the schema row immediately afterwards. Defer the label, and the evidence needed to assign it is already gone.

F. Features by Lay

The three layers needed different feature sets, and the way the sets change between them is itself instructive. Table III lists the families and their sizes. At the database layer a 15-feature core was settled for the downtime task: four temporal fields, three severity and confidence signals, five boolean state indicators and three encoded categorical context fields. Several columns were left out on purpose, for two separate reasons. The raw message text was withheld because keeping it would let the model memorise exact log strings without learning anything general. The rule description and sublabel were withheld because they derive from the target label and would have leaked it outright. Administrative fields such as the log path and server identifier went too, being identifiers with no signal in them.

Three connection-layer booleans covering aborted connections, access denials and DNS failures were added for the corruption task, bringing the set to 18. They contribute exactly nothing to corruption detection. We expected this when we added them, and the explainability analysis later confirmed it. We kept them anyway, because at the unauthorized access task the after-hours and access-denied indicators, flat at zero in the first two tasks, climb to third and fourth rank. This is the practical argument for a shared schema over per-task feature selection. Optimise each task on its own and those columns get pruned one task before they are needed.

G. Class Imbalan

Every layer of the corpus is imbalanced, and in each case the imbalance reflects how the hospital actually runs. The database engine writes many lines during crash recovery and

few during a planned maintenance window. Planned maintenance happens perhaps once a quarter. Unplanned crashes, driven largely by the local electricity supply, happen far more often. The imbalance ratio was computed as

$$IR = |N_{maj}| / |N_{min}| \quad (1)$$

and ranged from 3.00:1 at the corruption task to 48,710:1 at the web error task, as Table IV reports. SMOTE was applied to the training partition only, always after a stratified 80/20 split at a fixed seed of 42, so that the held-out partition keeps the original distribution and measures deployment conditions. Synthetic instances are interpolated linearly between a seed instance and one of its five nearest minority neighbours:

$$x_{new} = x_i + \lambda (\tilde{x}_i - x_i), \lambda \sim U(0, 1) \quad (2)$$

Where the expansion factor is extreme, we flag the result instead of presenting it at face value. The worst case in this corpus grew a four-instance class to 155,872, a factor of 38,968. Four seed points give SMOTE a very small simplex to work in. The perfect test recall later observed on that class, which has exactly one test instance, is therefore an artefact, and we report it as one. Without that qualification the number would mislead.

H. Evaluation Protoc

Seven supervised classifiers were trained, one per layer-specific task, alongside an Isolation Forest that scores every schema event on a continuous anomaly scale. All seven followed the same protocol, so that differences across layers reflect the data and not the experimental design. Four algorithms were compared at each task: Logistic Regression, Decision Tree, Random Forest and XGBoost. Each was assessed by five-fold stratified cross-validation on the balanced training partition and then evaluated once on the held-out partition in its original distribution. Six metrics were recorded: accuracy, precision, recall, macro F1, weighted F1 and ROC AUC. SHAP analysis was run on the selected model. Every experiment fixed the seed at 42 and serialised the partitions, so the same splits reproduce across runs.

The Isolation Forest used 200 estimators, a contamination parameter of 0.05, a maximum sample size of 256 and the same seed. Its output is remapped from the library decision function onto the unit interval and appended to the schema row as a fourteenth feature. In its training corpus, non-database sources were down-sampled to 30,000 events each, since the database stream would otherwise dominate the path-length geometry. Thirty thousand per source is the smallest budget at which the forest output stopped moving between bootstrap replicates. An optional temporal mode trains on 2023 to 2025 and tests on 2026 alone, which simulates deploying the models at the start of the final year and letting them meet unseen data.

