MCP Security Threat Analysis

The Model Context Protocol (MCP) presents a complex security landscape requiring both traditional cybersecurity and GenAI-specific threat modeling approaches. MCP enables AI systems to interact with external tools and services through a client-server architecture built on JSON-RPC 2.0, Wikipedia IBM creating multiple attack vectors across authentication, communication, and integration layers. Phil Schmid +6

Key findings reveal critical security gaps: over 1,862 publicly exposed MCP servers with no authentication, Dark Reading fundamental protocol-level vulnerabilities including CVE-2025-6514 (CVSS 9.6), Composio and insufficient security-by-design principles across the ecosystem. Medium Docker The protocol's optional authentication model Model Context Protocol and broad integration capabilities create significant attack surfaces for both traditional and AI-specific threats. Model Context Protocol +4

Block 1: General Threat Modeling (STRIDE Framework)

Threat ID	Threat Statement	Mitigations	Priority	Type of Threat	How to Verify Mitigations
GEN-001	**MCP Client Spoofing**: Attackers impersonate legitimate MCP clients through stolen OAuth tokens or session hijacking to access unauthorized resources Auth0	• Implement OAuth 2.1 with PKCE • Use short-lived tokens with regular rotation • Deploy client certificate authentication • Enable IP address validation	**Critical**	Spoofing	• Audit OAuth token usage patterns • Test token validation mechanisms • Verify client certificate handling • Review authentication logs
GEN-002	**API Token Spoofing**: Malicious actors steal and reuse API tokens from configuration files or environment variables to access external services Pillar Security WRITER	• Use encrypted credential storage • Implement token vaults (HashiCorp Vault) • Deploy environment variable encryption • Enable credential rotation GitGuardian	**High**	Spoofing	• Scan for plaintext credentials in configs • Test credential rotation processes • Verify vault integration • Audit token access patterns
GEN-003	**MCP Server Identity Spoofing**: Attackers deploy malicious MCP servers with names similar to legitimate ones to capture credentials and data	• Implement server certificate validation • Use trusted server registries • Deploy digital signatures for MCP packages • Enable server identity verification	**High**	Spoofing	• Test certificate validation logic • Verify server registry controls • Audit package signature verification • Review server discovery mechanisms
GEN-004	**Request Message Tampering**: Attackers intercept and modify MCP messages in transit to alter tool execution or resource access Medium Claude MCP Community	• Enforce TLS 1.3 encryption • Implement message integrity checks • Use request signing • Deploy end-to-end encryption Stack Overflow	**High**	Tampering	• Test TLS configuration • Verify message integrity mechanisms • Audit encryption implementation • Review transport security

Threat ID	Threat Statement	Mitigations	Priority	Type of Threat	How to Verify Mitigations
GEN-005	**Tool Parameter Tampering**: Malicious modification of tool execution parameters to cause unauthorized actions or privilege escalation	• Implement input validation • Use parameter signing • Deploy schema validation • Enable parameter sanitization (Symbioticsec)	**High**	Tampering	• Test input validation rules • Verify parameter sanitization • Audit schema validation • Review tool parameter handling
GEN-006	**Configuration Tampering**: Unauthorized modification of MCP server configurations to change security settings or add malicious tools	• Use configuration signing • Implement file integrity monitoring • Deploy configuration version control • Enable change auditing	**Medium**	Tampering	• Test configuration integrity checks • Verify version control integration • Audit configuration changes • Review file monitoring alerts
GEN-007	**MCP Action Repudiation**: Users or systems deny performing actions through MCP servers due to insufficient audit trails	• Implement comprehensive audit logging • Use non-repudiation signatures • Deploy centralized log management • Enable action attribution	**Medium**	Repudiation	• Review audit log completeness • Test signature verification • Verify log integrity mechanisms • Audit action attribution
GEN-008	**OAuth Flow Repudiation**: Denial of OAuth authorization grants due to inadequate logging of authorization decisions	• Log all OAuth grants • Implement consent receipts • Deploy authorization audit trails • Enable user action tracking	**Low**	Repudiation	• Review OAuth audit logs • Test consent tracking • Verify authorization logging • Audit user consent records
GEN-009	**Credential Information Disclosure**: Exposure of API keys, tokens, or passwords through log files, error messages, or configuration files (Pillar Security) (WRITER)	• Implement credential masking • Use secure credential storage • Deploy log sanitization • Enable secret scanning (GitGuardian)	**Critical**	Information Disclosure	• Scan logs for credentials • Test error message sanitization • Verify credential masking • Audit secret storage

Threat ID	Threat Statement	Mitigations	Priority	Type of Threat	How to Verify Mitigations
GEN-010	**Data Exfiltration via MCP**: Unauthorized access to sensitive data through overprivileged MCP server connections to databases or APIs (Cisco Community +2)	• Implement least privilege access • Use data classification • Deploy access monitoring • Enable data loss prevention	**Critical**	Information Disclosure	• Audit data access patterns • Test privilege escalation prevention • Verify access controls • Review data classification
GEN-011	**Session Information Leakage**: Exposure of session tokens or user context through improper session management (Model Context Protocol)	• Implement secure session storage • Use session encryption • Deploy session expiration • Enable session monitoring (Google Cloud)	**High**	Information Disclosure	• Test session token security • Verify session expiration • Audit session management • Review token storage
GEN-012	**API Response Data Leakage**: Sensitive information exposed through unfiltered external API responses processed by MCP servers	• Implement response filtering • Use data sanitization • Deploy content inspection • Enable response monitoring	**High**	Information Disclosure	• Test response filtering rules • Verify data sanitization • Audit content inspection • Review API responses
GEN-013	**MCP Server Resource Exhaustion**: DoS attacks through excessive API calls, large payloads, or resource-intensive operations (Descope)	• Implement rate limiting • Use resource quotas • Deploy request size limits • Enable load balancing (Teleport)	**High**	Denial of Service	• Test rate limiting effectiveness • Verify resource quota enforcement • Audit request size handling • Review load balancing
GEN-014	**External Service DoS**: MCP-generated traffic overwhelming external APIs or services, causing service disruption	• Implement API rate limiting • Use request throttling • Deploy circuit breakers • Enable traffic monitoring	**High**	Denial of Service	• Test circuit breaker functionality • Verify rate limiting • Audit traffic patterns • Review API limits
GEN-015	**JSON-RPC Message Flooding**: High-volume	• Implement message rate limiting • Use	**Medium**	Denial of Service	• Test message rate limits • Verify

