A few months ago, we announced[1] the release of our HTTP to MCP Bridge[2], a tool designed to test MCP services using the same methods and tools typically employed in regular web service security testing.

We are now releasing a major update to our tool, which includes the following enhancements:"

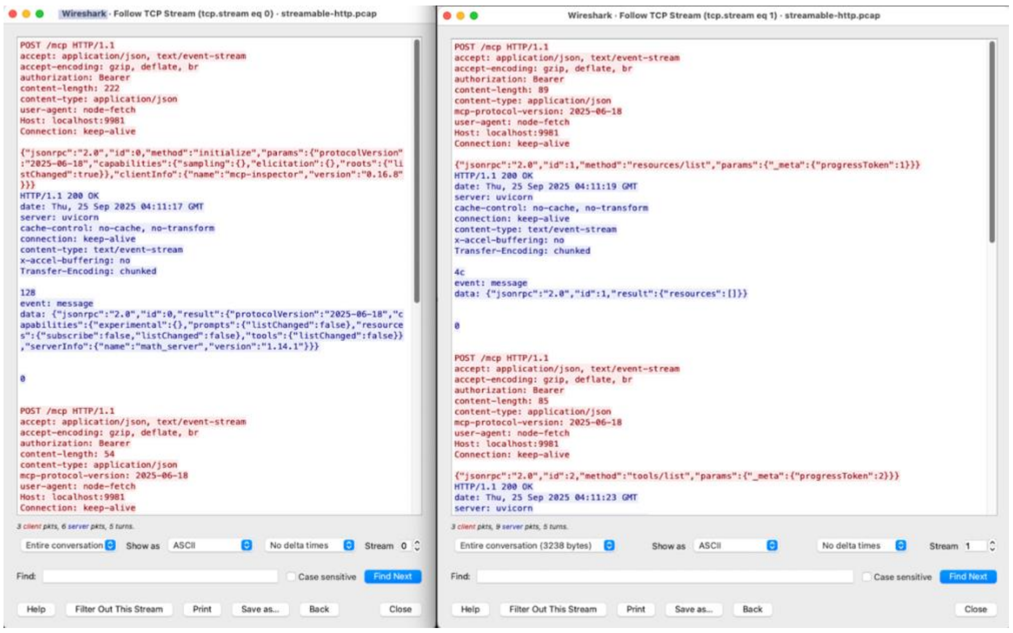
- Streamable HTTP Support
- Transport Mechanism Autodetection (Streamable HTTP or HTTP+SSE)
- Bridge endpoint update (now using /mcp/)
- Small refactoring and better error handling
- Added a simple math mcp server to help with testing

Some of them are detailed in the following sections.

Streamable HTTP Support

The most significant update in this release is the addition of support for the Streamable HTTP transport mechanism, alongside the previously supported HTTP+SSE.

As it was mentioned in the previous blog post[1], the HTTP+SSE is now considered deprecated in the MCP protocol, although it is still supported. The Streamable HTTP transport mechanism is the new and recommended option, so it was strongly necessary to add its support to our tool.



```
POST /mcp HTTP/1.1
accept: application/json, text/event-stream
accept-encoding: gzip, deflate, br
authorization: Bearer
content-length: 222
content-type: application/json
user-agent: node-fetch
Host: localhost:9981
Connection: keep-alive

{"jsonrpc":"2.0","id":0,"method":"initialize","params":{"protocolVersion"
:"2025-06-18","capabilities":{"sampling":{},"elicitation":{},"roots":{"li
stChanged":true}},"clientInfo":{"name":"mcp-inspector","version":"0.16.8"
}}}
HTTP/1.1 200 OK
date: Thu, 25 Sep 2025 04:11:17 GMT
server: uvicorn
cache-control: no-cache, no-transform
connection: keep-alive
content-type: text/event-stream
x-accel-buffering: no
Transfer-Encoding: chunked

128
event: message
data: {"jsonrpc":"2.0","id":0,"result":{"protocolVersion":"2025-06-18","c
apabilities":{"experimental":{},"prompts":{"listChanged":false},"resource
s":{"subscribe":false,"listChanged":false},"tools":{"listChanged":false}}
,"serverInfo":{"name":"math_server","version":"1.14.1"}}}


0


POST /mcp HTTP/1.1
accept: application/json, text/event-stream
accept-encoding: gzip, deflate, br
authorization: Bearer
content-length: 54
content-type: application/json
mcp-protocol-version: 2025-06-18
user-agent: node-fetch
Host: localhost:9981
Connection: keep-alive
```