TABLE III.
ENGINEERED FEATURE SETS BY EMR LAYER

Layer and task	Parser	Features	Feature families
Database: downtime	LIRA	15	4 temporal; 3 severity and confidence; 5 boolean state indicators; 3 encoded categorical context fields

Layer and task	Parser	Features	Feature families
Database: corruption	LIRA	18	The 15 above plus 3 connection-layer booleans (aborted connection, access denied, DNS failure)
Database: unauthorized access	LIRA	18	As above; the after-hours and access-denied indicators become primary discriminators at this task
Web: access traffic	log_parser	33	4 temporal; 5 HTTP request attributes; 4 traffic rate indicators; 3 user agent signals; 10 security flags; 7 ordinal application context codes
Web: server errors	error_parser	17	Temporal decompositions; business hours and weekend flags; content flags; categorical encodings of module, level and their interaction; continuous severity score
Endpoint: security events	WELA	8	Label-encoded event_id and channel_key; severity and severity_weight; 4 temporal features
Endpoint: code execution	WELA	8	As above, over a disjoint channel taxonomy; label applied in place at parse time

TABLE IV.
CORPUS COMPOSITION AND CLASS IMBALANCE BY TASK

Task	Events	Test set	IR	Imbalance character
Database downtime	6,198	1,033	77.26:1	Planned maintenance is quarterly and brief; crashes are frequent and generate long recovery sequences
Database corruption	2,838	473	3.00:1	Moderate; both classes are genuine emergencies
Unauthorized access	68,646	13,730	5.09:1	About one in six access anomalies is a confirmed breach; the rest are suspicious but benign
Web access traffic	148,384	29,677	23.99:1	Nine classes; backup and restore traffic alone is 37.7 per cent of the test set
Web server errors	280,930	46,822	48,710:1	Thirteen classes; the dominant DNS failure class is 83.2 per cent of the test set; the smallest class has four instances
Endpoint security	151,516	25,253	42.4:1	System baseline is generated continuously by policy refreshes and scheduled tasks
Endpoint code execution	104,800	17,467	39.9:1	WMI activity is rare and PowerShell abuse rarer in a hardened estate

IR is the imbalance ratio in the training partition before synthetic oversampling. Test partitions keep the original distribution throughout.

III. RESULTS AND DISCUSSION

A. The Corp

The pipeline reduced four years of live logs from three infrastructure layers to a single interface. It produced 187,446 label-free schema events for unsupervised training, drawn from all four sources, and a 178,408-event unified corpus for the cross-source meta-classifier. A per-source cap of 60,000 events was imposed on the unified corpus, with anything above the cap uniformly sub-sampled under the fixed seed, to stop the database stream from dominating the learned representation simply by being the loudest.

That a corpus of this size could be assembled at all, at an institution with no security tooling whatsoever, is the first result, and in some ways the most widely applicable one. The evidence was already there. What was missing was the layer that made it readable.

B. Performance Across the Sta

Table V gives the selected model and its held-out performance at each task. The pattern is not monotonic, and its shape is where the argument of this paper begins. Performance is near perfect at the database layer, falls apart at the web layer, and recovers to near perfect at the endpoint layer.

At the database downtime task, all four algorithms reached a macro F1 and ROC AUC of 1.0000 on held-out data, tracing

the upper left corner of the ROC space with an average precision of 1.0000 at every recall threshold. When four different algorithms agree that completely, the honest reading is that the task was already solved before any of them saw it.

The corruption task separated them slightly. Logistic Regression achieved a perfect macro F1, while each of the three tree-based models produced one false positive. All four achieved a recall of 1.0000, which is the number that matters clinically: no corruption event was routed to the business continuity protocol when it needed the data recovery protocol. That misrouting is the most dangerous error available in this task, because restarting a corrupted database does not repair it. It simply restarts a corrupted database.

Unauthorized access was the first task where nothing scored perfectly. Logistic Regression and Random Forest tied on macro F1 at 0.9948, with recall between 0.9827 and 0.9836. Logistic Regression was selected on the tie because its test set precision was perfect. It missed 39 breaches out of 2,256, a false negative rate of 1.73 per cent, and raised not one false alarm. The missed events clustered in a grey zone where the composite severity score falls in an overlapping interval and the secondary evidence is incomplete. We treat that as a threshold problem, since no algorithm changed it.

