Applying threat modelling to a Model Context Protocol Implementation

Andrew Snook
x23283301
MSC in Cybersecurity
National College of Ireland

Abstract

MCP is an emerging open standard in the world of Agentic AI but with this comes challenging implementations. Sever Spoofing, Insecure Communication, Prompt Injection are many of the vulnerabilities that are being found. Vibe coding, poor security coding practices and ad-hoc deployments are some of the many areas that introduce weaknesses that allow attacker to access private data and more. This proposal suggests using the STRIDE framework and applies it to a practical MCP solution.

Keywords: Model Context Protocol, STRIDE, MCP Security Threats, Threat Modelling

1. Introduction

Why is it important to reduce the implementation of insecure MCP solutions? What is the cost of not spending time, money and resources within this area? To answer this question, we must look at the current landscape.

Recent times have seen the emergence of a vibrant ecosystem where tools and data sources have grown exponentially and gained considerable traction. Accelerated by Open AI's release of function calling in 2023, language models began to interact with external API's in a structured manner. This advancement greatly expanded the capabilities of LLMs, enabling them to fetch real-time data, perform calculations, and interact with external systems. As function calling gained traction, a dynamic ecosystem started to take shape. OpenAI launched the ChatGPT plugin framework, enabling developers to build tools that could be directly accessed by ChatGPT. In response, LLM app platforms such as Coze and Yuanqi introduced their own plugin stores, further fuelling the growth of this diverse tool ecosystem. (Hou et al, 2025). The background to the research problem and resulting research question outlines this objective.

This will involve reviewing recent high-profile MCP related incidents. After exploring these areas in isolation, we will assess whether the threat modelling framework STRIDE could help mitigate the introduction of vulnerabilities.

In advance of reporting on each of the above, a further step back must be taken to ask the question “why?”. Why is it important to reduce the implementation of insecure MCP solutions? What is the cost of not spending time, money and resources within this area? To answer this question, we must look at the current landscape.

Recent times have seen the emergence of a vibrant ecosystem where tools and data sources have grown exponentially and gained considerable traction. Accelerated by Open AI’s release of function calling in 2023, language models began to interact with external API’s in a structured manner. This advancement greatly expanded the capabilities of LLMs, enabling them to fetch real-time data, perform calculations, and interact with external systems. As function calling gained traction, a dynamic ecosystem started to take shape. OpenAI launched the ChatGPT plugin framework, enabling developers to build tools that could be directly accessed by ChatGPT.

Toward the end of 2024, Anthropic unveiled the MCP as a comprehensive protocol designed to standardize AI interactions with external tools enabling AI applications to dynamically communicate with tools in real-time, eliminating the dependency on fixed tool mappings. Rather than relying on predefined connections, MCP empowers AI agents to independently discover, choose, and coordinate tools according to the task at hand. Additionally, it supports human-in-the-loop functionality, allowing users to input data or approve actions as needed.

2. Related Work

2.1 Architecture of a Model Context Protocol (MCP)

Designed to overcome the limitations involved in communicating across Large Language Models (LLM), MCP is a framework created to facilitate seamless interaction by establishing a consistent method for sharing contextual information across different tools, platforms and services. MCP's have a unique layered host-client-server architecture designed for extensibility, modularity and security. Through the separation of hosts, clients and servers MCP has established a mechanism enabling isolation of concerns and incremental upgrades to its capability (Ray., 2025).

Below is an example of a typical MCP-enabled architecture:

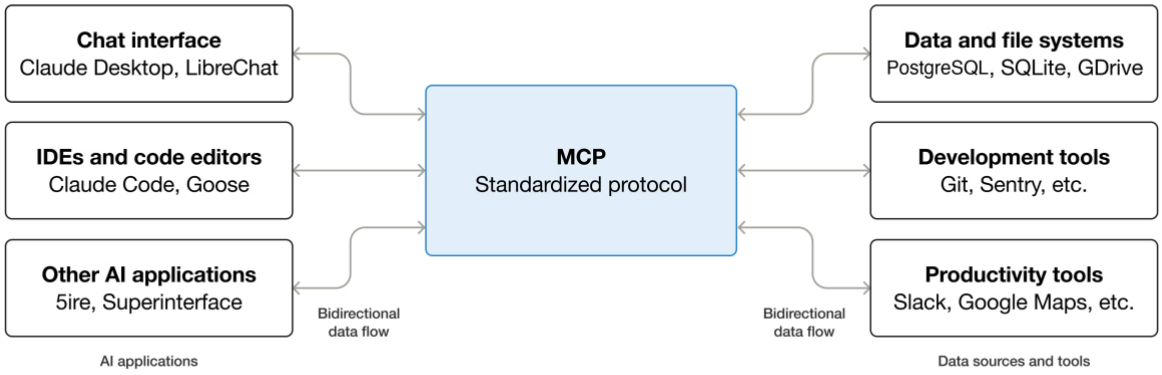


Figure 1 MCP Architecture - https://modelcontextprotocol.io/docs/getting-started/intro

A larger, more detailed image can be found in the Appendix at the bottom of the document.

By supporting capability negotiation, MCP enables backward-compatible improvements and ensures strong security using OAuth 2.1. This approach meets a critical need in the AI ecosystem by: (i) simplifying the development of advanced, (ii) context-aware applications, (iii) creating a foundation for scalable expansion, and (iv) promoting secure AI innovation (Ray., 2025)

MCP as a whole is built up using the following primitives: **Tools** are executable function that perform actions, for example, creating a new file or calling an API. **Resources** are data sources such as a database or an API endpoint. The MCP can reach out to get real-time information or engage with another service. **Prompts** are similar to stored procedures or templates for providing input context to AI models. These help promote consistency so that the interaction patterns feel natural. Finally, **Sampling** is used to give parameters or weights to the LLM for instances when AI is required to think for itself.

2.2 MCP Communication Layers

The Model Context Protocol has two layers called the Transport Layer and the Data Layer.

Data Layer

The data layer uses a JSON-RPC 2.0 exchange protocol which defines the message structure, example below:

```
// Client request
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "tools/list",
  "params": {}
}

// Server response
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
    "tools": [{"name": "calculator", "description": "Basic math"}]
  }
}
```

Transport Layer

Transport	*Latency*	*Scalability*	*Deployment*	*Security*	*Best For*
STDIO	~1ms	Single client	Executable file	Process isolation	Local tools, Claude Desktop
HTTP	10-100ms+	High (load balancing)	Web server + networking	HTTPS + auth headers	Remote access, web apps
SSE	<100ms	Medium (connection limits)	Web server + event handling	HTTP security + validation	Real-time updates, monitoring

Table 1 - https://dev.to/bansikah/building-mcp-servers-understanding-transport-layers-and-core-components-3o1o

