INCHEON NATIONAL UNIVERSITY

CYBERSECURITY CONVERGENCE LAB

MCP-CAT:

A Version- and Transport-Aware Conformance Automation Toolkit for MCP Servers

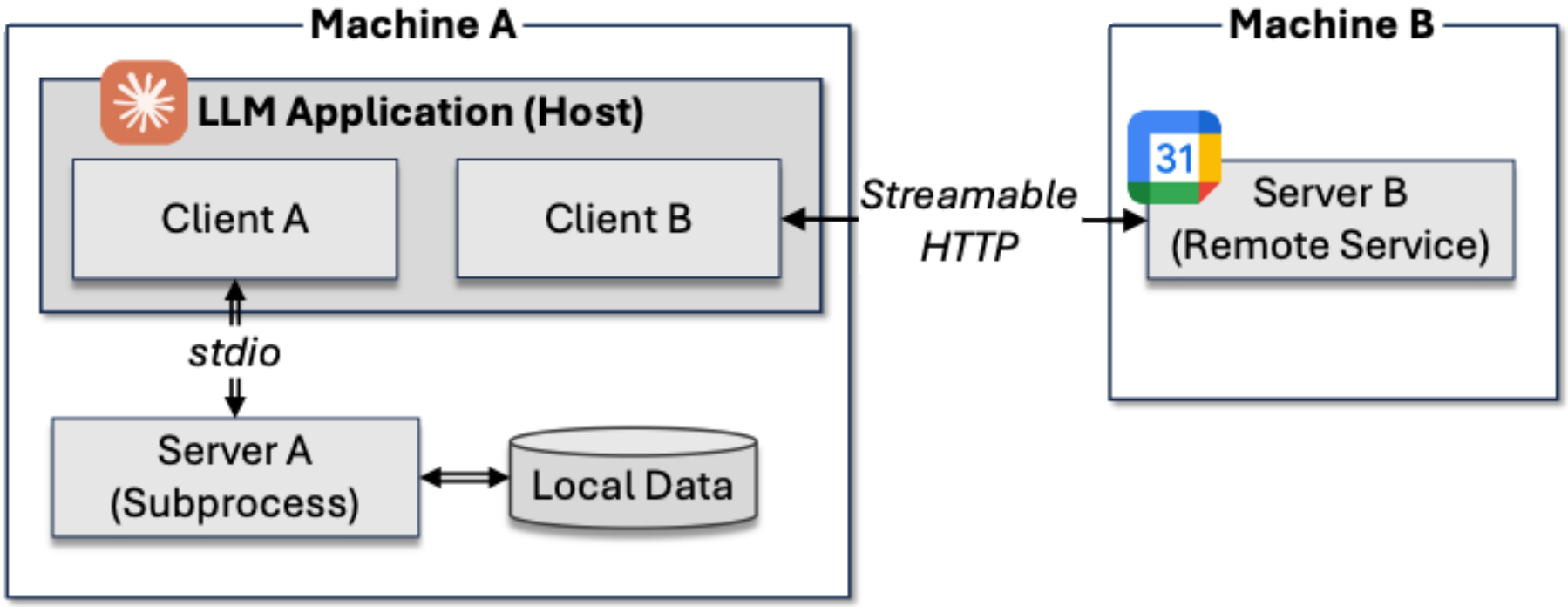
Ein Kim

Department of Computer Science & Engineering

Incheon National University

Background [1] Model Context Protocol (MCP)

- MCP lets LLM apps use external tools and data in a standard way.
  - A host runs multiple clients, each connected to one server via JSON-RPC 2.0.
  - Local servers use stdio, while remote servers use Streamable HTTP.



Background [2] Protocol Requirements

- The specification defines message formats and behavior through schemas and RFC2119 keywords.

Error responses are sent when the operation fails or encounters an error.

```
{
  jsonrpc: "2.0";
  id?: string | number;
  error: {
    code: number;
    message: string;
    data?: unknown;
  }
}
```

- Error responses **MUST** include the same ID as the request they correspond to (except in error cases where the ID could not be read due a malformed request).
- Error responses **MUST** include an `error` field with a `code` and `message`.
- Error codes **MUST** be integers.
- Error responses **MAY** include a `data` member with additional information of any type, such as nested errors.