Cross-validation and test rankings diverged here in a way worth noting. Random Forest led cross-validation on the balanced training partition, placing first in every fold, then

drew level with the linear model on the imbalanced test partition. Tree thresholds calibrated on balanced data generate false positives that a linear hyperplane simply does not. The same divergence recurs at the web layer, where it does far more damage.

C. Where It Breaks: the Web Lay

At the web access task the best model managed a macro F1 of 0.3584, against a random baseline of 0.111 across nine classes. At the web error task the best reached 0.5371 across thirteen. Neither is usable without supervision. We identified two causes, and both sit in the parser.

The first is a semantic confound, almost embarrassing in its simplicity. The backup engine deployed at the hospital writes credentials into request paths as a matter of routine. So the credential exposure flag reads 1 for legitimate backup traffic and for a genuine credential exposure alike. All four algorithms failed almost completely on that class, and the selected model achieved zero recall. Under the applicable data protection regulation, a confirmed exposure is a reportable breach with a 72-hour notification duty attached, so this is a blind spot no institution can live with [24]. A better classifier cannot fix it either. Two classes that the feature space has merged stay merged however much model capacity is applied. The fix is a supplementary control, specified in Section IV.

The second cause is what happens when extreme imbalance meets synthetic oversampling. At the web error task every algorithm reaches about 98.9 per cent accuracy, which sounds impressive until one notices that a single class covers 83.2 per cent of the test set. Macro F1 sits at roughly half that figure.

Every algorithm shows a drop of between 28 and 34 per cent from cross-validation to test. The ensembles memorise boundaries that SMOTE drew in the balanced folds and then fail to transfer them. A single shallow tree transfers best, which is why one was selected even though Random Forest won every cross-validation metric on paper.

D. here It Works Again: the Endpoint

Performance recovers sharply at the endpoint. The selected Decision Tree reached a macro F1 of 0.9999 at the security task with a ROC AUC of 1.0000. At the code execution task, the Decision Tree, Random Forest and XGBoost tied at a perfect 1.0000, with zero false negatives across all five classes. Cross-validation showed essentially no fold-to-fold variance for the trees and essentially no gap to the test set.

The SMOTE distribution shift that wrecked both web tasks is absent here for a structural reason. The classification boundary at this layer is the event identifier integer itself. The geometry is deterministic, so any tree recovers the same mapping from any fold partition. A small detail confirms the mechanism. Interpolating over label-encoded categorical features produces non-integer values that have to be rounded back to the nearest category, and the deterministic identifier-to-class mapping means the model lands on the right boundary regardless of that rounding noise. The noise cannot move a boundary that the category itself defines.

Logistic Regression behaves very differently across the two endpoint tasks, reaching only 0.5869 at the security task but 0.8345 at code execution. The explanation lies in which feature carries the signal, the subject of the next section.

TABLE V.
HELD-OUT PERFORMANCE OF THE SELECTED MODEL AT EACH LAYER

Task	Selected model	Macro F1	Recall	ROC AUC	Selection basis
Database downtime	Logistic Regression	1.0000	1.0000	1.0000	All four algorithms tied at 1.0000; fastest to retrain
Database corruption	Logistic Regression	1.0000	1.0000	1.0000	Only model with no false positive; trees each produced one
Unauthorized access	Logistic Regression	0.9948	0.9827	0.9983	Tied on macro F1; selected for perfect test set precision
Web access traffic	Decision Tree	0.3584	—	0.8867	Best test F1 and AUC; shallow pruning avoids boundary memorisation
Web server errors	Decision Tree	0.5371	—	—	Best test F1; transfers where ensembles do not; six-second training
Endpoint security	Decision Tree	0.9999	0.9999	1.0000	Recall of at least 0.9982 on every security class
Endpoint code execution	Decision Tree	1.0000	1.0000	1.0000	Three-way tie at 1.0000; zero false negatives on all five classes

Recall is reported for the binary and two-class tasks. For the multi-class web tasks, the per-class recall pattern is discussed in the text instead of being summarised by a single figure.

