GRAPH-AWARE LLM FOR WINDOWS LOGONS WITH A CLOSED-LOOP GUARDED DETECTION AGENT

JPCERT/CC Shusei Tomonaga

Shusei Tomonaga

- Incident Response Group at JPCERT/CC
- Malware analysis, forensic investigation.
- Posts on malware analysis and technical findings are available on our blog and GitHub.

http://blogs.jpcert.or.jp/

https://github.com/JPCERTCC/

Problem

Attackers abuse **legitimate authentication mechanisms**, leaving few forensic traces.

Blue Teams still struggle to tell **normal vs. malicious logons** in Windows environments.

Windows Event Logs are abundant - **but insights are rare**.

Motivation

Windows Event Logs were not designed for intrusion detection.

Active Directory generates millions of noisy events every day.

A method is needed to detect attack activity within this noise.

Goal of This Presentation

Identify compromised accounts and suspicious logon activity in Windows environments using AI

Primary focus

- post-compromise behavior, lateral movement, credential abuse

What we detect

- high-risk logon patterns that warrant investigation and containment

Takeaways

Understand a practical method to extract suspicious activity from Windows Event Logs for incident response without relying solely on signature-based detection.

Design and deploy a graph-oriented preprocessing pipeline that compresses large-scale Windows Event Logs into an authentication graph and feature set that an LLM can efficiently consume.

Apply techniques for automated LLM-based log analysis while constraining hallucinations.

Presentation Topics

1. Architecture Overview
2. Evaluation
3. Demonstration
4. Further Use of AI

Related Work: LLMs in Log Analysis

Common usage: LLM-based Analysis Survey


```xml
<EventID>4624</EventID>
<Version>0</Version>
<Level>0</Level>
<Task>12544</Task>
<Opcode>0</Opcode>
<Keywords>0x8020000000000000</Keywords>
<TimeCreated SystemTime="2025-11-07T08:18:27.2110477Z" />
<EventRecordID>2680039</EventRecordID>
<Correlation />
<Execution ProcessID="572" ThreadID="692" />
<Channel>Security</Channel>
<Computer>WIN-WFBHIBE5GXZ.example.co.jp</Computer>
<Security />
</System>
<EventData>
<Data Name="SubjectUserSid">S-1-0-0</Data>
<Data Name="SubjectUserName">-</Data>
<Data Name="SubjectDomainName">-</Data>
<Data Name="SubjectLogonId">0x0</Data>
<Data Name="TargetUserName">WIN11_64JP_08$</Data>
<Data Name="TargetDomainName">EXAMPLE</Data>
<Data Name="LogonType">3</Data>
<Data Name="LogonProcessName">Kerberos</Data>
<Data Name="AuthenticationPackageName">Kerberos</Data>
<Data Name="WorkstationName" />
<Data Name="LogonGuid">{dc0fd12b-b48e-7eed-89b0-05c4692771df}</Data>
<Data Name="KeyLength">0</Data>
<Data Name="ProcessId">0x0</Data>
<Data Name="IpAddress">192.168.16.108</Data>
<Data Name="IpPort">50894</Data>
</EventData>
</Event>
```


Summary

A successful Windows logon event (Event ID 4624) was recorded on WIN-WFBHIBE5GXZ.example.co.jp at 2025-11-07T08:18:27. The target account EXAMPLE¥WIN11_64JP_08$ authenticated using Kerberos via a network logon (LogonType 3) from source IP 192.168.16.108.

Related Work: LLMs in Log Analysis

Akhtar et al. (2025) surveyed LLM-based techniques for event log analysis[1].

Existing Approach

- Anomaly Detection
- LLM-based Analysis Survey
- LLM-based Log Parsing

Problem

LLMs show promise for text-based log parsing and few-shot or fine-tuned setups, but **scalability** and **real-time** analysis remain underexplored.

Our Approach

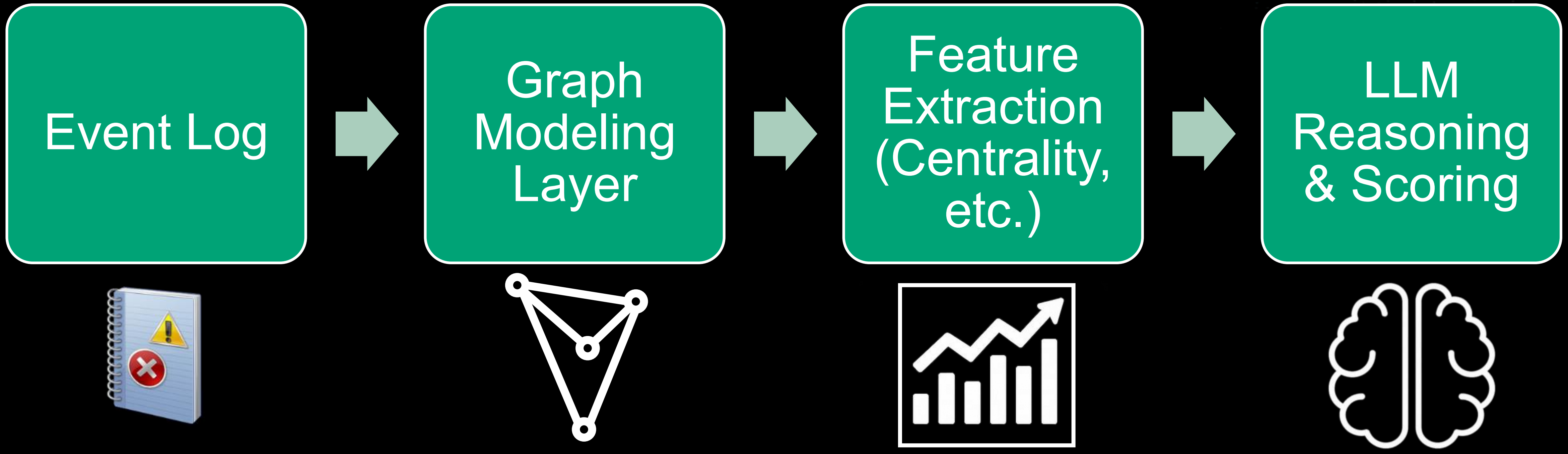
Core Idea

We integrate **graph-based relational modeling** with **LLM reasoning** to detect malicious logons that traditional methods miss.

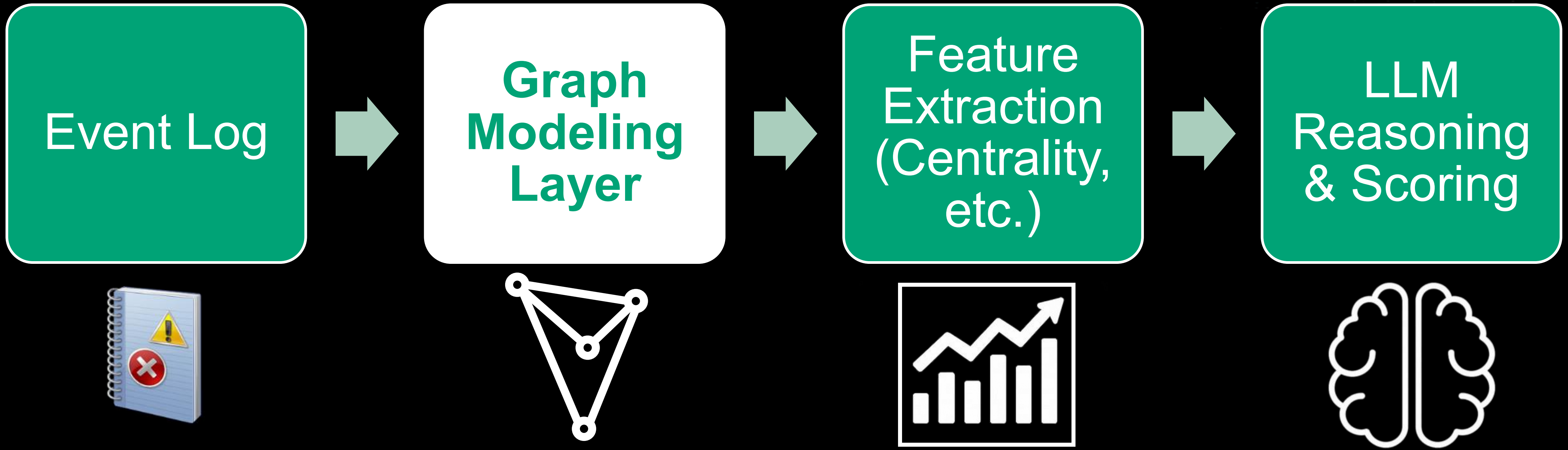
Key Advantages

- Captures **user-host relationships** in Windows network(between users and hosts).
- Scales efficiently across **millions of log events**.