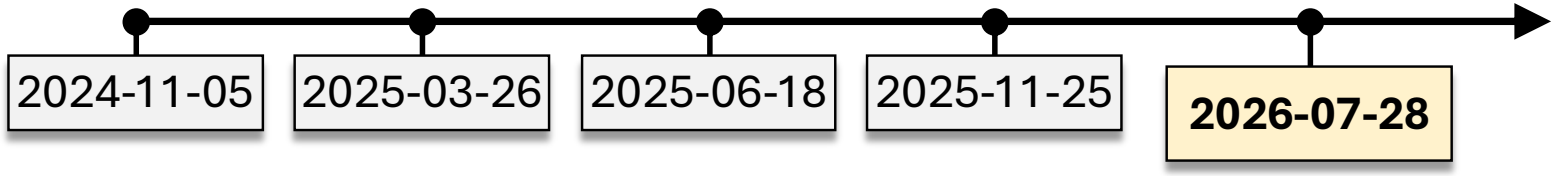
https://modelcontextprotocol.io/specification/2026-07-28/basic

```
298  export interface JSONRPCErrorResponse {
299    jsonrpc: typeof JSONRPC_VERSION;
300    id?: RequestId;
301    error: Error;
302  }
303
304  /**
305   * A response to a request, containing either the result or error.
306   *
307   * @category JSON-RPC
308   */
309  export type JSONRPCResponse = JSONRPCResultResponse | JSONRPCErrorResponse;
310
311  // Standard JSON-RPC error codes
312  export const PARSE_ERROR = -32700;
313  export const INVALID_REQUEST = -32600;
314  export const METHOD_NOT_FOUND = -32601;
315  export const INVALID_PARAMS = -32602;
316  export const INTERNAL_ERROR = -32603;
```

https://github.com/modelcontextprotocol/modelcontextprotocol/blob/main/schema/2026-07-28/schema.ts

Background [3] Protocol Versions

- Required procedures and message fields depend on the protocol version.
  - Legacy versions negotiate the protocol version and capabilities during initialization.
  - Modern versions carry the protocol version and client capabilities in each request.

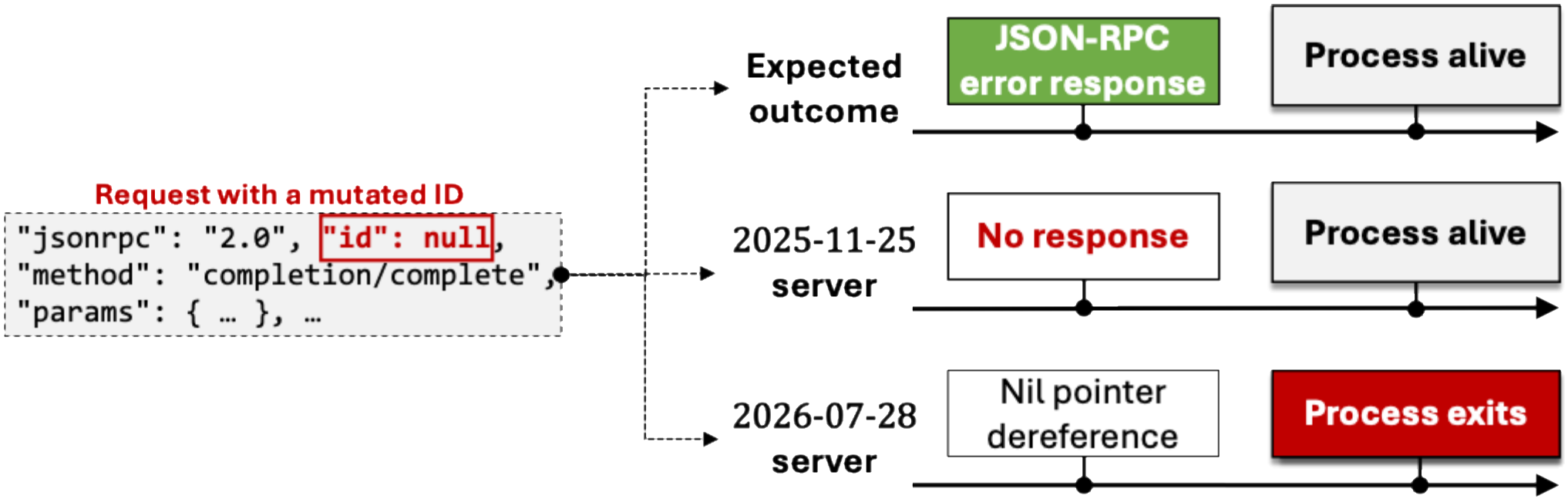



```
{ "jsonrpc": "2.0",
  "id": 1, "method": "initialize",
  "params": {
    "protocolVersion": "2025-11-25",
    "capabilities": {},
    "clientInfo": { "name": "Client", "version": "1.0" } }
}
```

```
{ "jsonrpc": "2.0", "id": 1, "method": "server/discover",
  "params": { "_meta": {
    "io.modelcontextprotocol/protocolVersion": "2026-07-28",
    "io.modelcontextprotocol/clientCapabilities": {},
    "io.modelcontextprotocol/clientInfo":
   { "name": "Client", "version": "1.0" } } }
}
```