E. What the Parser Had Already Decid

SHAP was computed on the selected model at every task. Table VI summarises the dominant features, and the results all point the same way.

At the downtime task, the composite severity score and the start-up recovery context dominate. That is exactly what the rules encode, since crash events score at or above 7 on the

composite scale and planned shutdowns at or below 2. At the corruption task, `is_crash_recovery` dominates across all four algorithms, and in the depth-one Decision Tree it carries the entire discriminative weight, with a mean absolute SHAP value of 0.5000. The three connection-layer features contribute precisely zero: corruption separates from crash on database engine signals alone. At unauthorized access, the composite severity score and the ordinal severity level are co-dominant,

with the after-hours indicator arriving as a new third-ranked contributor, picking up the tendency of unauthorized access to happen outside working hours.

At the endpoint security task, the event identifier is the top feature for all three tree-based algorithms, at roughly 50.9 per cent of total importance for the selected tree, followed by the channel key and severity weight. Every temporal feature sits below 0.5 per cent; they survive only because the parser needs them for its after-hours rule and because the schema is shared. The dominance of the event identifier reflects a vendor practice of assigning one identifier to one event type. The near-perfect result therefore validates the parser, more than the algorithm.

At code execution the hierarchy inverts. The channel key is the top feature for every algorithm, at 64.5 per cent of importance for Logistic Regression and 49.5 per cent for the selected tree. That inversion explains the linear model's different performance noted above, since a channel-dominated geometry is far more linearly separable than an identifier-dominated one. It also has a maintenance consequence. The channel taxonomy is stable across Windows Server releases, so the code execution detector generalises to new event identifiers without retraining, whereas the identifier-driven security detector would need its encoder extended.

Put the five together and the conclusion is hard to avoid. At every task with near-perfect performance, one engineered feature produced by the parser carries the weight, and the four algorithms agree because they are all recovering the same deterministic boundary. At every task that collapses, the cause is a defect in the feature space the parser built. The operative variable here is parser engineering.

F. Implications for Reported Accuracy

A macro F1 of 1.0000 in a log-based health informatics detector tells a reader very little on its own. It might indicate a well-engineered pipeline. It might equally indicate a task whose geometry was deterministic before any learning happened.

From the metric alone the two cannot be told apart, and this may account for some of the very high figures in this literature.

Results of this kind should therefore be reported with an explainability analysis attached and with the feature geometry stated outright, so a reader can tell which of the two has actually been shown. Where a result depends on an extreme synthetic expansion factor, the factor belongs next to the metric itself.

A second point concerns transfer. Because performance depends on local feature geometry, an institution adopting this architecture should re-derive its labelling rules and thresholds against its own logs instead of importing trained models. The parsers travel. The models do not.

The third point is economic, and for the setting this study addresses it matters most. What produced these results was 55 deterministic rules and a shared 13-field schema. There was no commercial security tooling, no security operations centre and no server infrastructure beyond what the hospital already ran. Once rule-governed parsing makes it analysable, the evidence inside such an institution is enough to support usable detection at two of three layers and to pin down exactly what is failing at the third. That is a smaller claim than the literature usually makes, and a far more affordable one.

G. Limitations

Three limitations bound these findings. The ground truth is parser-derived, not independently adjudicated, so parser bias cannot be quantified from inside the study and the true ceiling of log-based detection at each layer stays unknown. The corpus comes from one institution; the layer technologies are conventional, but the class distributions carry local fingerprints, not least an electricity supply that shapes the downtime class more than any adversary does. And validation is based on test harnesses and simulation, without adversarial testing. An attacker who knew the rule set could shape activity around it, and that has not been tested.