Wireshark · Follow TCP Stream (tcp.stream eq 1) · streamable-http.pcap
3 client pkts, 9 server pkts, 5 turns.
Entire conversation (3238 bytes)
Show as ASCII
No delta times
Stream 1
Find:
Case sensitive
Find Next
Help
Filter Out This Stream
Print
Save as...
Back
Close

```
POST /mcp HTTP/1.1
accept: application/json, text/event-stream
accept-encoding: gzip, deflate, br
authorization: Bearer
content-length: 89
content-type: application/json
mcp-protocol-version: 2025-06-18
user-agent: node-fetch
Host: localhost:9981
Connection: keep-alive

{"jsonrpc":"2.0","id":1,"method":"resources/list","params":{"_meta":{"progressToken":1}}}
HTTP/1.1 200 OK
date: Thu, 25 Sep 2025 04:11:19 GMT
server: uvicorn
cache-control: no-cache, no-transform
connection: keep-alive
content-type: text/event-stream
x-accel-buffering: no
Transfer-Encoding: chunked

4c
event: message
data: {"jsonrpc":"2.0","id":1,"result":{"resources":[]}}


0


POST /mcp HTTP/1.1
accept: application/json, text/event-stream
accept-encoding: gzip, deflate, br
authorization: Bearer
content-length: 85
content-type: application/json
mcp-protocol-version: 2025-06-18
user-agent: node-fetch
Host: localhost:9981
Connection: keep-alive

{"jsonrpc":"2.0","id":2,"method":"tools/list","params":{"_meta":{"progressToken":2}}}
HTTP/1.1 200 OK
date: Thu, 25 Sep 2025 04:11:23 GMT
server: uvicorn
```

Figure 1 – Streamable HTTP Connections

The Streamable HTTP mechanism uses two different HTTP connections, but in a different way than HTTP+SSE. The first connection is used for handshake purposes, while the second connection is where all the "send" and "receive" operations happen. We found that the official MCP Python library implements a function called '*streamablehttp_client*'[3], which is very similar to the '*sse_client*' function previously used for HTTP+SSE support. Therefore, we adopted a similar approach in our tool.

Transport Mechanism Autodetection

Theoretically speaking, endpoints serving MCP via HTTP+SSE should use the “/sse” path, whereas those serving via Streamable HTTP should use the “/mcp” path. Our initial approach was to detect which mechanism to use by searching for specific strings in the URL. However, we have now implemented a true autodetection feature: the tool first attempts to use the Streamable HTTP mechanism, and if that fails, it falls back to HTTP+SSE

Each try is based on establishing the communication using the corresponding transport mechanism and then invoking the “ping” method, which is available even before the initial handshake.

```
INFO:     127.0.0.1:50629 - "POST /mcp/messages/6455a210-191e-428f-81cb-1cbbc0ecdfeb HTTP/1.1" 400 Bad Request

INFO:     [mcp_client] Connecting to MCP server...

INFO:     [mcp_client] URL: http://127.0.0.1:9981/sse

INFO:     [mcp_client] Headers: {'host': '127.0.0.1:8000', 'accept-encoding': 'gzip, deflate, br', 'connection': 'keep-alive', 'user-agent': 'python-httpx/0.28.1', 'content-type': 'application/json', 'cache-control': 'no-store'}

INFO:     [mcp_client] Receiving message...

ERROR:    [mcp_client] Error closing MCP context: unhandled errors in a TaskGroup (1 sub-exception)

INFO:     [mcp_client] Receiving message...

INFO:     [mcp_client] Received message: SessionMessage(message=JSONRPCMessage(root=JSONRPCResponse(jsonrpc='2.0', id=0, result={})), metadata=None)

WARNING:  [mcp_client] SSE Transport (Deprecated)

INFO:     [mcp_client] MCP streams created
```