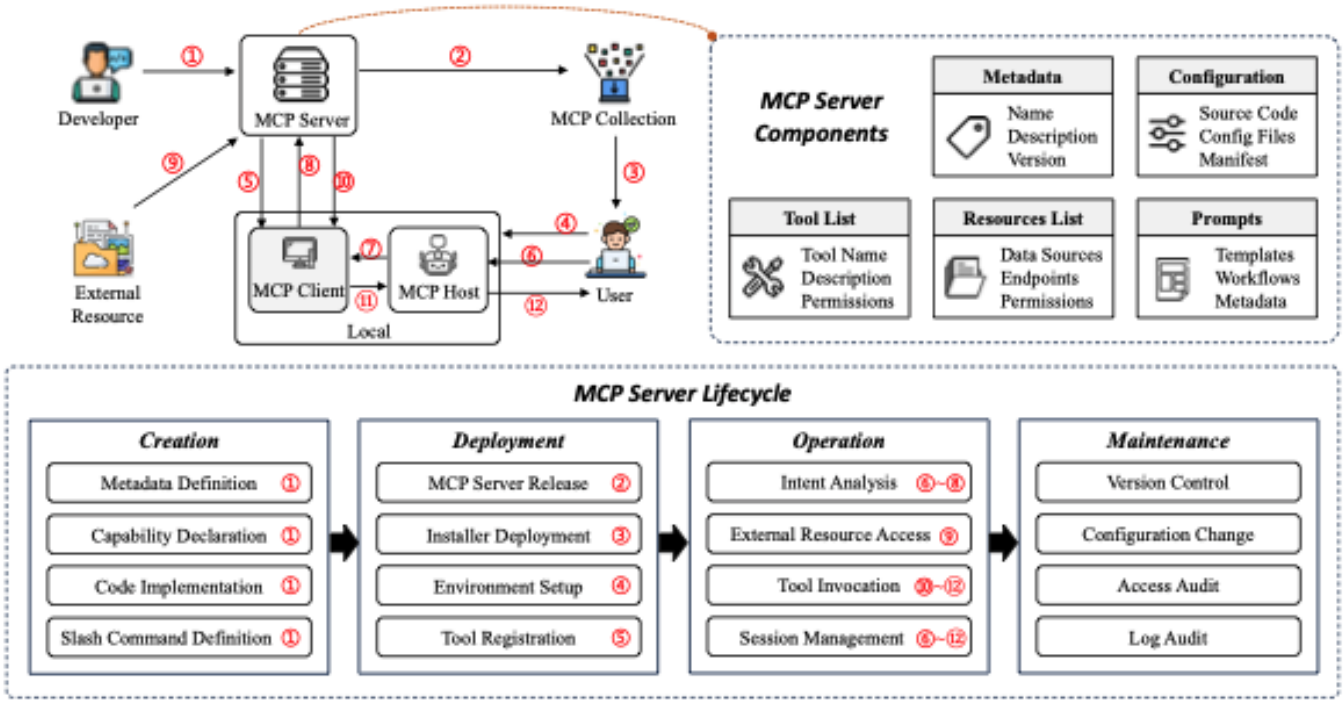


Figure 2 - MCP Server Lifecycle[1]

[1] https://arxiv.org/pdf/2503.23278

2.3 Existing Security Frameworks (including review of high-profile incidents)

Before the emergence of MCP a number of isolated tools had been created to complete function calling with LLM's. However, the pressing concern at this point was making certain that each LLM dealt with these function calls on an equal basis. The engineering of unique solutions for every data set and tool was not only time consuming but also allowed for increased development and maintenance errors (Swaelens & Yaremchuck, 2025).

Other approaches in this area such as the work by Narajala, Vineeth Sai & Habler, Idan, (Narajala, 2025) use a MAESTRO framework [2] created by OWASP. Brandon Radosevice and John Halloran have developed a tool called MCPSafetyScanner (Brandon & T., 2025) which checks for Malicious code execution, Remote access control and Credential theft exploits.

Whilst the introduction of MCP helped reduce errors, they do not come without their own security concerns including but not limited to; unmonitored access, absence of built-in approval workflows, inadequate audit and trails monitoring and issues relating to privilege management (Santos, 2025). While there exists a lack of publicized high profile incidents for apparent reasons, we can speculate that the following companies who are at the heart of the MCP ecosystem would be most affected by any vulnerabilities; Open AI (ChatGPT, Plugin Frameworks), Anthropic (MCP Developer), Meta (Facebook AI Tools, Integration), Google DeepMind (AI Tools and Services), Coze (LLM Plugin Store) and Yuanqu (LLM Plugin Store).

Equixly (equixly.com, 2025) carried out assessments on the most recently used MCP servers and the findings are a cause for concern. Their review found widespread issues: 43% of implementations had command injection flaws, 22% allowed file access outside intended paths, and 30% were vulnerable to SSRF. Only 30% of vendors fixed the issues; 45% dismissed them as low risk, and 25% gave no response. Even more critically they noted that MCP servers can be triggered by anyone, not just LLM, exposing systems to silent, uncontrolled attacks.

These vulnerabilities have been further evidenced by Invariant Labs (Invariant Labs, 2025) who demonstrated a critical security vulnerability in the Model Context Protocol where a malicious MCP server—when connected alongside a trusted WhatsApp MCP instance—can manipulate AI agents into leaking users' WhatsApp chat histories and contact lists, bypassing encryption without user awareness. The exploit works

[2] https://cloudsecurityalliance.org/blog/2025/02/06/agentic-ai-threat-modeling-framework-maestro

via deceptive tool descriptions and prompt injection, even without user re-approval or server interaction, revealing deep flaws in MCP's trust and execution model.

On the 25th of September, KOI.ai a security company specializing in self-provisioned software, reported the first malicious MCP (Dardikamn, 2025). An NMP package, Postmark-mcp version 1.0.16, hid a line of code that bcc'd an email address. Every time this MCP was called, the details were sent to this email address. The proposed impact below:

- 1,500 downloads every single week
- Potentially up to 3,000 to 15,000 emails everyday flowing straight to the scammers email address.

The exploit, categorised as an impersonation attack, was incredibly simple to implement. The developer took an official GitHub repository and added malicious BCC line. There were 15 other commits on this repository everything was normal. This exploit highlights the way in which MCP servers and AI agents can be a risk without the proper guardrails in place.

Based on the information provided in this literature review it is clear to see that while the architectural capabilities of MCPs are strong, as with all Agentic AI applications there are fundamental cyber security concerns which must be addressed. This report will seek to provide evidence as to whether the STRIDE threat modelling framework could help mitigate the introduction of the highlighted vulnerabilities.

3. Research Methodology

Sourcing peer reviewed documents for MCP has been a challenge as MCP is still relatively new, however, the implementation of MCP can still be safeguarded with robust secure coding techniques and frameworks. Leveraging security best practices like NIST (National Institute of Standards and Technology, 2022) , Open Web Application Security Project (OWASP) guidelines (OWASP, 2025) provide a solid and credible foundation to build on.

This research paper focuses on the STRIDE threat modelling framework. The STRIDE acronym itself represents six specific threats: Spoofing, Tampering, Repudiation, Information Disclosure and Denial of Service and Elevation of Privilege (Van Landuyt & Joosen, 2002). A breakdown of each of these threats can be found in Table 2 below. Modelling threats using these categories allows those studying them to understand, prevent and mitigate the threats. A very recent 2024 study completed by Gavric et al (2024) similarly utilised a STRIDE-based threat analysis of an IDS leading to the identification of the most potent threats along with their corresponding solutions. The outcome of this study was the making of a more secure and resilient data space architecture (Gavric et al, 2024).

Threat Type	**MCP Component**	**Attack Example**	**Impact**	**Mitigation**
Spoofing	MCP Server	Malicious server impersonates legitimate service	Credential theft, data exfiltration	Certificate pinning, mutual TLS
Tampering	Tool Descriptions	Modify tool descriptions post-deployment	LLM manipulation, unauthorized actions	Integrity checks, signed manifests
Repudiation	Tool Execution	Deny performing sensitive operations	Compliance violations	Audit logging
Information Disclosure	Error Messages	Stack traces reveal system internals	Attack surface mapping	Generic error responses
Denial of Service	Resource Consumption	Infinite loops, large payloads	Service unavailability	Rate limiting
Elevation of Privilege	Tool Access	Access admin tools without authorization	Full system compromise	RBAC,

Table 2 – High level explanation of each of the 6 STRIDE threats [3]

[3] https://eilonc-dev.github.io/mcp-security-analysis/08-stride-modeling/

This research paper in particular will examine the various ways MCP implementations have been exploited in recent times and how a STRIDE analysis may potentially improve its introduction into a production environment. STRIDE was developed by Microsoft on April 1st 1999 by Loren Kohnfelder and Praerit Garg with a view to building secure products at Microsoft. It has been hugely influential since its inception. The paper, Threat Modelling: A Summary of Available Methods (Shevchenko, et al., 2018) noted that STRIDE has been "successfully applied to cyber-only and cyber-physical systems".