TABLE VI.
 DOMINANT ENGINEERED FEATURES BY TASK FROM SHAP ANALYSIS

Task	Dominant feature(s)	Interpretation
Database downtime	severity_score; startup_context	Matches the operational definitions encoded in the rules; crash events score at or above 7.0, planned shutdowns at or below 2.0
Database corruption	is_crash_recovery	Carries the entire discriminative weight in the depth-one tree (mean absolute SHAP 0.5000); the three connection-layer features contribute exactly zero
Unauthorized access	severity_score; severity; is_after_hours	The after-hours indicator, near zero in the two preceding tasks, rises to third rank; access denied ranks third or fourth across algorithms
Endpoint security	event_id (50.9% of tree importance); channel_key; severity_weight	Reflects the vendor practice of one identifier per event type; all temporal features below 0.5 per cent, kept for schema uniformity
Endpoint code execution	channel_key (49.5% tree, 64.5% linear); severity_weight	Reverses the preceding hierarchy; the stable channel taxonomy means this detector generalises to new identifiers without retraining

IV. CONCLUSION

This study set out to make four years of heterogeneous EMR security telemetry machine-readable, in a hospital with no security tooling to read it with. The answer was four parsers

sharing a three-pass architecture, 55 deterministic rules across the two built for this work, and a 13-field Common Event Schema in which every label traces back to a named rule and a specific line. It yielded 187,446 label-free schema events and a

178,408-event unified corpus, and it needed no manual annotation of the corpus at any point.

Detection built on that corpus performed unevenly, and the unevenness is the finding. It was near perfect at the database and endpoint layers. It collapsed at the web layer, for two reasons that can be named precisely, neither of them algorithmic.

When the parser hands a model a deterministic feature geometry, the choice of learning algorithm stops mattering, because the discriminative work was finished before training started. That changes how this literature should be read. An accuracy figure in log-based health informatics security describes a parser as much as a model, and it should be published with an explainability analysis beside it and with any extreme synthetic expansion factor stated openly. It also means the lasting products of this work are the parsers and the schema, more than the seven trained models. Anyone adopting the architecture should re-derive their rules and thresholds against their own logs instead of importing models unchanged.

Four lines of future work follow. The first is external validation across several Nigerian tertiary hospitals. The second is independently adjudicated ground truth, so that parser bias becomes measurable and the real ceiling of log-based detection at each layer can be established. The third is adversarial evaluation of the rule sets themselves. The fourth is the supplementary control needed to break the credential exposure confound at the web layer, the one outstanding defect with a statutory duty attached. The recommended control excludes the backup engine's own service account and request namespace from the credential exposure rule before the flag is set, so that the class boundary is restored at the parser instead of being asked of the classifier.

ACKNOWLEDGMENT

The authors thank the management of David Umahi Federal University Teaching Hospital, Uburu, and its Research and Ethics Unit for granting controlled access to live EMR server logs, and the hospital information technology staff who supported the data collection, often outside their own working hours.

DECLARATIONS

Conflict of interest: The authors declare that they have no conflict of interest.

Data availability: The log corpus analysed in this study contains institutional infrastructure records and is not publicly released. The parser rule sets and the Common Event Schema specification are available from the corresponding author on reasonable request.

Ethical consideration: Access to EMR server logs was granted by the Research and Ethics Unit of David Umahi Federal University Teaching Hospital, Uburu, with written institutional consent. The logs analysed record server-level events only and contain no patient-identifiable data.

Author contributions: Obidinma Christian Alozie conceptualised the study, designed and implemented the LIRA and WELA parsers and the Common Event Schema, conducted the model training and explainability analysis, and drafted the manuscript. Chinagolum Ituma supervised the research and took part in result interpretation, manuscript editing and critical revision. Orih Nneka Bridget contributed to data collection and institutional coordination at the case hospital, and to validating the incident labelling rules against clinical operating procedures. Nweso Emmanuel Nwogbaga [contribution to be added]. Amadi Kelvin Emmanuel [contribution to be added]. Nnona Jennifer Onyinyechi [contribution to be added]. All authors reviewed and approved the final manuscript.