Problem Statement

- Sent a request with `id: null` to Go SDK via stdio
  - The same input can trigger different failures across protocol versions.
  - A missing response does not distinguish a dropped request from a server crash.



Challenges

- Requirements span prose and schemas, and vary by version and transport.
- Tests may not reach the target behavior without the correct prerequisites, supported features, and runtime information.
- Response absence alone cannot distinguish failures with different causes and impacts.

Related Works

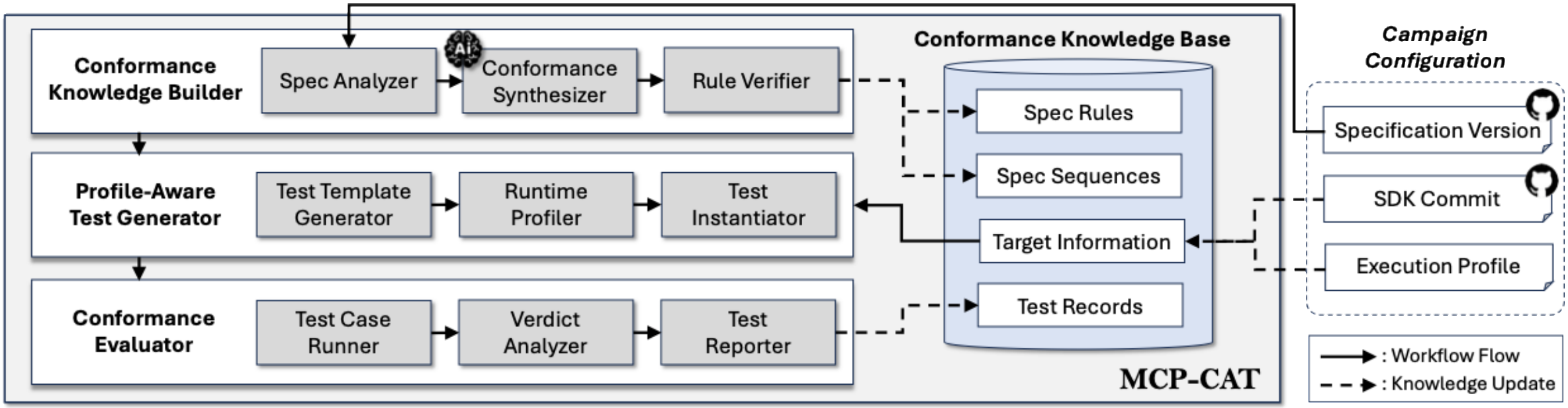
- Comparison of MCP-CAT with Prior Work

	ChatAFL [1]	RFCAudit [2]	Official Conformance [3]	MCP-CAT
Approach	LLM-guided fuzzing	LLM-agent code audit	Hand-authored scenarios	**Spec-derived tests**
Test basis	LLM-inferred grammar	RFC text vs. code	Scenario checks + schema	**specification + schema**
Detects	Crashes only	Functional bugs	Pass / fail per check	**Verdict, failure stage, scope**
Source code	Instrumented build	Required	Not needed	**Not needed**
Version / transport	–	–	Per-version, HTTP only	**Version-tagged, All standard transports**