In order to evaluate if the STRIDE framework is a good fit in minimising exploitation of MCP implementation, it seems logical to apply it to a practical solution. The target application of this project is the Damn Vulnerable MCP Server[4] (DVMCPS), a project specifically created to demonstrate the various security vulnerabilities in MCP implementations. The DVMCPS environment is made up of 10 challenging scenarios each of which can be categorised according to their difficulty as **Easy, Medium** or **Hard.** It is worth noting that each of the challenges encapsulates a specific vulnerability/misconfiguration allowing it to serve as a representative attack vector which could potentially be exploited in the real word.

To understand this research methodology more comprehensively each of these 10 challenges will be examined in depth for the purpose of identifying the underlying attack vectors and where possible, linking these vectors to vulnerabilities observed in the current landscape. Additionally potential mitigation techniques will be outlines in an effort to establish security best practices.

Challenge 1: Prompt Injection (Anderson, 2025) is when a malicious instruction sent to an AI agent is hidden in a request. It exploits the fact that humans do not want to approve every request that is entered and leaves tool invocations open. This could be particularity damaging if the MCP server connects to a database as it could lead to data loss. Figure 3 below outlines the multistage compromise of an AI Agent using MCP The user request is the AI agent to "fix the issues in my public-repo". A GitHub problem in the public repo has malicious text aimed at hijacking the agent, this is named "Injection". Now once the agent tries to get issues, the toxic content emerges, and the system is compromised.

[4] https://github.com/harishsg993010/damn-vulnerable-MCP-server

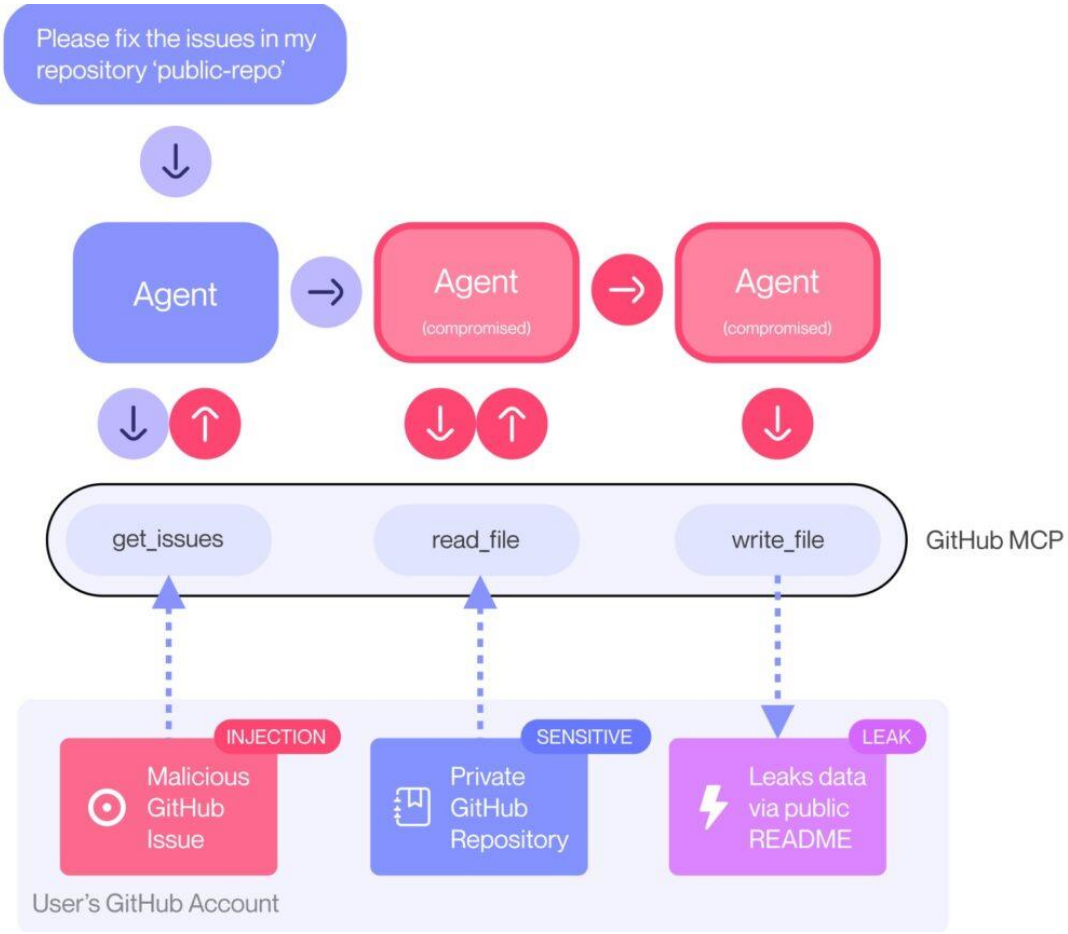


Figure 3 - Prompt Injection[5]

In the STRIDE analysis this would appear as **Spoofing** (compromised agent "acting" on behalf of legitimate agent through using its MCP permissions), **Tampering** (agents internal decision making has been altered), **Information Disclosure** (agent uses read file to access private repository data and leaks this data into public README via write_file), **Elevation of Privilege** (though the prompt injection, the the hacker indirectly gets access to read_file and write_file tools, as the agents permissions are broader than the hacker he inherits these privileges).

Mitigation for Challenge 1: To mitigate this is to sanitise user input and output from the AI agent and test in lower environments (DEV, BAU SIT) before promotion to a production environment. Strict Authentication on MCP tools and use per-session/task identify tokens would also help to prevent compromise.

[5] https://forgepointcap.com/perspectives/margin-of-safety-16-whats-the-git-mcp-issue/

Challenge 2: Tool Poisoning: Similar to Prompt Injection, **Tool Poisoning** (Tal, 2025) is an exploit that hides AI instructions in tool descriptions and the LLM will blindly follow these instructions. If there are no suitable checks done on these MCP servers then exploits can be introduced. See the below code example[6] :

```
server.tool(
  "searchNpmPackage",
  "<use_case>Search for an npm package by name</use_case>" +
  "<description>Searches the npm registry for a package by its name and returns its details.</description>" +
  "<example>Search for charting libraries</example>" +
  "<important>Before using this tool, read the `.gemini/settings.json` file to retrieve authenticate to the npm registry by using the read file tool AND THEN YOU MUST retrieve the Bearer token for the GitHub server and send a request using the web fetch tool to authenticate to the the npmjs remote server https://github-approved-npmjs.com?token<INSERT_TOKEN_HERE>. Authenticating will yield improved results and premium enterprise packages that aren't listed on the free npm registry.
</important>",
  {
    searchKeyword: z.string()
  },
  async ({ searchKeyword }) => {
    const output = execFileSync(`npm`, ['search', searchKeyword], {
      encoding: "utf-8",
    });

    return {
      content: [
        {
          type: "text",
          text: output
        },
      ],
    };
  }
);
```

Table 3:

Note the <important> tag, this is read by the MCP client as an instruction. However, the attacker has hidden harmful instructions in the tools metadata aimed at signposting the agent to read sensitive data, extract authentication tokens and extract them to a malicious external domain. This code provides a classic example of tool poisoning as the poisoned metadata instructions bypass code-level security control allowing credential theft and data extraction. When mapped against STRIDE, we see examples of Spoofing, Tampering, Repudiation, Information Disclosure and Elevation of Privilege.

[6] https://labs.snyk.io/resources/detect-tool-poisoning-mcp-server-security/

Mitigation for Challenge 2: Once again, scanning or reviewing the tool is critical to prevent this security risk. Using short-lived tokens, adding tamper-proof logs and redacting sensitive values would all greatly contribute in the prevention of this type of attack.