Architecture Overview

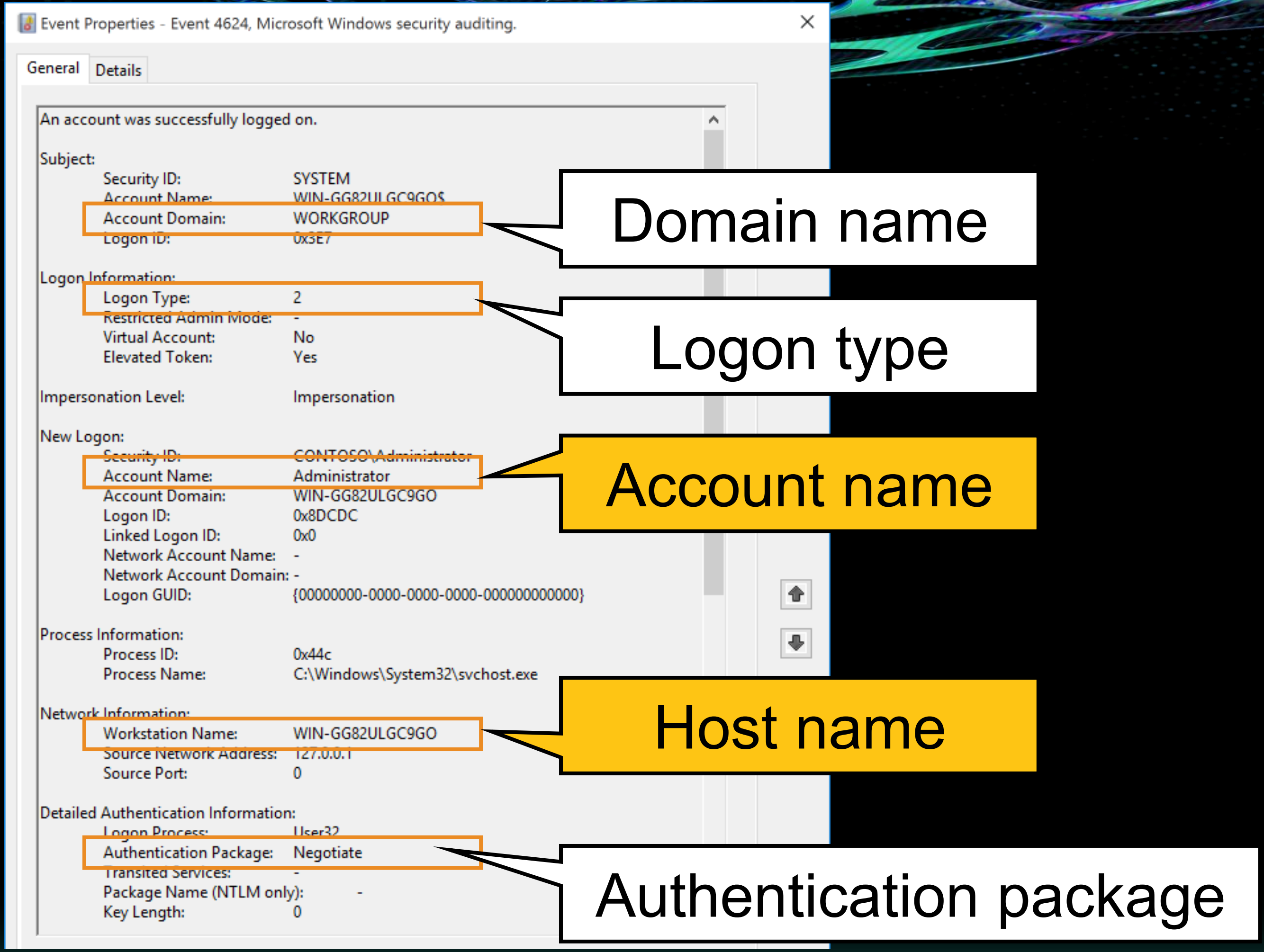


Architecture Overview



Event Log

Basic patterns of logs related to logon



How Many Logon Event IDs Exist?

Event ID	Detail	Event ID	Detail
4624	Successful logon	**4770**	Renew kerberos service ticket
4625	Logon failure	**4771**	Failed kerberos pre-authentication
4634	Account logoff	**4772**	Failed kerberos authentication
4647	User initiated logoff	**4773**	Failed kerberos service ticket
4648	Logon using explicit credentials	**4774**	Logon map
4672	Assign special privileges	**4775**	Failed logon map
4675	Filter SID	**4776**	NTLM authentication
4768	Kerberos authentication(TGT Request)	**4777**	Failed NTLM authentication
4769	Kerberos service ticket (ST Request)	**4964**	Assign special privilege groups

Windows provides many logon-related Event IDs, but not all of them are equally useful for intrusion detection.

Which Event IDs Matter?

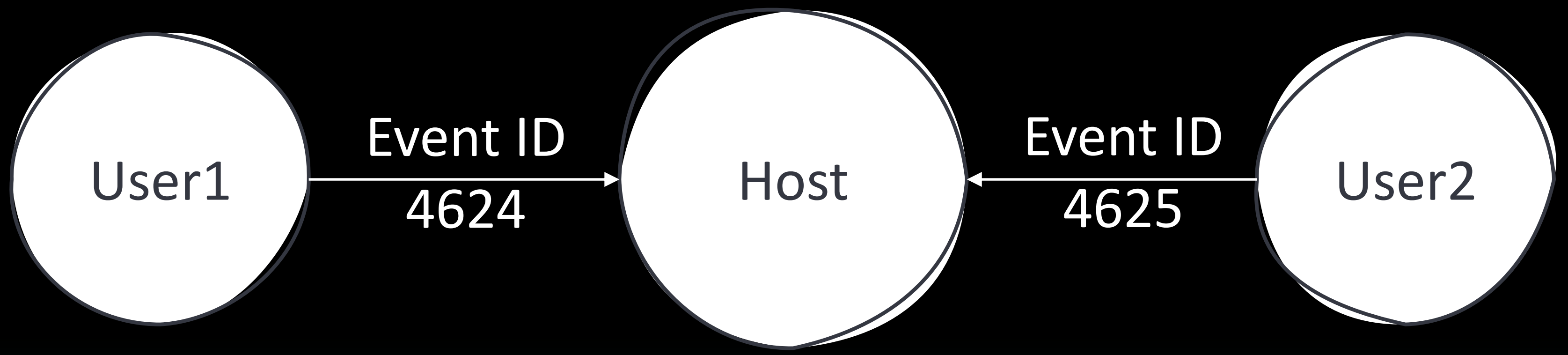
Event ID	Detail	Event ID	Detail
4624	Successful logon	**4770**	Renew kerberos service ticket
4625	Logon failure	**4771**	Failed kerberos pre-authentication
4634	Account logoff	**4772**	Failed kerberos authentication
4647	User initiated logoff	**4773**	Failed kerberos service ticket
4648	Logon using explicit credentials	**4774**	Logon map
4672	Assign special privileges	**4775**	Failed logon map
4675	Filter SID	**4776**	NTLM authentication
4768	Kerberos authentication(TGT Request)	**4777**	Failed NTLM authentication
4769	Kerberos service ticket (ST Request)	**4964**	Assign special privilege groups

We focus on **six Event IDs** that provide the strongest authentication signals for detecting malicious logons.