[1] R. Meng, M. Mirchev, M. Böhme, and A. Roychoudhury, "Large Language Model guided Protocol Fuzzing," in Proceedings of the 31st Annual Network and Distributed System Security Symposium (NDSS), 2024
[2] M. Zheng, C. Wang, X. Liu, J. Guo, S. Feng, and X. Zhang, "RFCAudit: AI Agent for Auditing Protocol Implementations Against RFC Specifications," in Proceedings of the 40th IEEE/ACM International Conference on Automated Software Engineering (ASE), 2025
[3] Model Context Protocol, "MCP Conformance Test Framework," GitHub. https://github.com/modelcontextprotocol/conformance

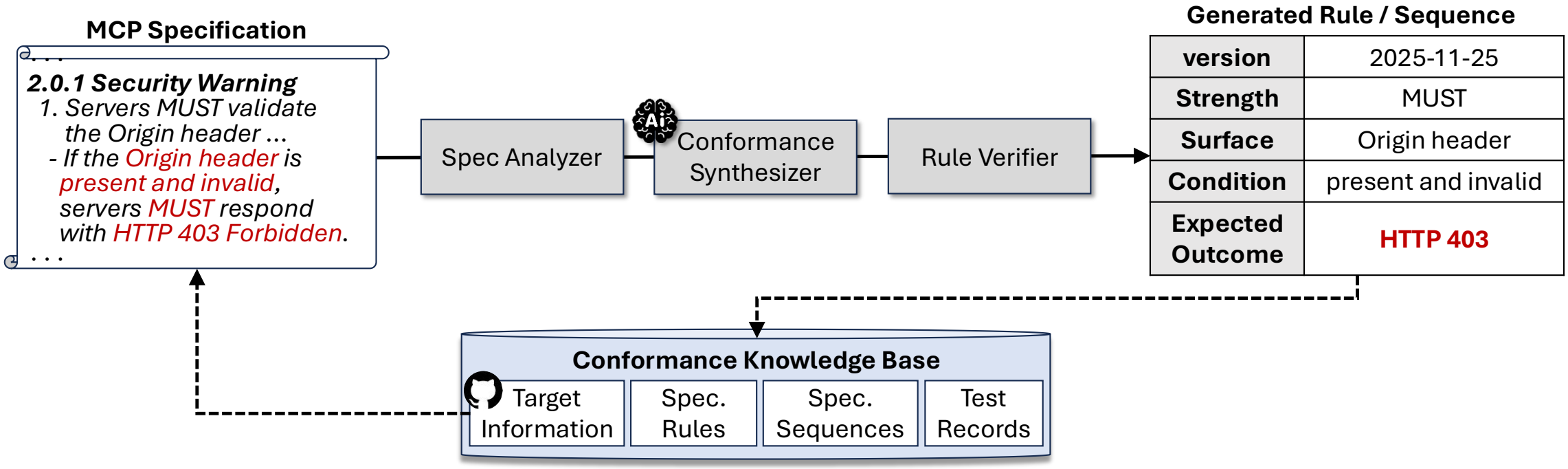
System Design

- MCP-CAT combines *specification analysis*, *profile-aware test generation*, and *evidence-based evaluation* to automate conformance testing.