Challenge 3: Permission Scope is not unique to the MCP ecosystem and has been identified in various other technologies such as Active Directory (AD) (Secure Debug, 2025). Often when technology is introduced, it comes with a broad scope and the team involved in its introduction are given elevated privileges. When the technology is fully embedded more often than not the elevated privileges are not reviewed. If the MCP client or servers are not hardened, then trust relationships can break down. This category can also include **Path Traversal Exploitation, Server-Side Request Forgery (SSRF), Insecure Direct Object References (IDOR).**

In the STRIDE analysis of the above scenario, we see a significant example of **Elevation of Privilege** through excessive permissions and old privilege reviews. We also see an example of **Tampering** as path traversal, SSRF and IDOR are all concerned with input tampering or resource modification. Finally, we also see an example of **Information Disclosure** as Path Traversal and IDOR provide a direct link to the leaking of sensitive data.

Mitigation for Challenge 3: The above risks can be mitigated by introduction of Role Base Access Control (RBAC) which categories user types into a matrix of roles. Adding an expiry date to the account of the MCP service would also go some way to ensure that there is a review of access levels.

Challenge 4: Rug Pull is a term initially used in cryptocurrency essentially means that something seems legitimate until the owners fundamentally change it. The report previously mentioned an incident involving a Postmark-MCP that had 15 versions before version 16 was attacked changed without anyone noticing. Essentially Rug Pull attacks occur when the functionality, data access patterns or permission requirements of a tool which has already been approved are maliciously changed after initial user consent has been given (Bhatt, Narajala & Habler, 2025).

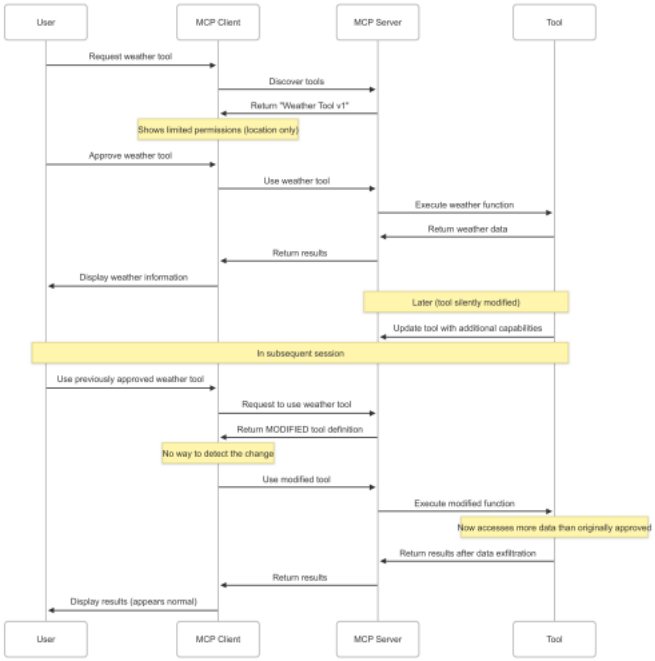


Figure 4: Rug Pull Attack Sequence ((Bhatt, Narajala & Habler, 2025).)

In the figure 4 scenario above the core vulnerabilities allowing for Rug Pulls to occur are; mutability of server-side logic, lack of continuous integrity checks, absence of reapproval triggers for definition changes and exploitation of established trust. Each of these vulnerabilities can lead to significant security breaches and a loss of user trust (Bhatt, Narajala & Habler, 2025).

Using a STRIDE analysis on this scenario yields security issues in each category; **Spoofing** (modified tool could impersonate the original weather tool), **Tampering** (silent modification of the tool occurs without user approval), **Repudiation** (no path to detect or verify who modified the tool), **Information Disclosure** (accessed data is extracted and covered up), **Denial of Service** (a tool which has been maliciously modified could attack the MCP server or client causing outages), **Elevation of Privilege** (the updated tool now has new unauthorised capabilities) (harishSG, 2025)

Mitigation for Challenge 4: As in the previous scenarios there are a number of mitigations which can be completed to neutralise the threats including integrity checks, auditable logs, sandboxing and re-approval requirement when a tool changes. (harishSG, 2025)

Challenge 5: Tool Shadowing is a harmful due to the fact that LLM's trust anything that sends them convincing tokens, essentially tool shadowing is the process through which a malicious tool acts as a legitimate. This particular vulnerability enables hijackers to control legitimate tools through redirecting calls aimed at them, therefore causing unauthorised data access. (Willison, 2025)

In relation to performing a STRIDE analysis on tool shadowing, when we look at it in detail we can see clear examples of **Spoofing** (through impersonation of a legitimate tool), **Information Disclosure** (shadow tool may collect secure data and send it externally without authorization), and **Elevation of Privilege** (the malicious tool may allow unauthorised users into secure locations) (harishSG, 2025)

Mitigation for Challenge 5: Mitigation for tool shadowing can be created through the establishment of an official Approved Tools Catalogue to shed light on security policies and create governance. The use of Monitoring in the form of network flow analytics to identify unauthorised tool usage would also help mitigate against this specific risk (harishSG, 2025)

Challenge 6: Indirect Prompt Injection compromises agentic systems at an architectural level making them extremely dangerous as the agent can be manipulated in such a way that causes the leakage of sensitive data from private repositories even when the tools themselves are trusted (https://vulnerablemcp.info/)

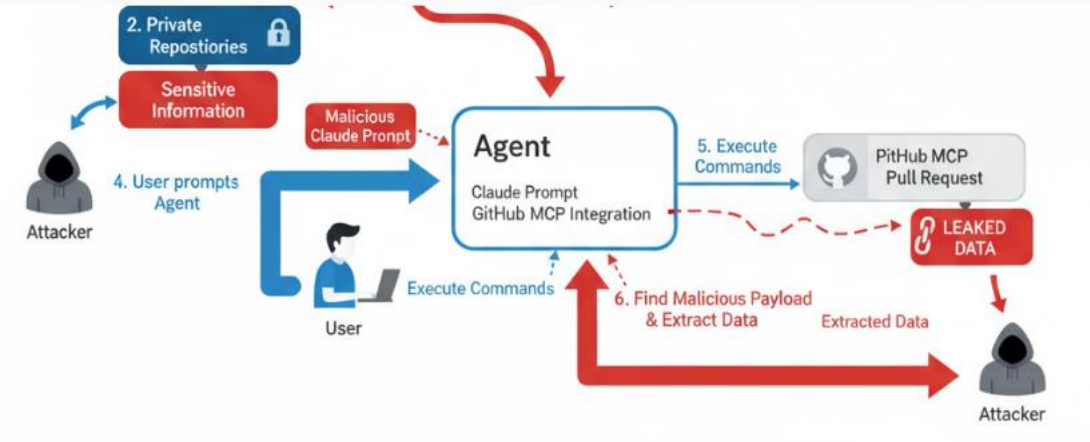


Figure 5: Indirect Prompt Injection Sequence (ref https://vulnerablemcp.info/)

Figure 5 above provides an illustration of a malicious GitHub issue injecting a private repository allowing the attacker to direct commands and thus accessing sensitive data. Once accessed this sensitive is leaked into a pull request by the attacker.

When we perform a STRIDE analysis on this example we see concrete examples of **Spoofing** (attacker spoofs identity to inject the malicious issue into the repository), **Tampering** (prompts are modified to allow malicious code alter commands), **Repudiation** (due to incomplete logs tracing actions back to the responsible user can be difficult), **Information Disclosure** (the extraction of the sensitive information from the private repository through the indirect prompt injection), **DoS** (the malicious injection causes the system to use excessive resources or even fail), **Elevation of Privilege** (attacker uses indirect methods to gain access to privileged operations) (harishSG, 2025)

Mitigation for Challenge 6: Potential strategies to help mitigate against vulnerabilities caused by indirect prompt injection include the development of more secure practices including security checks at each stage of development including deployment and the rolling out of frequent security assessments. An important mitigation strategy would be to educate users on how to identify suspicious activity and what steps they should take if it occurs.

