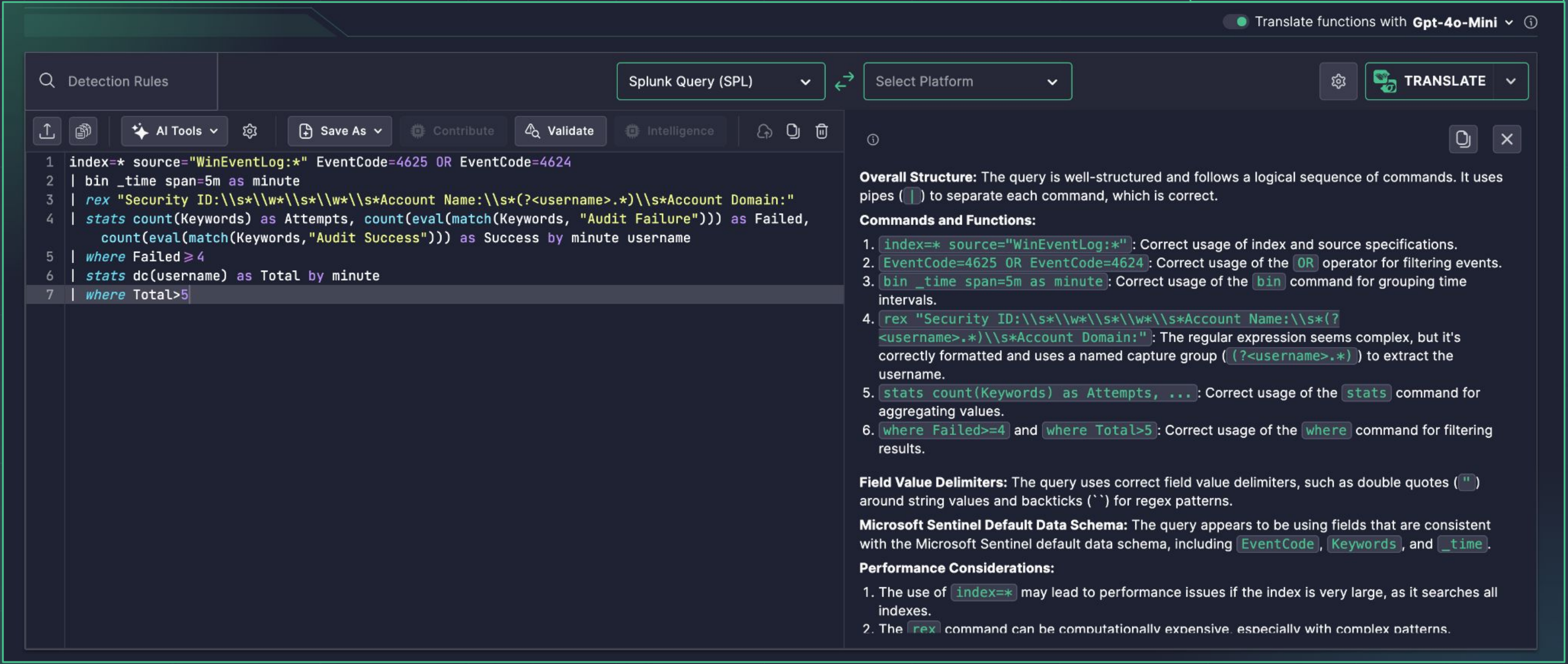
Rule Syntax and Structure Validation with AI

Uncoder AI

Rule Syntax and Structure Validation with AI

Released on April 16



```
index=* source="WinEventLog:*" EventCode=4625 OR EventCode=4624
| bin _time span=5m as minute
| rex "Security ID:\\s*\\w*\\s*\\w*\\s*Account Name:\\s*(?<username>.*)\\s*Account Domain:"
| stats count(Keywords) as Attempts, count(eval(match(Keywords, "Audit Failure"))) as Failed,
  count(eval(match(Keywords,"Audit Success"))) as Success by minute username
| where Failed>=4
| stats dc(username) as Total by minute
| where Total>5
```

Overall Structure: The query is well-structured and follows a logical sequence of commands. It uses pipes (`|`) to separate each command, which is correct.

Commands and Functions:

1. `index=* source="WinEventLog:*"`: Correct usage of index and source specifications.
2. `EventCode=4625 OR EventCode=4624`: Correct usage of the `OR` operator for filtering events.
3. `bin _time span=5m as minute`: Correct usage of the `bin` command for grouping time intervals.
4. `rex "Security ID:\\s*\\w*\\s*\\w*\\s*Account Name:\\s*(?<username>.*)\\s*Account Domain:"`: The regular expression seems complex, but it's correctly formatted and uses a named capture group (`(?<username>.*)`) to extract the username.
5. `stats count(Keywords) as Attempts, ...`: Correct usage of the `stats` command for aggregating values.
6. `where Failed>=4` and `where Total>5`: Correct usage of the `where` command for filtering results.

Field Value Delimiters: The query uses correct field value delimiters, such as double quotes (`"`) around string values and backticks (`` ` ` ``) for regex patterns.

Microsoft Sentinel Default Data Schema: The query appears to be using fields that are consistent with the Microsoft Sentinel default data schema, including `EventCode`, `Keywords`, and `_time`.

Performance Considerations:

1. The use of `index=*` may lead to performance issues if the index is very large, as it searches all indexes.
2. The `rex` command can be computationally expensive, especially with complex patterns.

Rule Syntax and Structure Validation with AI

Everyone makes mistakes, especially in a rapid rule development environment. Uncoder AI analyzes the syntax and structure of a rule/query and flags errors, suggests improvements, or confirms that everything is correct. For this purpose, Uncoder AI uses Llama 3.3 customized for detection engineering and threat intelligence processing, hosted at SOC Prime SOC 2 Type II private cloud for maximum security, privacy, and IP protection.

- Multiple languages supported
- Data doesn't leave SOC Prime's infrastructure
- Reduces time spent manually debugging syntax or structural issues to enable faster iteration and deployment of new detections
- Assists less experienced engineers by offering real-time, contextual feedback and improvement suggestions
- Flags logic flaws (e.g., overly broad conditions, redundant clauses), not just syntax

Rule syntax and structure validation with AI

56 languages supported

ANOMALI	APACHE kafka	ArcSight	CORTEX XSIAM BY PALO ALTO NETWORKS	CORTEX XDR BY PALO ALTO NETWORKS	AWS Athena	OpenSearch	DEVO
CROWDSTRIKE	FORTINET	REGEX	graylog	DATADOG	DNIF	ElastAlert	Elastic Stack
HUNTERS	LOGPOINT	LogRhythm	Falco	CrowdStrike Falcon LogScale	FIREEYE	Google Security Operations	hawksearch
Radar	LACEWORK	LIMA CHARLIE	LOGIQ	Microsoft Defender for Endpoint	Microsoft Sentinel	nviso	PowerShell
Qualys	GREP	ROOTA	RSA NETWITNESS	securonix	SentinelOne	snowflake	splunk>
SQL	SQLite	STIX	StreamAlert	sumo logic	Sysmon	uberAgent	Carbon Black.
onum	Logsign	SURICATA	TANIUM	SOPHOS EDR	logz.io	TREND MICRO XDR	exabeam