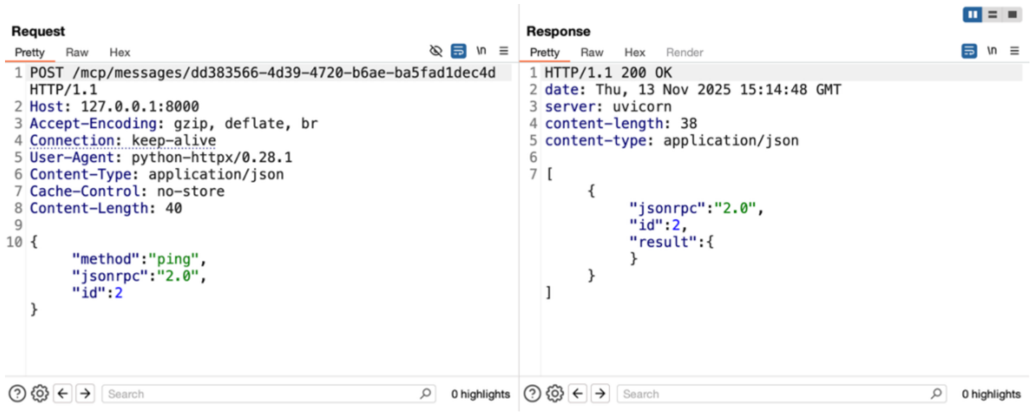
The output above shows how the tool invokes the ping method twice (“Receiving message…” entry). The first one fails, because it tries the Streamable HTTP mechanism and it is not supported by the target endpoint. The second succeeds, as it used the HTTP+SSE mechanism, and it receives the expected ping response. After that, it shows a “SEE Transport (Deprecated)” message to inform that the tool will be using that transport mechanism.

At present, it is not possible to enforce the use of a specific transport mechanism from the command line, but we plan to add this feature soon.

Bridge Endpoint Update

Initially, the available endpoints in the HTTP to MCP Bridge included ‘/sse’, ‘/sse/messages’, and ‘/raw/’. We have now renamed the ‘/sse*’ endpoints to ‘/mcp*’, while keeping the rest of the path unchanged from the previous release.

The reason for this change is to show that the tool is focused on MCP communications, and not only those MCP communications using SSE.



```
POST /mcp/messages/dd383566-4d39-4720-b6ae-ba5fad1dec4d
HTTP/1.1
Host: 127.0.0.1:8000
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
User-Agent: python-httpx/0.28.1
Content-Type: application/json
Cache-Control: no-store
Content-Length: 40

{
     "method":"ping",
     "jsonrpc":"2.0",
     "id":2
}
```

```
HTTP/1.1 200 OK
date: Thu, 13 Nov 2025 15:14:48 GMT
server: uvicorn
content-length: 38
content-type: application/json

[
     {
          "jsonrpc":"2.0",
          "id":2,
          "result":{
          }
     }
]
```

Figure 2 – HTTP to MCP Bridge using the new /mcp endpoint

In addition, we have removed the old '/raw*' endpoints. Although these were part of the initial functionality, their features are now included in the '/mcp*' endpoints, so we removed them for simplicity.

Simple Math MCP Server

If you would like to try the HTTP to MCP Bridge or a similar tool, we have included a simple implementation of an MCP server in 'tests/math_mcp_server/'. You can run it as follows:

```
$ cd http-mcp-bridge/tests/math_mcp_server

$ python3 -m venv .venv

$ source .venv/bin/activate

$ pip3 install -r requirements.txt

$ uv run .
```

This command runs the server at port 9981 using the Streamable HTTP transport mechanism. Alternatively, use the following command to run it using HTTP+SSE.

```
$ uv run . --sse
```

Then, you can run the HTTP to MCP Bridge, as usual:

```
$ python3 main.py --remote-url http://127.0.0.1:9981/sse

INFO:     Started server process [7622]

INFO:     Waiting for application startup.

INFO:     Starting up...

INFO:     Application startup complete.

INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
```

Further Steps

We plan to add support for the 'stdio' transport mechanism in the coming months, enabling our tooling to work with local MCP servers as well. Additionally, we aim to explore options for evaluating MCP client capabilities.

Please report any problems that you find, so we can make the tool more robust and useful for the infosec and AI security community.

References

[1] “HTTP to MCP Bridge”, https://www.nccgroup.com/research-blog/http-to-mcp-bridge/

[2] “Github – HTTP to MCP Bridge”, https://github.com/nccgroup/http-mcp-bridge

[3] “MCP Python SDK – streamablehttp_client”, https://github.com/modelcontextprotocol/python-sdk/blob/be730674adae118ad89806cd3011dc699fef8801/src/mcp/client/streamable_http.py#L447