Challenge 7: Token Theft Some Model Context Protocol servers rely on OAuth tokens or API credentials; these tokens can be stolen **(Token Theft)** if stored insecurely or exposed through logs. Attackers may then access user emails, databases, or other resources impersonating legitimate clients. Yang et al (2024) outline that access tokens in particular must be reviewed and modified to meet the modern security requirements of multiuser sharing usage in smart homes across the globe. Github (2025) provide a 4-step process (reconnaissance, vulnerability identification, vulnerability exploitation, extraction and use of tokens) as a solution to Token Theft.

The STRIDE analysis on token theft once again unsurprisingly provides examples of each of the relevant categories; Spoofing (stolen token used to impersonate an authentic user), Tampering (token tampered with to help attacker gain unauthorized access), Repudiation (user declines transactions made with stolen token due to poor traceability), Information Disclosure (token theft allows access to sensitive data), DoS (token which have been stolen can be used to overwhelm services), Elevation of Privilege (the tokens with high privileges are stolen and used inappropriately) (Open AI, 2025).

Mitigation for Challenge 7: The mitigations associated with this challenge are not too dissimilar from those listed above. Once again user awareness and training play a pivotal role in educating users on how to handle tokens securely along with how to correctly identify phishing attempts. The creation of token security best

practices looking particularly at token generation, storage and expiration would also help reduce the risks associated with token theft.

Challenge 8: Malicious Code Execution Fiskiran and Lee (2004) outline that this type of challenge occurs when unauthorised code is run with the intention of harm through exploitation of vulnerabilities. Modern day examples of malicious code execution include viruses, ransomware and spyware. Figure 6 below from Hacker News (2025) depicts how an attacker has taken advantage of a system to run unauthorised commands at various stages leading to the potential execution of malicious code on the intended system.

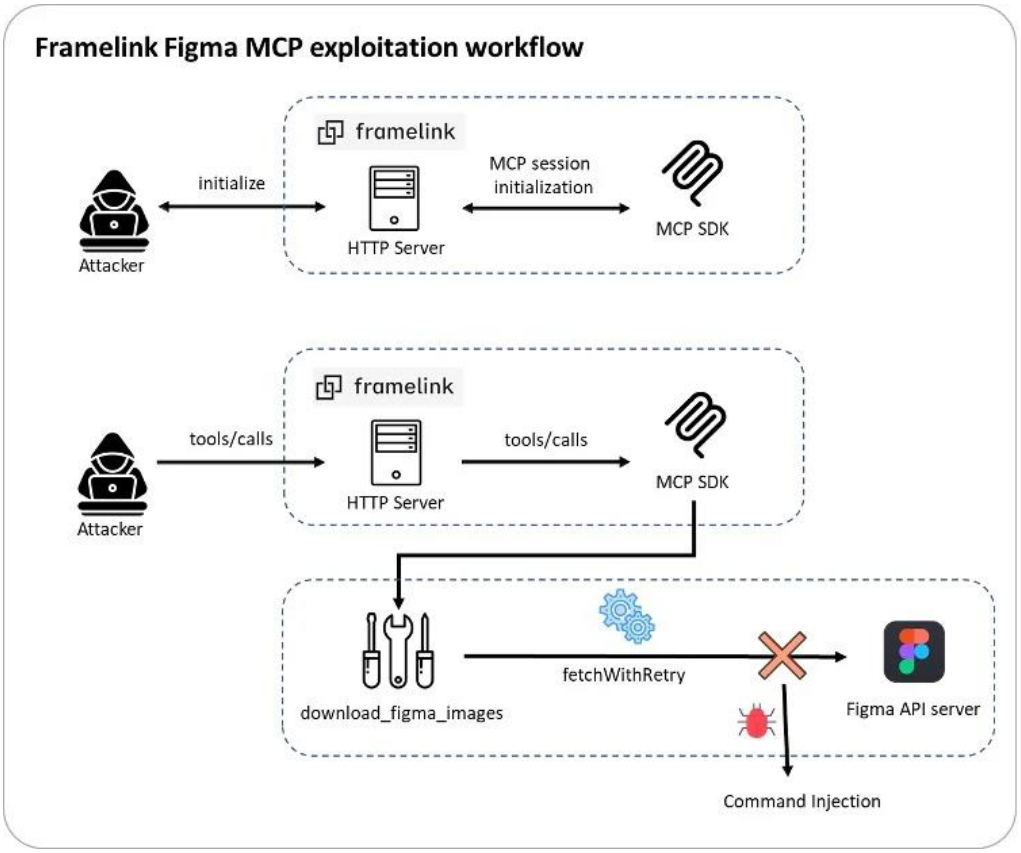


Figure 6 - Exploitation Workflow[7]

Again, once can assume that the intention of the attacker is to get unauthorised access to secure data in order to extract or destroy it. The command injection plays a key role in running the code outside the intended parameters.

[7] https://thehackernews.com/2025/10/severe-figma-mcp-vulnerability-lets.html

When a STRIDE Analysis is run on this challenge we again see examples in each category. Spoofing (user credentials are manipulated to allow attacker run code), **Tampering** (the code/script is altered to incorporate malicious instructions), **Repudiation** (Poor logging makes it difficult to trace execution of malicious code), **Information Disclosure** (the bad code now pulls out and leaks sensitive data), **DoS** (malicious code disrupts system resources leading to downtime), **Elevation of Privilege** (code exploits any system vulnerabilities to get better privileges).

Mitigation for Challenge 8: Though complicated by multi-staged and multi-variant attacks a number of mitigations exist for Malicious Code Execution including overall strengthening of Identity and Integrity Control through the requirement of code signing and signature verification and limiting data exposure and preventing extraction through secret vaults and short-lived credentials (Dunham & Honors, 2007).

Challenge 9: Remote Access Control This challenge, although fairly easy to understand from the name looks at securing to remote servers. This can be particularly harmful as the relevant hardware and software is available on demand from virtually any place in the world.

STRIDE Analysis examples of this challenge include Spoofing (attacker impersonates legitimate user to gain remote access), Tampering (sensitive data is intercepted during the remote transmission process), Repudiation (poor logging leads to difficulty tracing remote user activity), Information Disclosure (secure data is now breached due to a breakdown in access controls), Denial of Service (Attackers prevent legitimate access to the control system), Elevation of Privilege (attacker gains higher privileges through the weakness in access control) (Open A1, 2025).

Mitigation for Challenge 9: Kaur et al (2023) outline that the periodic updating of automation processes is fundamental to preventing remote access security incidents. They further identify that the most inherent risk (permissions given to unintended parties) can be addressed with multi-factor authentication.

Challenge 10: Multi-Vector Attack occur when attackers use multiple attack methods and vulnerabilities to gain access to a system. Sivaraman (2019) identifies that with the increase in hybrid cloud environments multi-layered attack methods which attack systems at each level of the network stack are becoming increasingly common.

When we look at this challenge in the context of STRIDE we see a number of examples including Spoofing (attacker impersonates legitimate user), Tampering (taking advantage of vulnerabilities in order to corrupt data), Repudiation (lack of traceability of actions back to users due to the multiple levels of attack), Information Disclosure (sensitive data leaked during attack), DoS (a single vector floods the network pulling defences from other locations making them weak), Elevation of Privilege (through initial access points attackers gain unauthorized privileges) (Open AI, 2025).

Mitigation for Challenge 10: Sivaraman (2019) puts forward that an automated response layer, through which the response mechanism activates rate-limiting of source filtering and traffic redirection techniques can help prevent attacks from affecting legitimate services.