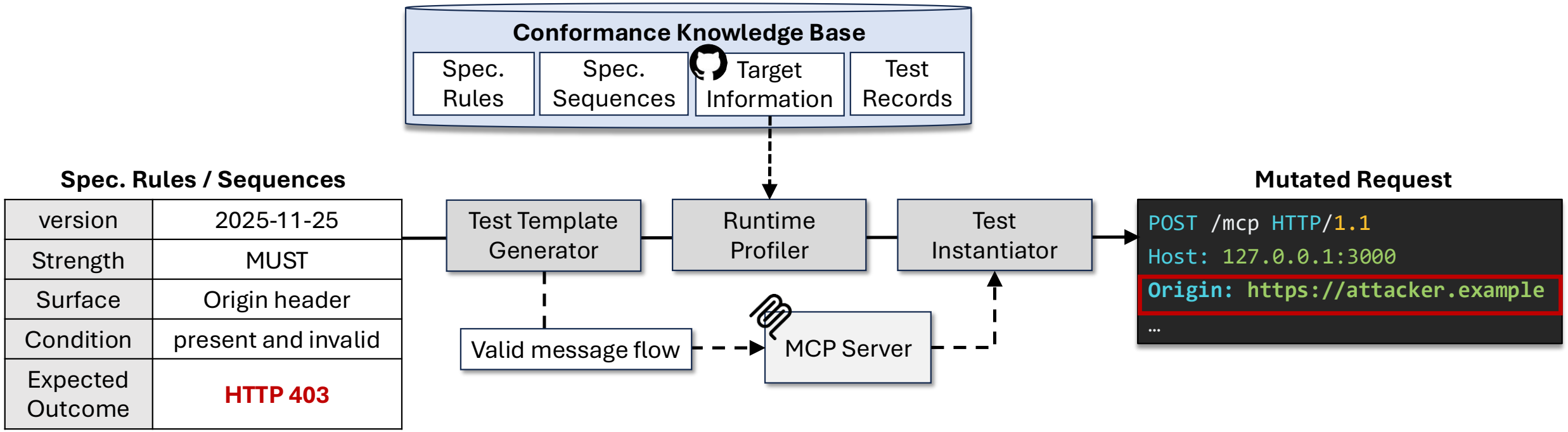
Details [1] Knowledge Construction

- The builder combines specification text and schemas to construct rules and message sequences.
  - It rejects items with invalid references or conflicting constraints.



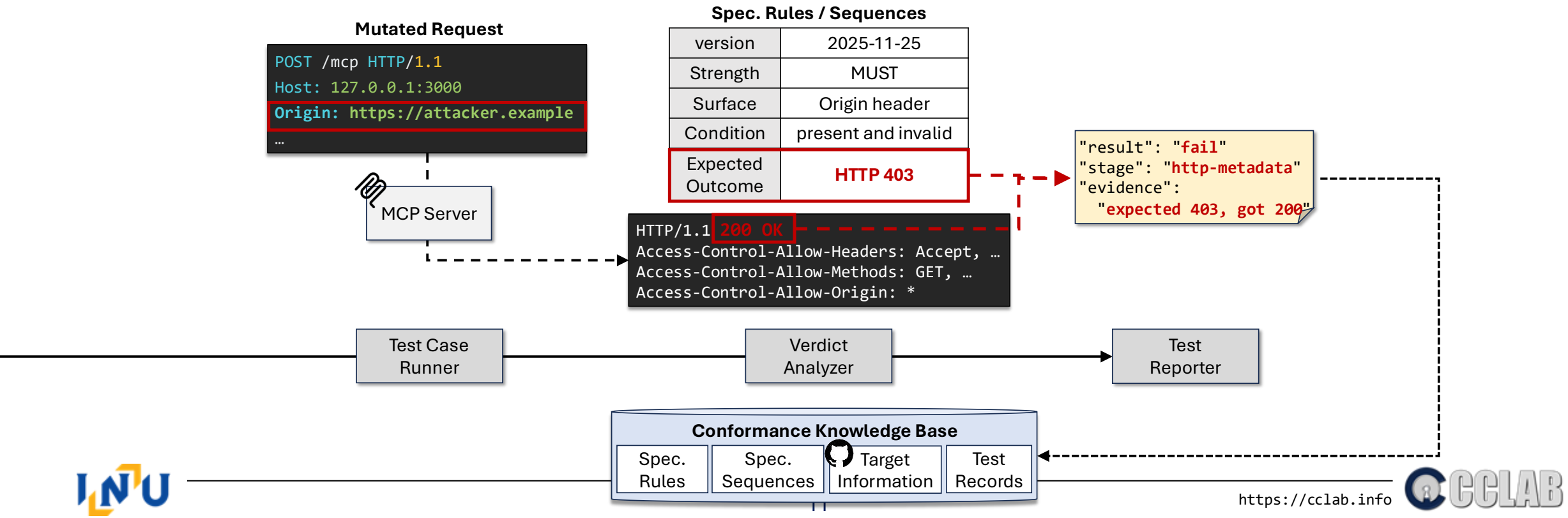
Details [2] Test Generation

- The generator uses runtime information to build executable tests for each target.
  - It mutates one condition after confirming successful baseline execution.



Details [3] Evidence-based Evaluation

- The evaluator derives verdicts, failure stages, and impact scopes from execution records.
  - It marks results as inconclusive when the specification or evidence is insufficient.



Evaluation [1] Setup

- Uses Qwen3.6-27B-AWQ to structure requirements.
- Covers 23 configurations across 9 SDKs, 2 spec versions, and 2 transports.