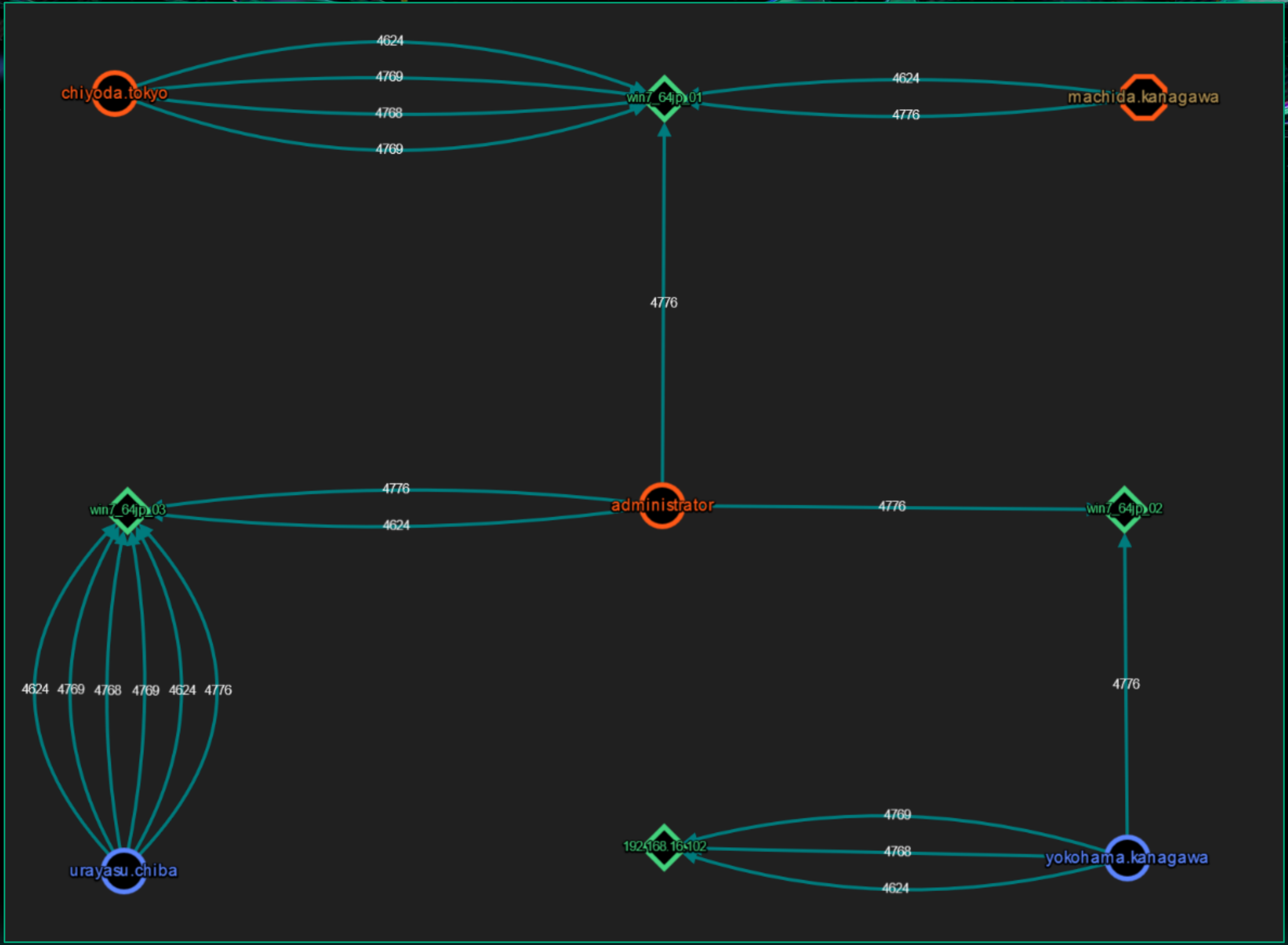
Graph Modeling for Event Log

Connect the account name and host from each logon event ID.

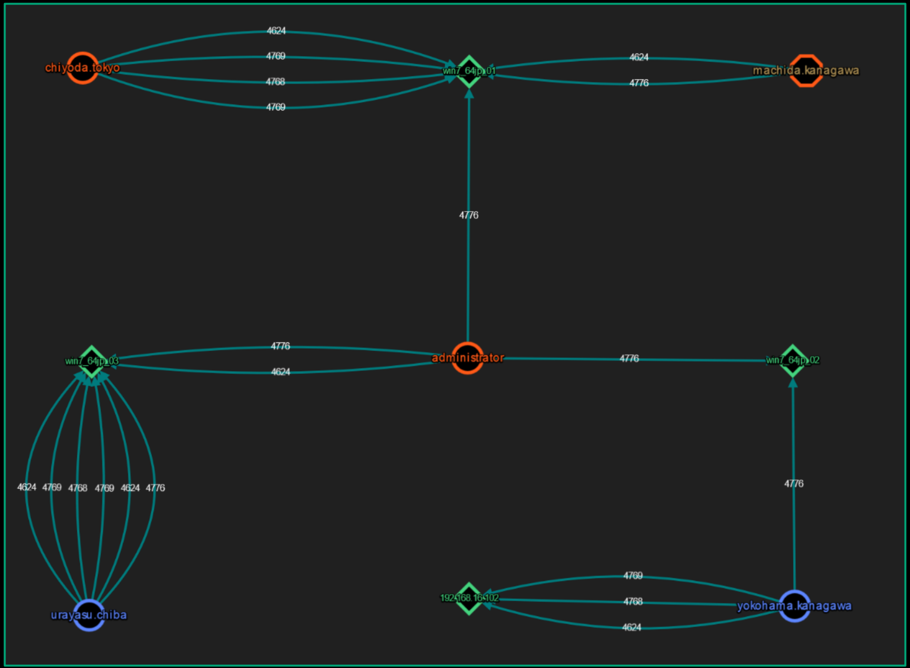
It is easy to check which account name was used to logon to a host.



Graph Based Event Log



Graph-Aware Context



Graph to JSON


```json
[
  {
    "u.user": "sysg.admin",
    "i.hostname": "192.168.16.109",
    "i.IP": "win10_64jp_09",
    "e.id": 4624,
    "e.logintype": 3,
    "e.authname": "Kerberos",
    "e.servicename": "-",
    "e.ticketencryptiontype": "-",
    "e.status": "-"
  },
  {
    "u.user": "sysg.admin",
    "i.hostname": "192.168.16.109",
    "i.IP": "win10_64jp_09",
    "e.id": 4624,
    "e.logintype": 3,
    "e.authname": "Kerberos",
    "e.servicename": "-",
    "e.ticketencryptiontype": "-",
    "e.status": "-"
  },
  {
    "u.user": "sysg.admin",
    "i.hostname": "192.168.16.109",
    "i.IP": "win10_64jp_09",
    "e.id": 4769,
    "e.logintype": 0,
    "e.authname": "-",
    "e.servicename": "WIN-WFBHIBE5GXZ$",
    "e.ticketencryptiontype": "AES256-CTS-HMAC-SHA1-96",
    "e.status": "-"
  }
]
```

The Advantages of Graph Modeling

Relationships matter

Representing users and hosts as nodes and logons as edges exposes hidden connections among entities.

Structural awareness

Attackers often use a single account to access multiple hosts — this forms distinct graph patterns detectable via metrics like degree or centrality.

Anomaly amplification

Graph structures highlight abnormal connectivity, enabling LLMs to focus on truly suspicious behavior instead of isolated log lines.

Scalability

Aggregating millions of log entries into a compact relational graph reduces data volume and computation cost dramatically.

Scalability - From Millions of Events to a Manageable Graph

Collapse raw logons into unique (user, host) edges over a time window; store counts & time stats instead of every line.

Raw Events
50MB

>

Graph Data
1MB

Token cost scale for LLM with **nodes/edges**, not with **all events**.

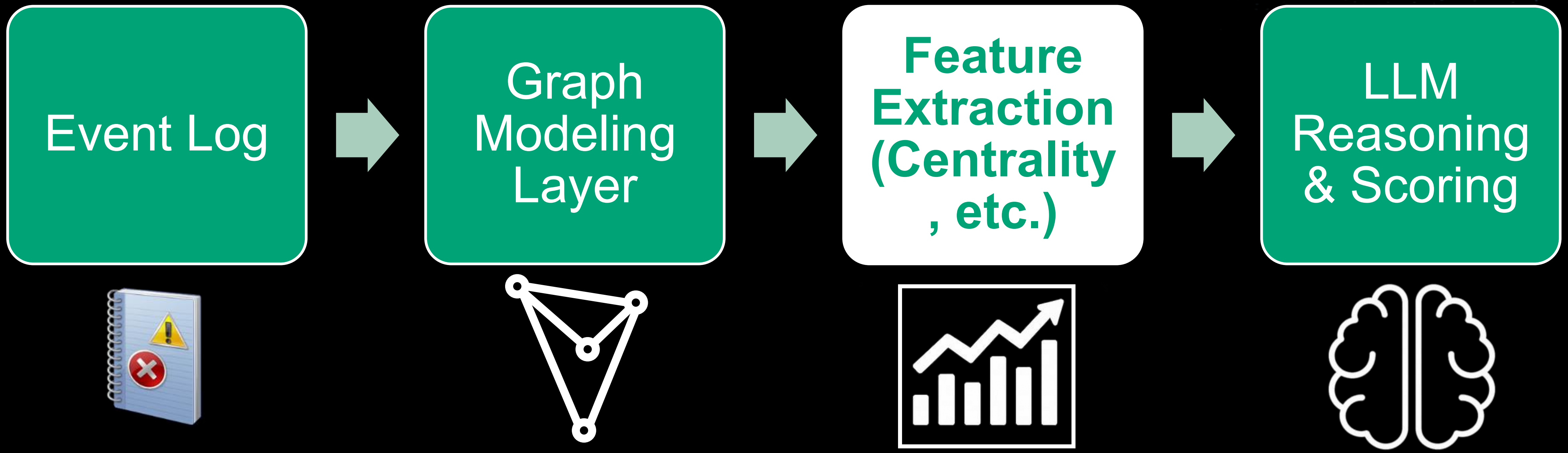
Cost of “LLM-only” Raw Event Log Analysis

Sample case

Event log file size	600MB
Input volume	1,230,000 events (469 tokens per event)
Total input tokens	469 token × 1,230,000 events ≈ **576,870,000 tokens** (about **576.9M**)
API cost (input only)	576M / 1M × $1.75* ≈ **$1,010 per analysis run** * GPT-5.2 costs $1.75 per 1M input tokens

Context limit problem: GPT-5.2 context window is **400,000 tokens**
476M / 400k ≈ **1,400+ chunked requests** (before adding instructions and output)

Architecture Overview



Feature Extraction

Why Add Features on Top of Graphs

Pre-extracted features reduce tokens, inference latency, and improve precision and explainability.