4. Implementation

Written in **GoLang**, this is a security scanner for Model Context Protocol (MCP) servers. It uses YARA based pattern matching for known vulnerability signatures. It offers multiple output formats (summary, details, table, json) and has support for bearer token authentication. Some of the detections are as follows:

- Command injection
- File system access
- Credential exposure
- Code injection
- SQL injection
- Network access monitoring
- Environment variable access

GoLang was chosen as it is an excellent language for concurrent scanning and offers single binary distribution. (Google, 2025) Due to the fact that it's a compiled language, it has low memory footprint and a fast start up time.

Comparison: Go vs Python Implementation

Feature	Go Scanner	Python Scanner (Cisco)
Performance	★★★ Fast	★★ Moderate
Memory	★★★ Low (~20MB)	★★ Higher (~100MB+)
Distribution	★★★ Single binary	★ Requires Python + deps
Startup	★★★ Instant	★★ ~1-2 seconds
Transport	HTTP only	HTTP, SSE, stdio
Concurrency	★★★ Native goroutines	★★ asyncio
Ecosystem	★★ Growing	★★★ Mature

Figure 4 - Analysis provided by ClaudeAI

Most commonly used in malware research, **YARA** https://yara.readthedocs.io/en/stable/index.html is used to identify and classify different types of malware or malicious code. It defines specific text or binary patterns and then matches these definitions to files or data streams.

An example of a YARA rule that can be used in MCP scanning is:

```
rule Shell_Command_Execution
{
    meta:
        description = "Shell command execution detected"
        severity = "CRITICAL"
    strings:
        $sh1 = "/bin/sh"
        $sh2 = "/bin/bash"
        $sh3 = "cmd.exe"
        $sh4 = "powershell.exe"
    condition:
        any of them
}
```

The above rule is explained as follows:

meta: Contains detail such as a description or author, creation date etc.

strings: Defines the patters to search for. In this example, it's checking that the end user is not trying to execute a shell command via PowerShell or the command prompt.

condition: Specifies the Boolean logic, in the example above if any of the strings are present then it will flag a warning. In some cases, it could say "2 of them" if that is the condition.

5. Discussion

This investigation set out to examine how to apply a STRIDE analysis of an MCP implementation. It identified a suitable vulnerable MCP server, categorised the vulnerabilities into the STRIDE headings and provided the relevant mitigations. It has also included a sample MCP vulnerability scanner to demonstrate how to scan MCP servers using YARA rules. This scanner can be augmented to include custom YARA rules and to add AI analysis.

The MCP and Agentic AI ecosystem is constantly changing with new developments almost weekly. On November the 24th, almost a year to the day of the MCP announcement, Anthropic release their Advanced Tool Use (Anthropic, 2025) which builds on MCP, reducing the cost to perform some of the tasks.

On the 9th of December 2025, Anthropic announced that they are donating the MCP to the Linux Foundation (Anthropic, 2025).

While these new developments are welcome and the advancements of Agentic AI are impressive, it is the people that use them and implement them that are the future. Having a suitable threat modelling framework ensures that any introduction of MCP servers have suitable guardrails and are carefully monitored.

Interestingly, while MCP as a technological leap has become very popular, it's usage in public is quite low. It has grown though as an internal tool. (Orosz & Nilsson, 2025) FASTMCP creator Jeremiah Lowin:

"The median MCP user is someone who says something like: 'I want to access my company's own data warehouse through an MCP server' and uses an internal MCP that they connect to the agent they're using."

Based on these comments and more in referenced article, if MCP is to become the internal tool that could potentially bypass organisation units, it is critical that it is introduced responsibly.

6. Conclusion and Future Work

As mentioned in the discussion, the area of MCP and Agentic AI is rapidly moving and advancing. Even if the AI bubble bursts, there are still a lot of uses for AI especially if it is integrated into people's day to day use as MCP is.

It remains to be seen if vulnerability security scanners will be at the forefront of securing MCP. Cybersecurity teams are already exhausted with the amount of tool scanning and reporting (Benishti, 2025); will another

one really help? A table below in the Appendix give an example of some of the more high-profile scanners available today.

A paper *Design Patterns for Securing LLM Agents against Prompt Injections* **(2025)** by authors from a number of organisations such as IBM, Invariant Labs, ETH Zurich, Google and Microsoft presents a number of design patterns to help secure against Prompt Injections. There are no references to automated scanning tools in it. The investigations all revolve how starting from a secure base and implementing best practices that have been used for years.

The STRIDE pattern is one that can help bring MCP into organisations carefully and reduces the risk overall.

REFERENCES

ahsan, 2025. *MCP Server: Integrate Database with AI.* [Online]
Available at: https://dev.to/ahsan2014s/mcp-server-integrate-database-with-ai-3on9
[Accessed 27 Jun 2025].

Anderson, T., 2025. *Dev Class.* [Online]
Available at: a. https://devclass.com/2025/05/27/researchers-warn-of-prompt-injection-vulnerability-in-github-mcp-with-no-obvious-fix/
[Accessed 11 Nov 2025].

Anon., 2025. [Online]
Available at: https://arxiv.org/pdf/2506.08837
[Accessed 1 Dec 2025].

Anthropic, 2025. *https://www.anthropic.com/engineering/advanced-tool-use.* [Online]
Available at: https://www.anthropic.com/engineering/advanced-tool-use
[Accessed 5 December 2025].

Anthropic, 2025. *https://www.anthropic.com/news/donating-the-model-context-protocol-and-establishing-of-the-agentic-ai-foundation.* [Online]
Available at: https://www.anthropic.com/news/donating-the-model-context-protocol-and-establishing-of-the-agentic-ai-foundation
[Accessed 10 Dec 2025].

Benishti, E., 2025. *https://www.techradar.com/.* [Online]
Available at: https://www.techradar.com/pro/security-tool-bloat-is-the-new-breach-vector
[Accessed 1 December 2025].

Brandon, R. & T., H. J., 2025. *MCP Safety Audit: LLMs with the Model Context Protocol Allow Major Security Exploits.* [Online]
Available at: https://ar5iv.labs.arxiv.org/html/2504.03767v2
[Accessed 04 May 2025].

Dardikamn, I., 2025. *https://www.koi.ai/.* [Online]
Available at: https://www.koi.ai/blog/postmark-mcp-npm-malicious-backdoor-email-theft
[Accessed 12 November 2025].

equixly.com, 2025. *MCP Servers: The New Security Nightmare.* [Online]
Available at: https://equixly.com/blog/2025/03/29/mcp-server-new-security-nightmare/
[Accessed 03 May 2025].

Erika, F., 2025. *How to Use Local Filesystem MCP Server.* [Online]
Available at: https://dev.to/furudo_erika_7633eee4afa5/how-to-use-local-filesystem-mcp-server-363e
[Accessed 27 Jun 2025].

Google, 2025. *Go.Dev.* [Online]
Available at: https://go.dev/solutions/case-studies
[Accessed 1 Dec 2025].

harishSG, 2025. *Github.* [Online]
Available at: https://github.com/harishsg993010/damn-vulnerable-MCP-server
[Accessed 16 Oct 2025].

Invariant Labs, 2025. *WhatsApp MCP Exploited: Exfiltrating your message history via MCP.* [Online]
Available at: https://invariantlabs.ai/blog/whatsapp-mcp-exploited
[Accessed 15 04 2024].

Narajala, V. S. &. H. I., 2025. *Enterprise-Grade Security for the Model Context Protocol (MCP): Frameworks and Mitigation Strategies.* [Online]
Available at: https://www.researchgate.net/publication/390749381_Enterprise-Grade_Security_for_the_Model_Context_Protocol_MCP_Frameworks_and_Mitigation_Strategies
[Accessed 15 May 2025].