REFERENCES

- [1] S. Ghafur, E. Grass, N. R. Jennings, and A. Darzi, "The challenges of cybersecurity in health care: The UK National Health Service as a case study," *The Lancet Digital Health*, vol. 1, no. 1, pp. e10–e12, 2019, doi: 10.1016/S2589-7500(19)30005-6.
- [2] I. Keshta and A. Odeh, "Security and privacy of electronic health records: Concerns and challenges," *Egyptian Informatics Journal*, vol. 22, no. 2, pp. 177–183, 2021, doi: 10.1016/j.eij.2020.07.003.
- [3] N. V. Verde, L. Gitto, and A. Pasquini, "Cybersecurity threats in healthcare: Compromised data, privacy violations, and patient safety risks," *International Journal of Environmental Research and Public Health*, vol. 20, no. 5, Art. no. 4043, 2023, doi: 10.3390/ijerph20054043.
- [4] N. Basil, S. Ambe, C. Ekhatior, and E. Fonkem, "Health records database and inherent security concerns: A review of the literature," *Cureus*, vol. 14, no. 10, Art. no. e30168, 2022, doi: 10.7759/cureus.30168.
- [5] R. Sendelj and I. Ognjanovic, "Cybersecurity challenges in healthcare," in *Achievements, Milestones and Challenges in Biomedical and Health Informatics*. Amsterdam, The Netherlands: IOS Press, 2022, pp. 190–202, doi: 10.3233/SHTI220951.
- [6] A. Zamfiroiu and R. C. Sharma, "Cybersecurity management for incident response," *Romanian Cyber Security Journal*, vol. 4, no. 1, pp. 69–75, 2022, doi: 10.54851/v4i1y202208.
- [7] F. Luh and Y. Yen, "Cybersecurity in science and medicine: Threats and challenges," *Trends in Biotechnology*, vol. 38, no. 8, pp. 825–828, 2020, doi: 10.1016/J.TIBTECH.2020.02.010.
- [8] A. Ahmad, S. B. Maynard, K. C. Desouza, J. Kotsias, M. T. Whitty, and R. L. Baskerville, "How can organizations develop situation awareness for incident response: A case study of management practice," *Computers & Security*, vol. 101, Art. no. 102122, 2021, doi: 10.1016/j.cose.2020.102122.
- [9] H. Naseer, S. B. Maynard, and K. C. Desouza, "Demystifying analytical information processing capability: The case of cybersecurity incident response," *Decision Support Systems*, vol. 143, Art. no. 113476, 2020, doi: 10.1016/j.dss.2020.113476.
- [10] T. Alam, "Automated incident response system," *International Journal of Novel Research and Development*, vol. 10, no. 4, pp. c533–c536, 2025, doi: 10.30574/ijnrd.2025.10.4.IJNRD2504288.
- [11] A. O. Aljahdali and R. Alsulami, "Streamlining threat response and automating critical use cases with Security Orchestration, Automation and Response (SOAR)," *Journal of Digital Security and Forensics*, vol. 2, no. 1, pp. 36–57, 2025, doi: 10.29121/DigiSecForensics.v2.i1.2025.45.
- [12] D. Schlette, M. Caselli, and G. Pernul, "A comparative study on cyber threat intelligence: The security incident response perspective," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2525–2556, 2021, doi: 10.1109/COMST.2021.3117338.
- [13] C. Onwubiko and K. Ouazzane, "SOTER: A playbook for cybersecurity incident management," *IEEE Transactions on Engineering Management*, vol. 69, no. 6, pp. 1–21, 2020, doi: 10.1109/TEM.2020.2979832.
- [14] A. Shaked, Y. Cherdantseva, P. Burnap, and P. Maynard, "Operations-informed incident response playbooks," *Computers & Security*, vol. 134, Art. no. 103454, 2023, doi: 10.1016/j.cose.2023.103454.