- Rank by features.
  - Used as data to limit the scope of analysis for the LLM.
- Combine **graph context + numeric evidence** for stable decisions.

Rank using PageRank

PageRank is an algorithm used by Google Search to rank websites in their search engine results.

$$PR(A) = (1-d) + d\sum_{i=1}^{n}\frac{PR(T_i)}{C(T_i)}$$

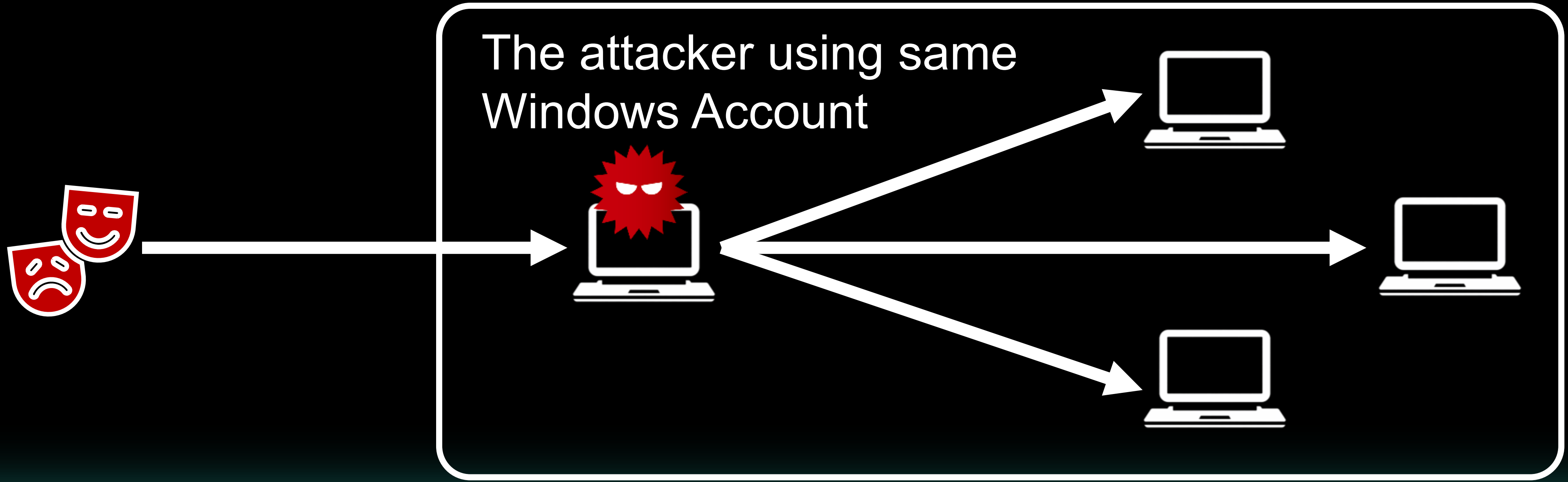
PageRank Concept

- Important websites are linked by many other websites.
- Websites linked by important websites tend to have higher importance.

Why use PageRank

Features of Lateral Movement

- Lateral movement looks like **fan-out**: one compromised account touches many hosts.
- **PageRank** turns that pattern into a robust, scalable score.



PageRank with Hidden Markov Model

Improve PageRank accuracy by combining it with a Hidden Markov Model (HMM).

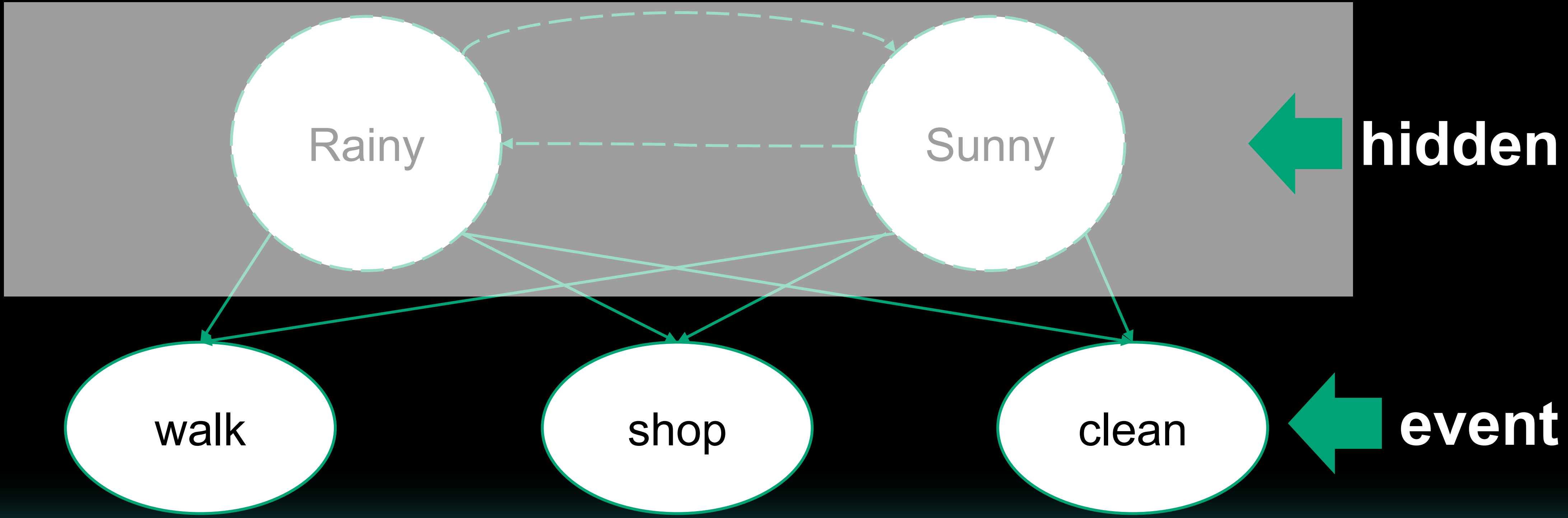
$$PR(A) = (1 - d(A)) + d(A) \sum_{i=1}^{n} \frac{PR(T_i)}{C(T_i)}$$

What's HMM

- The HMM is one of the most important machine learning models in speech and language processing.
- This model is the state transition of the past in hidden state, and the event is modeled from the output result of that state.

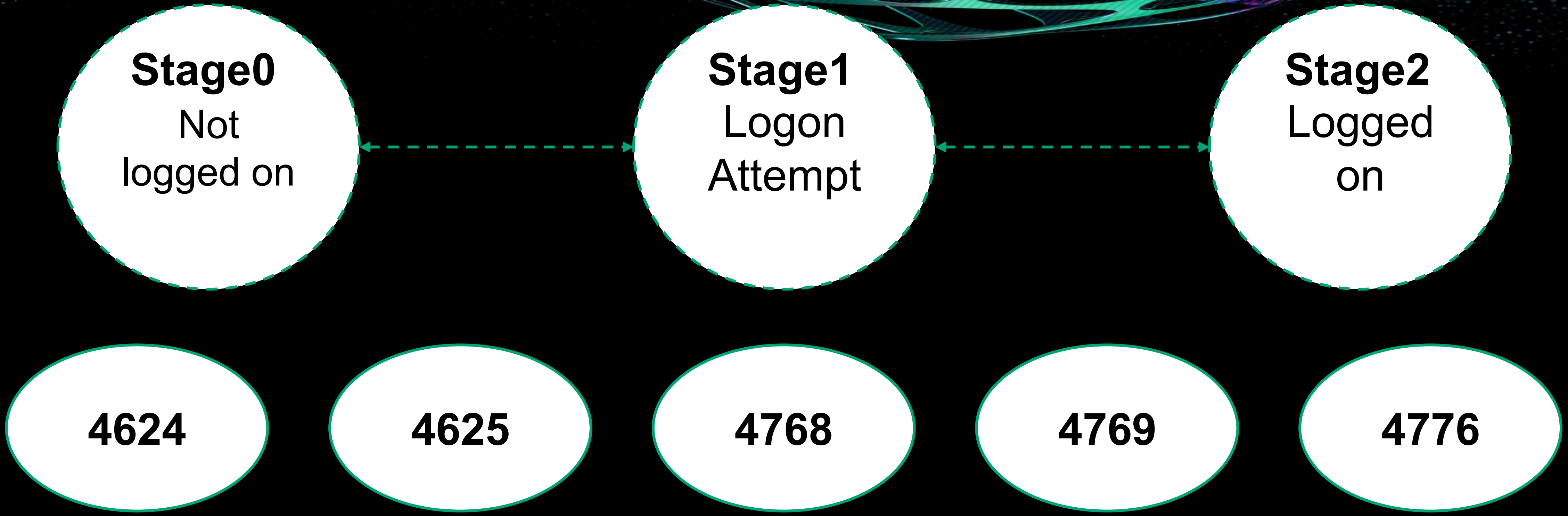
Hidden Markov Model (HMM)

The underlying states are hidden, and events are generated from those states.