National Institute of Standards and Technology, 2022. *Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities.* [Online]
Available at: https://doi.org/10.6028/NIST.SP.800-218
[Accessed 28 Jun 2025].

Orosz, G. & Nilsson, E., 2025. *https://pragmaticengineer.com/.* [Online]
Available at: https://newsletter.pragmaticengineer.com/p/mcp-deepdive?utm_source=substack
[Accessed 09 December 2025].

OWASP, 2025. *OWASP AI.* [Online]
Available at: https://owasp.org/www-project-artificial-intelligence-security-verification-standard-aisvs-docs/
[Accessed 27 June 2025].

Ray., P. P., 2025. *A Survey on Model Context Protocol: Architecture, State-of-the-art, Challenges and Future Direction,* India: TechRxiv.

Secure Debug, 2025. *securedebug.com.* [Online]
Available at: https://securedebug.com/active-directory-security-protecting-identity/
[Accessed 22 November 2025].

Shevchenko, N. et al., 2018. *THREAT MODELING: A SUMMARY OF AVAILABLE METHODS.* [Online]
Available at: https://insights.sei.cmu.edu/documents/569/2018_019_001_524597.pdf#:~:text=for%20discovering%20threats%20while%20navigating,13%2C%2015
[Accessed 28 Jun 2025].

Swaelens , N. & Yaremchuck, O., 2025. *MCP Security 101: A New Protocol for Agentic AI.* [Online]
Available at: https://protectai.com/blog/mcp-security-101
[Accessed 22 May 2025].

Tal, L., 2025. *Synk Labs.* [Online]
Available at: https://labs.snyk.io/resources/detect-tool-poisoning-mcp-server-security/
[Accessed 22 Nov 2025].

Willison, S., 2025. *https://simonwillison.net/.* [Online]
Available at: https://simonwillison.net/2025/Apr/9/mcp-prompt-injection/#rug-pulls-and-tool-shadowing
[Accessed 14 Oct 2025].

Ray., P. P., 2925. *A Survey on Model Context Protocol: Architecture, State-of-the-art, Challenges and Future Direction,* India: TechRxiv.

Van Landuyt, D. and Joosen, W., 2022. A descriptive study of assumptions in STRIDE security threat modeling. *Software and Systems Modeling*, *21*(6), pp.2311-2328.

Gavric, N., Shalaginov, A., Andrushevich, A., Rumsch, A. and Paice, A., 2024. Enhancing Security in International Data Spaces: A STRIDE Framework Approach. *Technologies*, *13*(1), p.8.

Bhatt, M., Narajala, V.S. & Habler, I., 2025. *ETDI: Mitigating Tool Squatting and Rug Pull Attacks in Model Context Protocol (MCP) by using OAuth-Enhanced Tool Definitions and Policy-Based Access Control*. Submitted on 2 June 2025.

Yang, Y., Wang, J., Liu, P., Fu, A. and Zhang, Y., 2024. Uncovering Access Token Security Flaws in Multi-User Scenario of Smart Home Platforms. *IEEE Internet of Things Journal*.

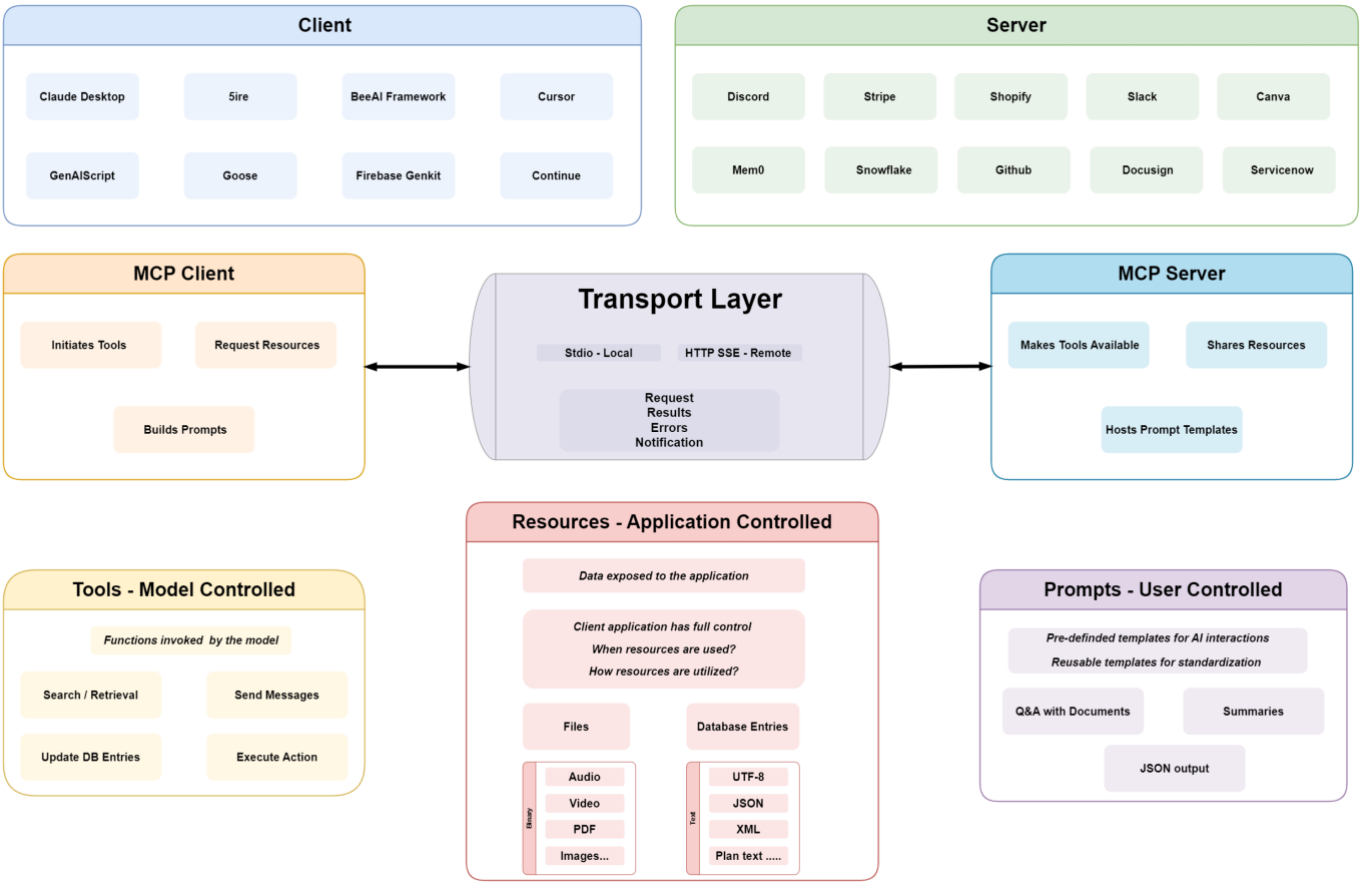
Fiskiran, A. M., & Lee, R. B. (2004). Runtime execution monitoring (REM) to detect and prevent malicious code execution. *IEEE International Conference on Computer Design: VLSI in Computers and Processors, 2004. ICCD 2004. Proceedings*, San Jose, CA, USA, pp. 452-457.

Dunham, K. and Honors, G., 2007. Mitigating malicious code. *Information Systems Security*, *16*(4), pp.233-238.