SDK	2025-11-25		2026-07-28	
	stdio	HTTP	stdio	HTTP
TypeScript	v1.29.0	v1.29.0	N/A	95d28cb
Python	v1.27.1	v1.27.1	N/A	0da9092
Go	v1.6.0	v1.6.0	65ece59	65ece59
Rust	v1.7.0	v1.7.0	N/A	N/A
Ruby	v0.16.0	v0.16.0	N/A	N/A
Kotlin	0.12.0	0.12.0	N/A	N/A
PHP	v0.5.0	v0.5.0	N/A	N/A
C#	v1.3.0	v1.3.0	d6b1615	d6b1615
Java	N/A	v1.1.2	N/A	N/A
Total	**8**	**9**	**2**	**4**

2025-11-25: release tags, 2026-07-28: pinned commits, N/A: no official reference server

Evaluation [2] Findings

ID	Ver.	Test case	Status by SDK
I1	L	Extreme JSON nesting	Fixed (Rs)
I2	L	Version conflict: header vs. body	Fixed (Rs, Rb, C#), In progress (TS, Py, Kt)
I3	L	Malformed session ID	Fixed (PHP)
I4	L	Duplicate initialize, modified params #1	Fixed (Go, Rb, Rs, PHP, Kt), In progress (TS, Py)
V1	L	Duplicate initialize, modified params #2	Fixed (Kt)
V2	L	Invalid jsonrpc/method types	Fixed (Go, Rb)
V3	L	Empty or null method/id	Fixed (Go), In progress (TS)
V4	L	Invalid MCP version header	Fixed (PHP, Rb)
V5	L	Disallowed Origin header	Fixed (PHP)
V6	L	initialize missing a required field	Fixed (Go, Rb, Kt)
V7	L, M	Modern version in legacy handshake	Fixed (Go, C#)
V8	L, M	Unknown method or invalid params	Fixed (Go)
V9	L, M	Cancellation notification with an ID	Fixed (Rb)
V10	M	Missing Mcp-Name header	Fixed (C#)
V11	M	Required resultType omitted	Fixed (Go, C#)
V12	M	id:null request	Fixed (Go)
V13	M	Server request with unset mode	Fixed (Go)

I: inconclusive, V: violation, L: 2025-11-25, M: 2026-07-28, Kt: Kotlin, Py: Python, Rb: Ruby, Rs: Rust, TS: TypeScript.

Evaluation [2] Findings

- Duplicate initialization behaved differently across SDKs and transports.

SDK	stdio	Streamable HTTP
Go	Accepted, stored initialization parameters overwritten	Accepted, stored initialization parameters overwritten
Ruby	Accepted, client info and capabilities overwritten	Accepted, **new session created** with updated client info and capabilities
Rust	Accepted, stored peer info unchanged	Accepted, stored peer info unchanged
PHP	Rejected, with an incorrect response ID	Rejected, with an incorrect response ID
Kotlin	Accepted, client info and capabilities overwritten	Rejected, but the response omitted the request ID
TypeScript	Accepted, client info and capabilities overwritten	Rejected, as already initialized
Python	Accepted, stored initialization parameters overwritten	Accepted, stored initialization parameters overwritten

Evaluation [3] Quantitative Results

- Valid test construction exceeded 99%, with violation precision above 91%.
  - 3 researchers reviewed the results against specification and execution records.

- Results differed across transports in about 1/6 comparable test pairs.

Metric	2025-11-25	2026-07-28
Valid test construction	481/484 (99.4%)	1,588/1,592 (99.7%)
Violation precision	91/99 (91.9%)	53/57 (93.0%)
Inconclusive precision	163/167 (97.6%)	300/304 (98.7%)
Failure-stage accuracy	92/99 (92.9%)	54/57 (94.7%)
Impact-scope accuracy	98/99 (99.0%)	55/57 (96.5%)
Cross-transport divergence	54/336 (16.1%)	6/35 (17.1%)

Conclusion

- **Conclusion**
  - MCP-CAT automates server conformance testing across protocol versions and standard transports.
  - Evaluation across 9 SDKs and 23 configurations led to 35 reports, with 29 cases fixed.

- **Future Work**
  - Extend coverage to full SDK APIs and client implementations.
  - Improve verdict accuracy for server-to-client and long-running requests.

Thank you for listening