Anomaly Detection of State Transition Using HMM

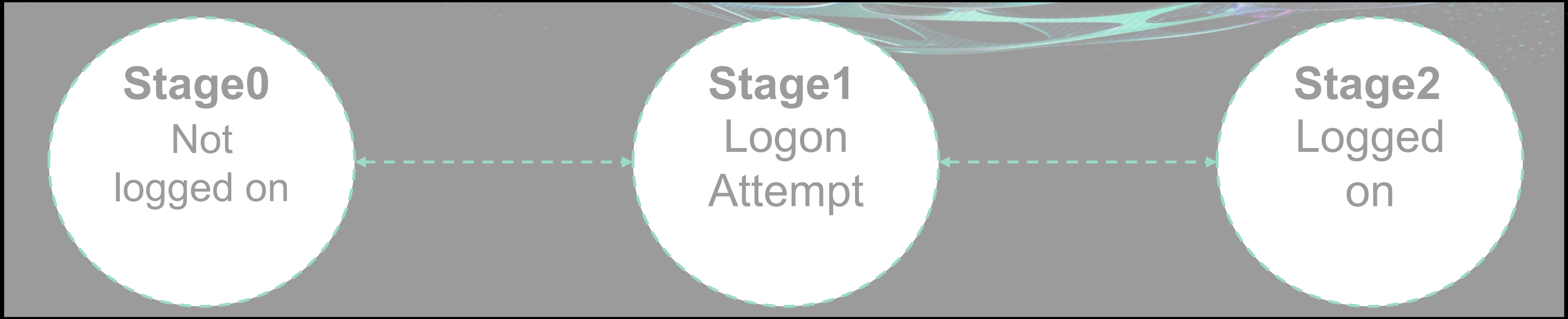
Logon state transitions HMM



Predict the state from the event ID timeline.
Detect missing transitions to Stage 1 or Stage 2.

Anomaly Detection of State Transition Using HMM

Logon state transitions HMM



4624 **4625** **4768** **4769** **4776**

➡ Predict the state from the event ID timeline.
Detect missing transitions to Stage 1 or Stage 2.

Anomaly Detection of State Transition Using HMM

Predict using HMM

Normal Logon

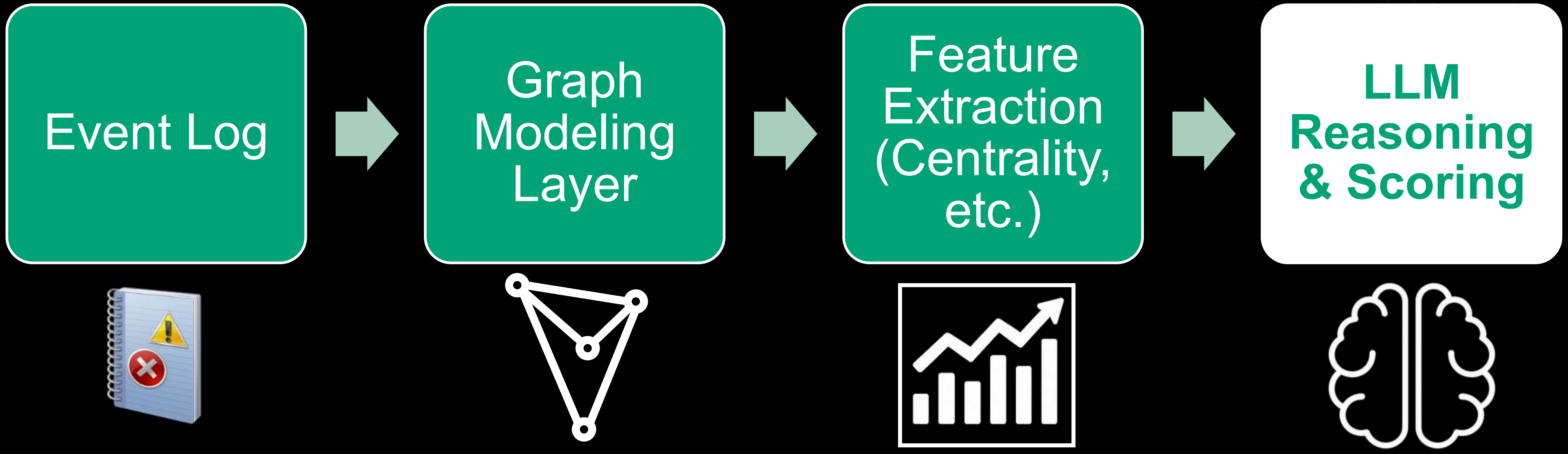
ID	4624	4624	4624	4624	4769	4624	4624	4769	4624	4624	4624	4624
Stage	0	0	0	1	2	0	0	2	0	0	0	0

Pass-the-Ticket

ID	4768	4769	4769	4624	4769	4624	4624	4776	4624	4769	4769	4624
Stage	0	2	2	0	2	0	0	2	0	2	2	0

➡ In the case of PtT, the sequence may skip Stage1.

Architecture Overview



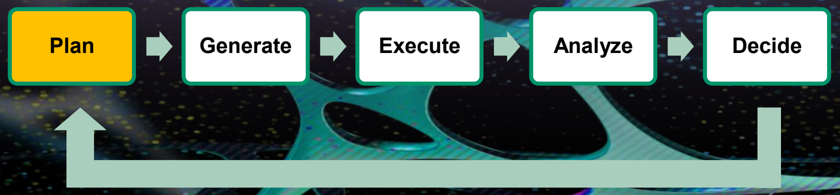
LLM Agents for Malicious Logon Hunting

LLM Agent Workflow

Plan → Generate → Execute → Analyze → Decide

Plan a hypothesis
Generate a parameterized query from vetted templates
Execute the query with safety limits; collect compact rows.
Analyze results and decide the next step.
Decide to continue or stop

LLM Agents: Plan Phase



Input

- **Rank** (PageRank+HMM)
- **Graph context** format (Event Log)
- **Attack patterns** for Windows Network
- Guardrails rules

Output

- Investigation focus
- Threat indicators (Event ID, Service, Ticket etc.)
- Target user or host

How do we prioritize investigations?

Attack patterns

Risk weights

PageRank + HMM

🔍 Investigation Timeline

- **Step 1: Privilege escalation scan**
  Query returned 7 results - **Threat Detected!**
- **Step 2: Privilege escalation scan**
  Query returned 18 results - **Threat Detected!**
- **Step 3: AS-REP weak TGT (proxy)**
  Query returned 1 results - **Threat Detected!**
- **Step 4: Kerberoasting exposure (4769 weak ticket crypto)**
  Query returned 1 results - **Threat Detected!**
- **Step 5: RDP movement**
  Query returned 0 results - **No threats found**
- **Step 6: NTLM/4776 activity on specific host**
  Query returned 8 results - **Threat Detected!**
- **Step 7: Network / RDP logon spread (logintype)**
  Query returned 100 results - **No threats found**
- **Step 8: Kerberoasting exposure (krbtgt requests)**
  Query returned 100 results - **No threats found**
- **Step 9: NTLM failed authentications (bad-password) on hosts**
  Query returned 3 results - **Threat Detected!**
- **Step 10: Privilege escalation scan**
  Query returned 10 results - **Threat Detected!**

LLM Agents: Query Generation Phase

Plan → **Generate** → Execute → Analyze → Decide

Input

- Investigation focus
- Threat indicators (Event ID, Service, Ticket etc.)
- Target user or host

Output

```
# Cypher query
MATCH (u:Username)-
[e:Event]-(i:IPAddress)
WHERE u.status
CONTAINS 'AddGroup'
AND e.id IN
[4624,4776,4768,4769]
RETURN u.user,
i.hostname, i.IP, e.id,
e.logintype, e.authname,
e.servicename,
e.ticketencryptiontype,
e.status LIMIT 100
```

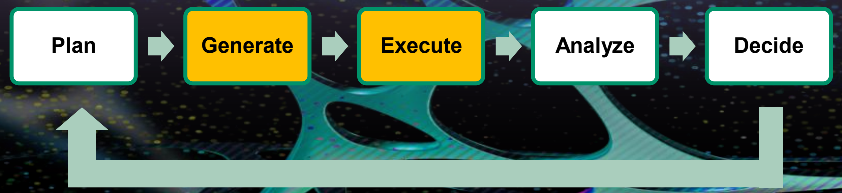
Prompt-based Guardrails and Auditability

Query safety	• allow-listed templates, parameterized queries, mandatory limits and time windows
Loop control	• max iterations, token budget, row caps, and strict stop conditions
Evidence requirement	• every claim must reference event IDs, timestamps, hosts, and statuses
Structured output	• bounded JSON with label, confidence, evidence, and recommended actions
Abstain policy	• if evidence is insufficient, the agent must abstain and request more data
Audit trail	• each step is logged as plan, query, result summary, and decision

Prompt Sample (System prompt < 1,500 tokens)