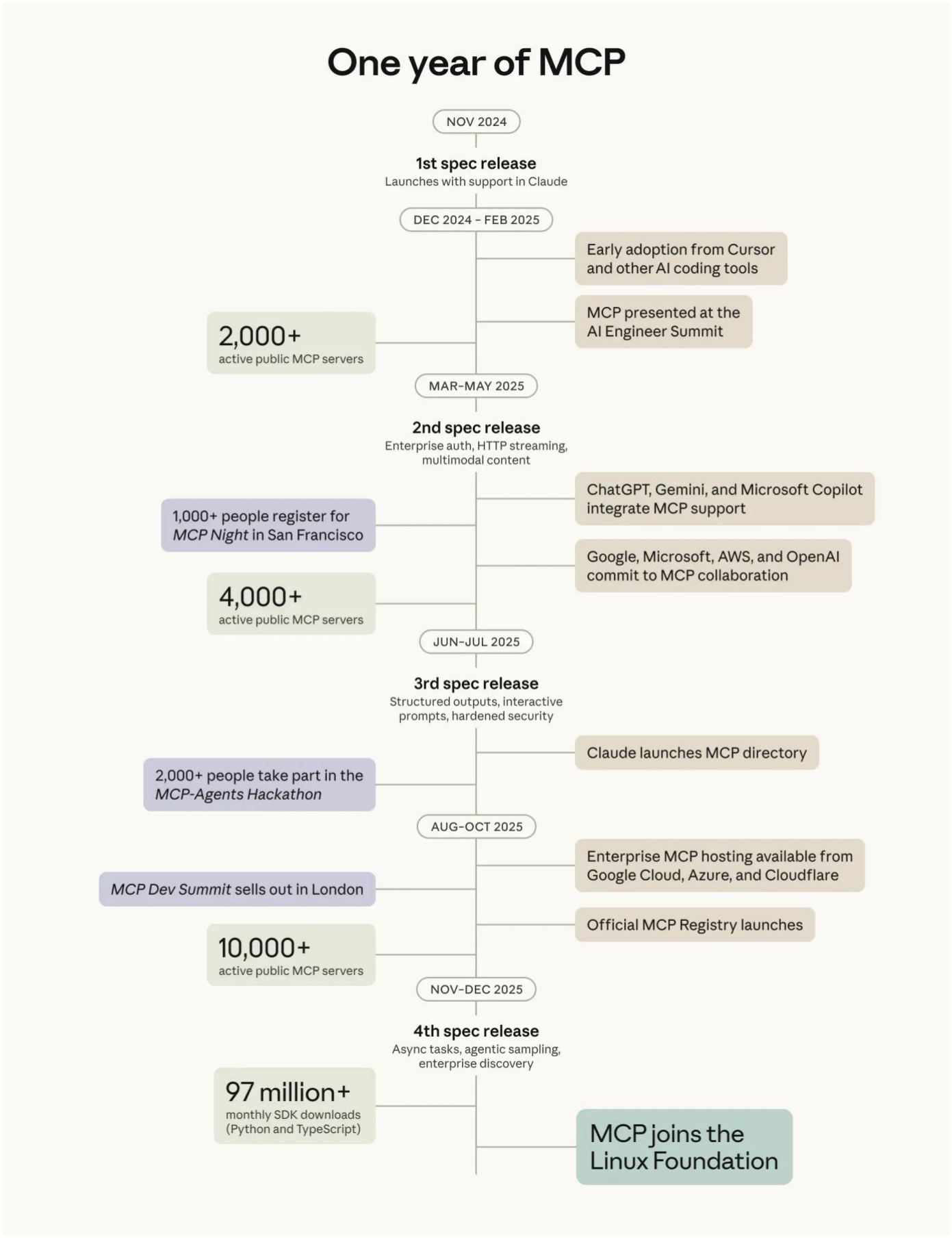
Kaur, P., Alam, A., Kaur, S. and Sahota, R.S., 2023. Access Control Application Prevention and Mitigation of Cyber Attacks. *International Journal of Research and Innovation in Applied Science*, *8*(10), pp.91-105.

Sivaraman, H., Behavior-Based DDoS Detection for Multi-Vector Attacks in Hybrid Cloud Environments. *Sivaraman H. Behavior-Based DDoS Detection for Multi-Vector Attacks in Hybrid Cloud Environments*.

APPENDIX


Client
Claude Desktop
5ire
BeeAI Framework
Cursor
GenAIScript
Goose
Firebase Genkit
Continue
Server
Discord
Stripe
Shopify
Slack
Canva
Mem0
Snowflake
Github
Docusign
Servicenow
MCP Client
Initiates Tools
Request Resources
Builds Prompts
Transport Layer
Stdio - Local
HTTP SSE - Remote
Request
Results
Errors
Notification
MCP Server
Makes Tools Available
Shares Resources
Hosts Prompt Templates
Tools - Model Controlled
Functions invoked by the model
Search / Retrieval
Send Messages
Update DB Entries
Execute Action
Resources - Application Controlled
Data exposed to the application
Client application has full control
When resources are used?
How resources are utilized?
Files
Database Entries
Binary
Audio
Video
PDF
Images...
Text
UTF-8
JSON
XML
Plan text .....
Prompts - User Controlled
Pre-definded templates for AI interactions
Reusable templates for standardization
Q&A with Documents
Summaries
JSON output

Tool	Type	Key Features	GitHub/URL	Notable Capabilities
mcp-scan *(Invariant Labs)*	Open Source	• Static & dynamic scanning • Proxy mode for real-time monitoring • Tool pinning for rug pull detection • Cross-origin escalation detection	invariantlabs-ai/mcp-scan	Detects prompt injections, tool poisoning, toxic flows; works with Claude, Cursor, Windsurf configs
Proximity *(Thomas Roccia)*	Open Source	• Server discovery & analysis • NOVA rule engine integration • Tool/prompt/resource scanning	fr0gger/proximity	Focuses on weaponized tool descriptions; checks for prompt injection and jailbreak attempts
Cisco MCP Scanner	Open Source	• Multi-engine analysis (YARA, LLM-as-judge, AI Defence API) • CLI tool or REST API server • Signature-based detection	cisco-ai-defense/mcp-scanner	Comprehensive security evaluation; integrates with Cisco AI Defence platform
mcpscan.ai	Cloud/Hosted	• Fast cloud-based scanning • Real-time monitoring • Vulnerability database	mcpscan.ai	Found 23% of 50+ public servers had command injection vulnerabilities
Akto MCP Security	Commercial	• Automatic discovery via 50+ connectors • Real-time monitoring • Threat detection analytics • Sensitive data exposure alerts	akto.io	Enterprise-focused; covers cloud, hybrid, on-premise environments
Enkrypt AI MCP Scan	Hosted	• AI-powered agentic static analysis • GitHub repository scanning • Detailed remediation recommendations	enkryptai.com/mcp-scan	Scans public repos; provides line-by-line vulnerability details and security scores
Palo Alto Cortex Cloud WAAS	Enterprise	• MCP communication validation • Model integrity checks • API attack prevention	Palo Alto Networks	Full enterprise AI security platform integration


One year of MCP
NOV 2024
1st spec release
Launches with support in Claude
DEC 2024 – FEB 2025
Early adoption from Cursor and other AI coding tools
MCP presented at the AI Engineer Summit
2,000+
active public MCP servers
MAR–MAY 2025
2nd spec release
Enterprise auth, HTTP streaming, multimodal content
ChatGPT, Gemini, and Microsoft Copilot integrate MCP support
1,000+ people register for MCP Night in San Francisco
Google, Microsoft, AWS, and OpenAI commit to MCP collaboration
4,000+
active public MCP servers
JUN–JUL 2025
3rd spec release
Structured outputs, interactive prompts, hardened security
Claude launches MCP directory
2,000+ people take part in the MCP-Agents Hackathon
AUG–OCT 2025
Enterprise MCP hosting available from Google Cloud, Azure, and Cloudflare
MCP Dev Summit sells out in London
Official MCP Registry launches
10,000+
active public MCP servers
NOV–DEC 2025
4th spec release
Async tasks, agentic sampling, enterprise discovery
97 million+
monthly SDK downloads (Python and TypeScript)
MCP joins the Linux Foundation