- [15] M. Akbari Gurabi *et al.*, "Requirements for playbook-assisted cyber incident response, reporting and automation," *Digital Threats: Research and Practice*, 2024, doi: 10.1145/3688810.
- [16] A. Staves, T. Anderson, H. Balderstone, B. Green, A. Gouglidis, and D. Hutchison, "A cyber incident response and recovery framework to support operators of ICS and critical national infrastructure," *International Journal of Critical Infrastructure Protection*, vol. 37, Art. no. 100505, 2022, doi: 10.1016/j.ijcip.2021.100505.
- [17] Y. He, L. Maglaras, A. Aliyu, and C. Luo, "Healthcare security incident response strategy: A proactive incident response (IR) procedure," *Security and Communication Networks*, vol. 2022, Art. no. 2775249, 2022, doi: 10.1155/2022/2775249.
- [18] K. Fysarakis *et al.*, "PHOENIX: A European cyber resilience framework with artificial-intelligence-assisted orchestration, automation and response capabilities for business continuity and recovery, incident response, and information exchange," in *Proc. IEEE Int. Conf. Cyber Security and Resilience (CSR)*, Piscataway, NJ, USA, 2023, doi: 10.1109/csr57506.2023.10224995.
- [19] E. C. Thompson, *Designing a HIPAA-Compliant Security Operations Center: A Guide to Detecting and Responding to Healthcare Breaches and Events*. Berkeley, CA, USA: Apress, 2020, doi: 10.1007/978-1-4842-5608-4.
- [20] A. O'Neil, A. Ahmad, and S. B. Maynard, "Cybersecurity incident response in organisations: A meta-level framework for scenario-based training," in *Proc. Australasian Conf. Information Systems (ACIS)*, 2021, doi: 10.48550/arXiv.2108.04996.
- [21] C. Ojo, E. A. Osoko, J. N. Okolo, and M. Jaji, "Incident response: A structured model from detection to containment and recovery," *World Journal of Advanced Research and Reviews*, vol. 24, no. 1, pp. 1401–1407, 2024, doi: 10.30574/wjarr.2024.24.1.3148.
- [22] C. D. Akwaowo, A. E. Ekpenyong, A. M. Ekanem, E. A. Umoh, I. Asangansi, M. E. Ekpenyong, and E. Henriksen, "Electronic medical records in a resource-constrained setting: Perceptions and experiences of clinicians in South-South Nigeria," *Frontiers in Digital Health*, vol. 4, Art. no. 823312, 2022, doi: 10.3389/fdgth.2022.823312.
- [23] A. G. Awobele, A. O. Ogundipe, and A. B. Bello, "Cybersecurity in healthcare systems: Safeguarding electronic health records (EHRs) and medical devices against emerging cyber threats," *World Journal of Advanced Research and Reviews*, vol. 25, no. 2, pp. 2249–2262, 2025, doi: 10.30574/wjarr.2025.25.2.0592.
- [24] A. AbdulKareem, "Establishing a robust legal framework for EMR implementation: Regulatory compliance issues in Nigeria's tertiary healthcare institutions," *Journal of Health Law and Policy in Africa*, vol. 3, no. 1, pp. 14–29, 2024.
- [25] S. G. Johnson, S. Speedie, G. Simon, V. Kumar, and B. L. Westra, "Application of a data quality framework to clinical decision support," *Applied Clinical Informatics*, vol. 14, no. 2, pp. 235–244, 2023, doi: 10.1055/s-0043-1764262.
- [26] B. Aldosari, "Cybersecurity in healthcare: New threat to patient safety," *Cureus*, vol. 17, no. 5, Art. no. e83614, 2025, doi: 10.7759/cureus.83614.