You are a DFIR-focused detection engineer who produces high-quality, production-ready Sigma rules. PRIMARY OBJECTIVE - Generate valid Sigma rules from the provided investigation evidence. - Evidence-first: every detection condition MUST be justified by fields/values present in the input. - Do not use any fields/values that do not exist in the evidence. INPUT CONTRACT - The investigation evidence may include: - threat_type, severity, description - evidence lines (human-readable evidence strings) - raw results (arrays of objects with keys like "e.id" or "e.logintype") - Treat the input as the complete dataset. Do not assume any information outside the input. SIGMA RULE QUALITY BAR - Follow the Sigma specification and common conventions. - Always use the following logsource: - product: windows - service: security - Each rule MUST include: - title, id, status, description, author, date, logsource, detection, falsepositives, level, tags - detection MUST be syntactically valid YAML and Sigma-compatible (no pseudo-fields). - Keep rules simple and clear. Avoid overly complex logic or deep nesting. - The condition do not use "not selection" or "1 of them". FIELD MAPPING (Neo4j → Windows Security Event Log concepts) - Evidence uses Neo4j-like keys. Map them to Windows Security event log fields as follows: - e.id → EventID - e.logintype → LogonType - e.authname → AuthenticationPackageName OR PackageName - e.status → Status - e.servicename → ServiceName - e.ticketencryptiontype → TicketEncryptionType - u.user → TargetUserName - i.IP → IpAddress OR WorkstationName OR Workstation - i.hostname → IpAddress FIELD VALUE NORMALIZATION - The following tokens in evidence are derived from Event IDs. Treat them as these Event IDs in Sigma rules: - AddGroup: 4728, 4732, 4756 - RemoveGroup: 4729, 4733, 4757 - Created: 4720 - Deleted: 4726 - DCShadow: 5137, 5141 - DCSync: 4662 - TicketEncryptionType values are normalized as follows. In Sigma rules, use the HEX values: - "0x1": "DES-CBC-CRC" - "0x3": "DES-CBC-MD5" - "0x11": "AES128-CTS-HMAC-SHA1-96" - "0x12": "AES256-CTS-HMAC-SHA1-96" - "0x17": "RC4-HMAC" - "0x18": "RC4-HMAC-EXP" - "0xffffffff": "-" EVENT LOG-SPECIFIC FIELD WHITELIST - Different Windows Security EventIDs expose different fields. When generating Sigma rules, you MUST: 1) Use ONLY fields that are valid for that EventID, 2) If a field is not listed for the EventID OR not present in Sample Event Data, DO NOT use it in detection. FIELD WHITELIST BY EVENTID (Sigma field names) 4624 - Allowed fields: IpAddress WorkstationName LogonType TargetUserName TargetDomainName TargetUserSid AuthenticationPackageName 4625 - Allowed fields: LogonType TargetUserName TargetDomainName TargetUserSid AuthenticationPackageName WorkstationName IpAddress Status 4776 - Allowed fields: TargetUserName Workstation Status PackageName 4768, 4769 - Allowed fields: TargetUserName TargetDomainName ServiceName TicketEncryptionType PreAuthType IpAddress Status 4672, 4662, 5137, 5141 - Allowed fields: SubjectUserName 4719 - Allowed fields: SubjectUserName CategoryId SubcategoryGuid 4728, 4732, 4733, 4729, 4756, 4757 - Allowed fields: TargetUserName MemberSid 4720, 4726 - Allowed fields: TargetUserName LANGUAGE CONTROL {language_note} FINAL CHECK BEFORE RESPONDING - YAML parses cleanly (proper indentation, lists, strings). - detection uses ONLY these fields: - EventID, LogonType, AuthenticationPackageName, PackageName, Status, ServiceName, TicketEncryptionType, TargetUserName, IpAddress, WorkstationName OR Workstation - detection uses ONLY fields and values that appear in the provided evidence. - Do not invent any values.

LLM Agents: Query Generate Phase



Generated Query Example

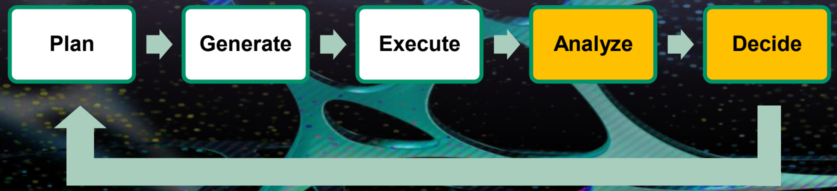
```
# Check privilege escalation

MATCH (u:Username)-[e:Event]-(i:IPAddress) WHERE u.status
CONTAINS 'AddGroup' AND e.id IN [4624,4776,4768,4769] RETURN
u.user, i.hostname, i.IP, e.id, e.logintype, e.authname, e.servicename,
e.ticketencryptiontype, e.status LIMIT 100


# Check brute force

MATCH (u:Username)-[e:Event]-(i:IPAddress) WHERE u.rights
CONTAINS 'system' AND e.id=4776 RETURN u.user, i.hostname, i.IP,
e.id, e.logintype, e.authname, e.servicename, e.ticketencryptiontype,
e.status LIMIT 100
```

LLM Agents: Analyze / Decide Phase



Input

- Cypher query
- Query results
- Investigation focus
- Guardrails rules

Output

- Threat detection status
- Threat type
- Risk level (low, medium, high)
- Evidence
- Investigation complete status
  - Next investigation focus

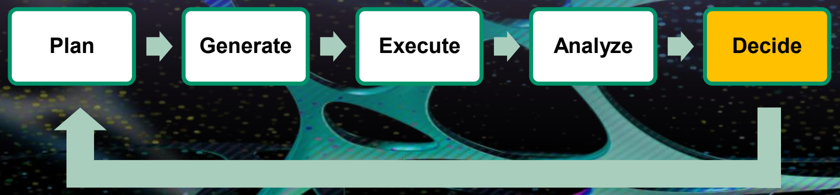
Closed-Loop Investigation Workflow

The AI agent does not stop after a single query. It keeps investigating until it finds enough evidence or decides to stop.

Choose the next step

- If evidence is strong, it continues narrowing the investigation
- If nothing is found, it tries a new pattern
- If needed, it expands the search scope
- If enough evidence is collected, it stop and generates the report

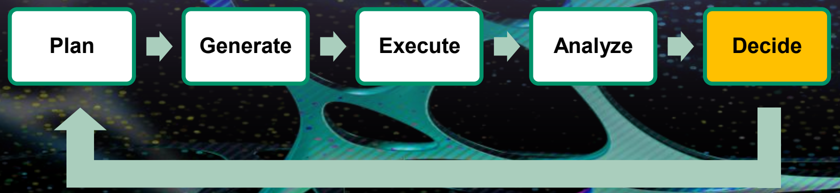
LLM Agents: Report Phase



Output

- Analysis summary
- Risk level
- Recommendations
- Total threats
  - Threat type
  - Evidence

LLM Agents: Report Phase



Output

Report Example

\# Analysis summary

Top hotspots: machida.kanagawa -> win10_64jp_09 ('192.168.16.109') and maebashi.gunma -> win10_64jp_09 using mixed NTLM/Kerberos.

machida.kanagawa shows status containing 'AddGroup: Domain Admins' and rights='system' with Event.id=4624 (logintype=3, authname=NTLM), multiple Event.id=4776 (logintype=0) and multiple Event.id=4625 (logintype=3, authname=NTLM, status='0xc000006d'), indicating active use after a privilege change.

maebashi.gunma generated Event.id=4768/4769 for servicename='krbtgt' with ticketencryptiontype='RC4-HMAC' and multiple 4776 NTLM events on the same host, indicating Kerberos weak-crypto exposure and mixed auth.

All confirmed activity consolidates to win10_64jp_09 ('192.168.16.109’).

Evidence supports at least one **compromised/high-risk account (machida.kanagawa)** and a **credential-theft exposure (maebashi.gunma) suitable for offline cracking**.

1 Architecture Overview

2 Evaluation

3 Demonstration

4 Further Use of AI

Evaluation Details

Detection Accuracy (vs. Sigma Baseline)
Precision and Recall on Unified Host and Network Data Set using Sigma as ground truth

Run-to-Run Stability (Reproducibility)
Repeated runs, measured by Jaccard overlap of detected compromised accounts

Technique Coverage (Attack Pattern Results)
Which attack techniques are detected, partially detected, or missed

Operational Validation (Real Incident Case Study)
Performance and analyst value demonstrated on a real IR engagement

Evaluation: Detection Accuracy

Precision & Recall

- Measure the LLM agent's detection capability against a rule-based baseline (SigmaHQ[3]).

Ground Truth

- The dataset is the Unified Host and Network Data Set [2].(12.8GB JSON log)
- Defined the ground truth as the set of compromised accounts identified by Sigma.

Results

Precision	Recall
0.90	0.931

n=32

Evaluation: Reproducibility Protocol

Goal

- Assess how consistently the LLM-based detection agent identifies the same compromised account across repeated runs.

Metric

- Jaccard index
  - statistic used for gauging the similarity and diversity of sample sets.
  - Measure the similarity between each response in the LLM.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Evaluation: Reproducibility Protocol

Jaccard index example

Run 1st

- detected accounts
- A, B, C

Run 2nd

- detected accounts
- A, B, D

$$\text{Jaccard} = \frac{|\{A,B\}|}{|\{A,B,C,D\}|} = 0.5$$

Evaluation: Reproducibility Protocol

Ground Truth

- Dataset is Unified Host and Network Data Set[2]. (12.8GB JSON log)

Results

$$\text{Average Jaccard Index} = 0.806 \quad \text{n=32}$$

High agreement across runs: LLM consistently identified the same account.
Minimal false-positive account detections.

Detection Results for Attack Patterns

Attack patterns

- Pass-the-Ticket
- Kerberoast
- NTLM relay
- DCSync
- DCShadow
- AS-REP Roasting

Detection Results for Attack Patterns

Attack patterns

- ☑ Pass-the-Ticket
- ☑ Kerberoast
- ☐ NTLM relay
- ☑ DCSync
- ☑ DCShadow
- ☑ AS-REP Roasting

NTLM relay attacks may be detectable based on the attacker's logon patterns.

Real Incident Case: Environment and Intrusion Overview

Incident Details

- Initial access via **Ivanti Connect Secure** exploitation
  - Activity associated with SPAWNCHIMERA
- Compromise of multiple internal systems
  - Predominantly living-off-the-land (LOTL) activity

Threat Actor

- Cluster consistent with **UNC5221**

Internal Network Environments

- Enterprise Windows environment with multiple Active Directory
- Total hosts: **2,720**
- Total accounts: **2,806**

Real Incident Case: Case Scale and Analysis Outcome

Dataset at a Glance

- Log sources: Windows Security logs from DC
- Total event log size: **869 MB**
- Total events: **821,272**
- Investigation window: 2025-05-08 to 2025-05-09

Processing Time

- End-to-end AI analysis time: **195 seconds** (using GPT-5-mini)

Findings

- Threats identified: **2**
    - Kerberoasting exposure
    - Multiple privileged accounts performing remote or network logons

Real Incident Case: AI Agent Analysis Steps

1. Privilege escalation scan with change group policy
2. Privilege escalation scan with more details
3. Kerberoasting exposure with weak encryption types
4. Privilege escalation scan with muitl-logon success — **Find threat!**
5. Kerberoasting exposure with logon success — **Find threat!**

Real Incident Case: AI Agent Analysis Step Details

4. Privilege escalation scan with multi-logon success

AI agent created query

MATCH (u:Username)-[e:Event]-(i:IPAddress) WHERE u.rights CONTAINS 'system' AND e.id=4624 RETURN u.user, i.hostname, i.IP, e.id, e.logintype, e.authname, e.servicename, e.ticketencryptiontype, e.status LIMIT 1000

AI agent description

Multiple privileged accounts (Username.user=AAA, BBB, CCC, DDD,EEE) show widespread remote/network logons (Event.id=4624) to many distinct hosts (e.g., IPAddress.IP=10.0.71.227, 10.0.71.177, 10.0.71.138, 10.0.74.91) using logintype=3 and authname=Kerberos (with intermittent NTLM observed for some users/hosts). Pattern indicates network/remote spread of privileged accounts across ≥3 hosts (Event.logintype=3) consistent with **lateral movement risk**.

Real Incident Case: AI Agent Analysis Step Details

5. Kerberoasting exposure with logon success

AI agent created query

```
MATCH (u:Username)-[e:Event]-(i:IPAddress) WHERE e.id=4769 RETURN
u.user, i.hostname, i.IP, e.id, e.logintype, e.authname, e.servicename,
e.ticketencryptiontype, e.status LIMIT 1000
```

AI agent description

Username.user='FFF' -> IPAddress.hostname='10.0.15.111' at unknown via Event.id=4769 (logintype=0) — servicename='APPSERV02$' using ticketencryptiontype='RC4-HMAC', status='-'. This indicates Kerberos ticket requests using weak RC4 crypto (**kerberoasting exposure**). Other 4769 records show AES256 for different users/hosts;

Real Incident Case: AI Agent Analysis Steps

6. Kerberoasting exposure by privileged accounts

7. AS-REP weak TGT (proxy)

Find same threat!

8. Kerberoasting exposure (recheck)

9. Incident summary reporting

10. Analysis finished

Real Incident Case: AI Agent Analysis Step Details

10. Analysis finished

Analysis summary

privileged account AAA shows widespread remote Kerberos logons to many hosts (Event.id=4624, logintype=3), and account BBB requested a non-AES Kerberos service ticket to APPSERV02$ on host 10.0.15.111 (Event.id=4769, ticketencryptiontype=RC4-HMAC). The AAA pattern (many distinct IPAddress.IP values: 10.0.74.91, 10.0.71.227, 10.0.71.177, 10.0.71.138, etc.) indicates lateral movement risk via network/remote logons. The BBB→ 10.0.15.111 RC4 TGS request is a Kerberoasting exposure and was raised in iterations 5 and 10. Multiple privileged users (CCC, DDD and others) also show logintype=3 Kerberos/NTLM activity to multiple hosts, and NTLM observed on DC03 for CCC suggests mixed auth methods. ticketencryptiontype non-AES and cross-host spread are concrete risks. Based on aggregated evidence, **the most immediate actionable compromise candidate is BBB(Kerberoast target) and the highest lateral-movement hotspot is AAA targeting many hosts (possible compromised privileged account).**

Failure Cases and Evasion Considerations

Low lateral movement attacks

- limited lateral movement may not stand out structurally

Shared admin accounts and jump hosts

- legitimate workflows can inflate centrality

Incomplete telemetry

- logging gaps reduce confidence and coverage

Evasion

- slow and low movement, account hopping, blending into maintenance windows

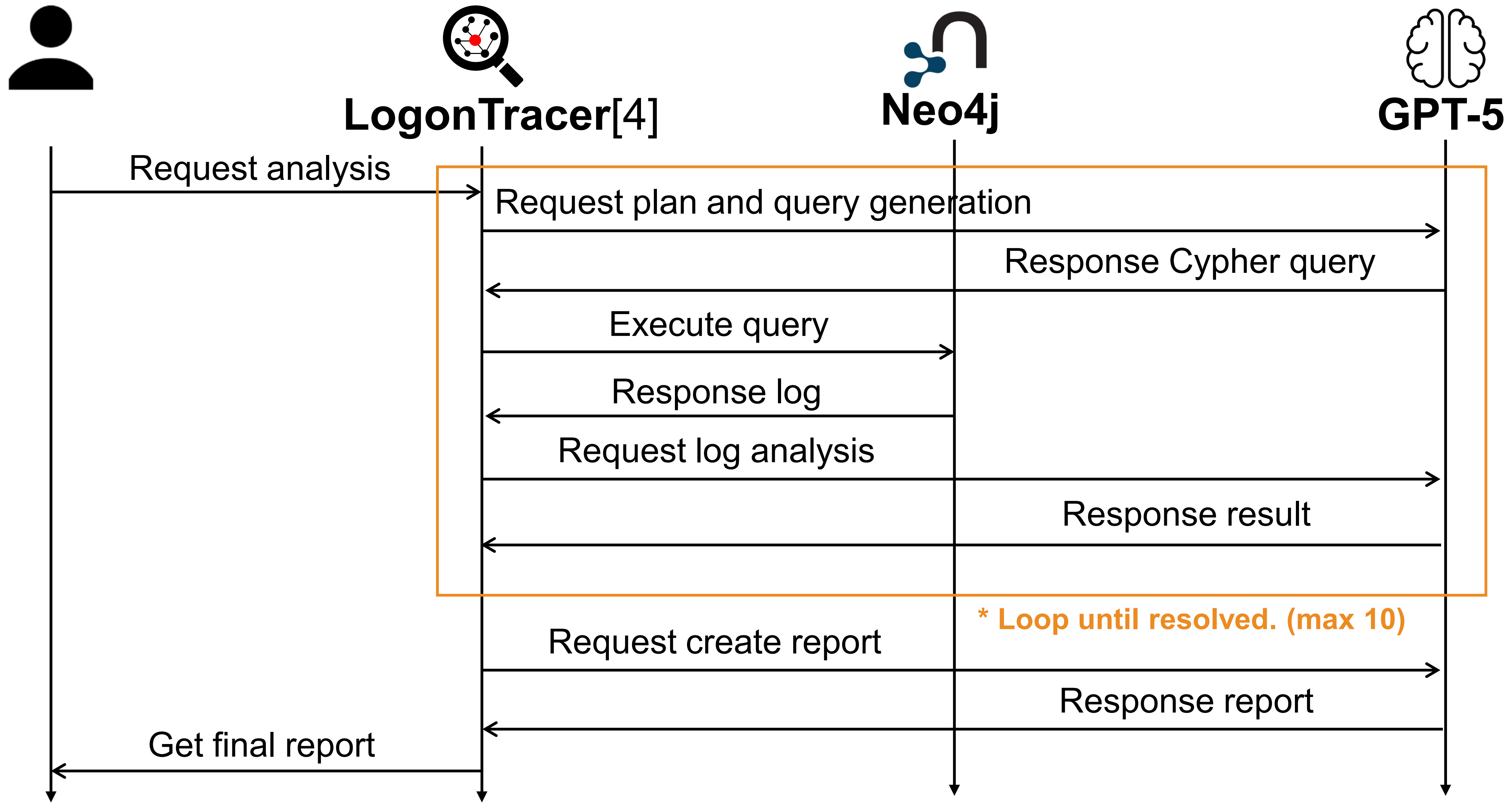
1 Architecture Overview

2 Evaluation

3 Demonstration

4 Further Use of AI

Architecture Details



1 Architecture Overview

2 Evaluation

3 Demonstration

4 Further Use of AI

From Detection Agents to Rule-Generation Agents

Use AI to turn incident findings into Sigma rules for recurrence prevention.

Security incident phase

Detection → Investigate → Prevent

Detection and investigation should lead to prevention. Sigma rules codify lessons learned into durable detection.

LLM Agents: Generate Sigma Rules

Input

- Sigma sample rule
- Event log columns
- Risk level (low, medium, high)
- Event Log Evidence

Output

- Sigma rules

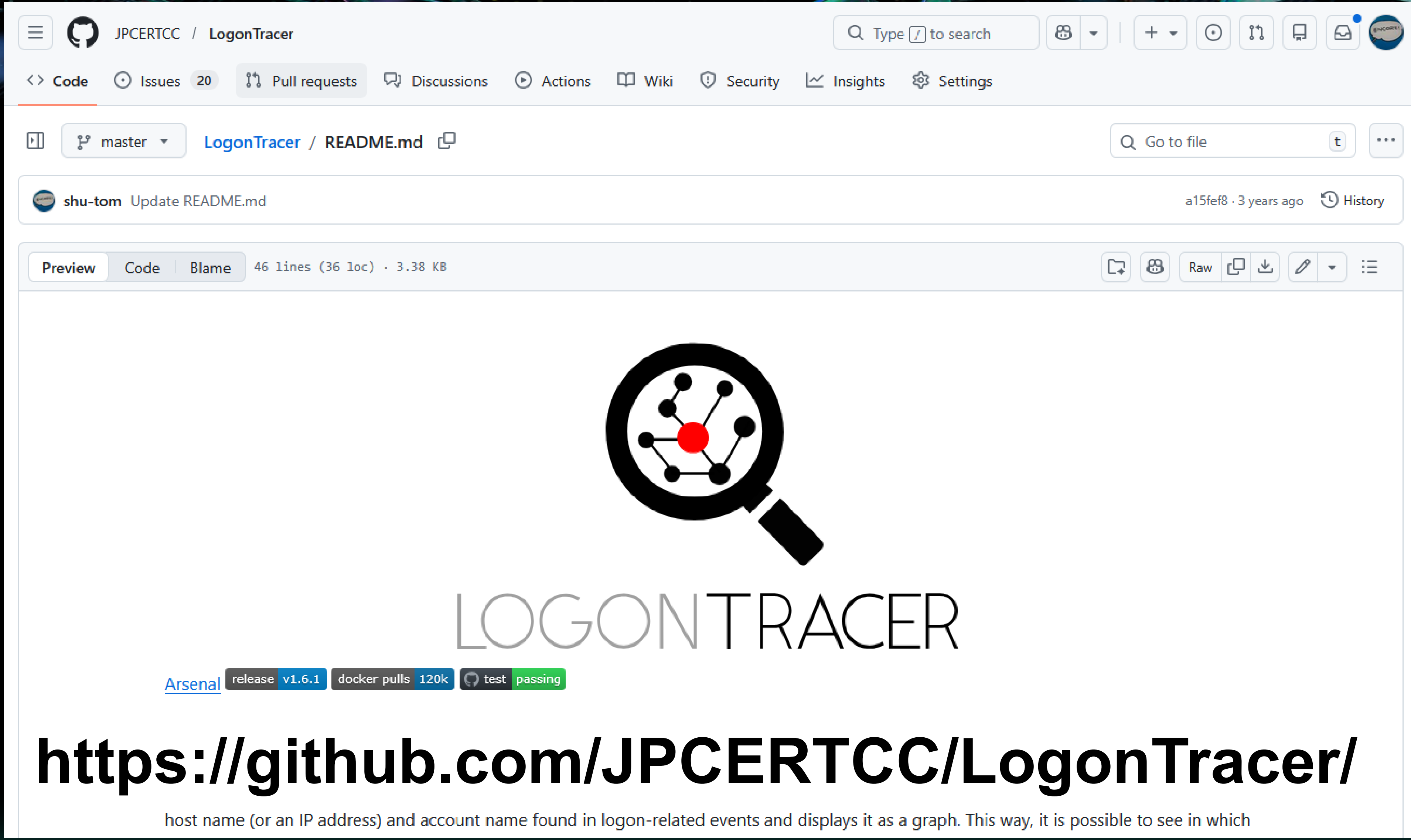
Event Log Evidence

```
# Sample of Event Log Evidence
"evidence": [
"Username.user='maebashi.gunma' Username.rights='system' IPAddress.hostname='192.168.16.109' IPAddress.IP='win10_64jp_09'",
"Event.id=4776 Event.logintype=0 Event.authname='MICROSOFT_AUTHENTICATION_PACKAGE_V1_0' same user/host",
"Event.id=4624 Event.logintype=3 Event.authname='Kerberos' Username.user='maebashi.gunma' IPAddress.hostname='192.168.16.109'",
"Event.id=4624 Event.logintype=3 Event.authname='NTLM' Username.user='maebashi.gunma' IPAddress.hostname='192.168.16.109'",
"Event.id=4768 servicename='krbtgt' ticketencryptiontype='AES256-CTS-HMAC-SHA1-96' Username.user='maebashi.gunma'",
"Event.id=4769 servicename='WIN10_64JP_09$' ticketencryptiontype='AES256-CTS-HMAC-SHA1-96' Username.user='maebashi.gunma'",
"Event.id=4769 servicename='WIN-WFBHIBE5GXZ$' ticketencryptiontype='AES256-CTS-HMAC-SHA1-96' Username.user='maebashi.gunma'"]
```

Created Sigma Rules

```yaml
# Rule 1: kerberos_service_ticket_with_aes256_for_privileged_account
title: Kerberos service ticket with AES256 for privileged account
id: 9b3f2d6e-3a7a-4f2c-9d2e-1f4a6b8c2d55
status: stable
description: Detect Kerberos service ticket requests using AES256 for a high privilege account
author: LogonTracer AI Analysis
date: 2026/01/27
references:
  - N/A
tags:
  - attack.privilege_escalation
  - attack.credential_access
logsource:
  product: windows
  service: security
detection:
  selection_krbtgt:
    EventID: 4768
    TargetUserName|contains: 'maebashi.gunma'
    ServiceName|contains: 'krbtgt'
    TicketEncryptionType|contains: '0x12'
    IpAddress|contains: '192.168.16.109'
  condition: selection_krbtgt
falsepositives:
  - Legitimate service ticket requests during normal account operations or service renewals
level: high
```

Tool Release on GitHub: LogonTracer v2.0



https://github.com/JPCERTCC/LogonTracer/

Limitations of AI Log Analysis

Knowledge gap

General LLMs can summarize logs, but deeper analysis needs extra knowledge such as event meaning, environment context, and attack patterns.

Constraints for trust

To keep results useful and repeatable, add constraints such as clear rules, structured outputs, and evidence requirements.

Bigger model is not always better

Using a more powerful model can produce overly complex analysis that is hard to act on for detection.

Practical goal

Guide the model with the right context and constraints, rather than simply using a bigger model.

Future Works

Automated optimization

Moving forward, **reinforcement learning** (or similar optimization loops) can help **automate fine-tuning and prompt/policy optimization** against measurable objectives such as precision, false positives, and reproducibility.

Evaluate **graph compression** for **proxy** and **access logs** to improve the efficiency of AI-agent investigations.

Black Hat Asia Sound Bytes

Compress first, then reason: Raw logs overwhelm LLMs. Graph compression makes investigation feasible.

Guardrails beat bigger models: Templates, evidence requirements, and structured output create trustworthy results.

AI should shorten the search, not replace the analyst: Reduce millions of events to a handful of evidence-backed leads you can act on.

Thank you!

Contact @jpcert_en ir-info@jpcert.or.jp

PGP https://www.jpcert.or.jp/english/pgp/

References

- [1] Siraaj Akhtar, Saad Khan, Simon Parkinson: LLM-based event log analysis techniques: A survey
  - https://arxiv.org/pdf/2502.00677
- [2] M. Turcotte, A. Kent and C. Hash, “Unified Host and Network Data Set”, in Data Science for Cyber-Security. November 2018, 1-22
  - https://arxiv.org/abs/1708.07518
- [3] GitHub: Sigma
  - https://github.com/SigmaHQ/sigma
- [4] GitHub: LogonTracer
  - https://github.com/JPCERTCC/LogonTracer/blob/master/sample/data.tar.gz

black hat®
ASIA 2026

APRIL 21-24, 2026

MARINA BAY SANDS / SINGAPORE