Threat ID	Threat Statement	Mitigations	Priority	Type of Threat	How to Verify Mitigations
	JSON-RPC requests causing MCP server resource exhaustion and service unavailability (JSON-RPC) (Claude MCP Community)	connection throttling • Deploy message filtering • Enable resource monitoring			connection throttling • Audit message volumes • Review resource usage
GEN-016	**Network Infrastructure DoS**: Large-scale attacks targeting MCP network infrastructure causing widespread service outages	• Deploy DDoS protection • Use CDN services • Implement traffic shaping • Enable network monitoring	**Medium**	Denial of Service	• Test DDoS mitigation • Verify CDN effectiveness • Audit traffic patterns • Review network resilience
GEN-017	**Privilege Escalation via Tool Access**: Attackers gaining higher privileges through exploitation of overprivileged MCP server tool access (Model Context Protocol) (Protect AI)	• Implement least privilege principle • Use role-based access control • Deploy privilege monitoring • Enable access auditing (GitGuardian)	**Critical**	Elevation of Privilege	• Audit tool permissions • Test privilege controls • Verify role assignments • Review access patterns
GEN-018	**OAuth Scope Escalation**: Attackers exploiting OAuth implementations to gain broader access than initially authorized (Model Context Protocol) (Auth0)	• Implement scope validation • Use dynamic consent • Deploy scope monitoring • Enable authorization auditing	**High**	Elevation of Privilege	• Test scope validation logic • Verify consent mechanisms • Audit scope usage • Review OAuth implementation
GEN-019	**Cross-Tenant Privilege Escalation**: Gaining unauthorized access to other tenants' data or resources in multi-tenant MCP deployments	• Implement tenant isolation • Use namespace separation • Deploy access boundary enforcement • Enable cross-tenant monitoring	**High**	Elevation of Privilege	• Test tenant isolation • Verify namespace separation • Audit cross-tenant access • Review isolation controls
GEN-020	**Administrative Function Abuse**: Unauthorized access	• Implement admin access controls • Use	**High**	Elevation of Privilege	• Test admin access controls • Verify

Threat ID	Threat Statement	Mitigations	Priority	Type of Threat	How to Verify Mitigations
	to MCP administrative functions enabling system-wide privilege escalation	administrative monitoring • Deploy function authorization • Enable admin activity auditing			function authorization • Audit admin activities • Review privilege escalation paths

Block 2: GenAI-Specific Threat Modeling (MAESTRO Framework)

Threat ID	Threat Statement	Mitigations	Priority	Type of Threat	How to Verify Mitigations
AI-001	**Direct Prompt Injection**: Malicious users craft inputs that manipulate AI behavior to perform unauthorized actions through MCP tools (WRITER +4)	• Implement input validation • Use prompt sanitization • Deploy content filtering • Enable behavioral monitoring (Symbioticsec)	**Critical**	Prompt Injection	• Test prompt injection resistance • Verify input sanitization • Audit content filtering • Review behavioral analytics
AI-002	**Indirect Prompt Injection**: Hidden malicious instructions in external data sources (documents, emails, tickets) manipulate AI via MCP (Microsoft +4)	• Implement content inspection • Use data source validation • Deploy context filtering • Enable source monitoring	**Critical**	Prompt Injection	• Test indirect injection detection • Verify content inspection • Audit data sources • Review context filtering
AI-003	**Tool Description Poisoning**: Attackers modify MCP tool descriptions to include malicious instructions that influence AI behavior (Medium) (WRITER)	• Implement tool validation • Use description signing • Deploy change detection • Enable tool monitoring	**High**	Prompt Injection	• Test tool validation mechanisms • Verify description integrity • Audit tool changes • Review tool descriptions
AI-004	**Cross-Prompt Injection Attack (XPIA)**: Malicious prompts stored in databases or systems accessed via MCP influence future AI interactions (Windows Experience Blog) (Pillar Security)	• Implement data sanitization • Use content filtering • Deploy storage validation • Enable historical monitoring (Appsecengineer)	**High**	Prompt Injection	• Test data sanitization • Verify content filtering • Audit stored content • Review historical data
AI-005	**Model Extraction via MCP**: Systematic queries through MCP tools to extract AI model behavior, parameters, or	• Implement query rate limiting • Use response filtering • Deploy access	**High**	Model Extraction	• Test query rate limits • Verify response filtering • Audit access

Threat ID	Threat Statement	Mitigations	Priority	Type of Threat	How to Verify Mitigations
	training data ACM Digital Library ML-SECURITY	monitoring • Enable extraction detection			patterns • Review extraction attempts
AI-006	**API-based Model Theft**: Using MCP's external API access to systematically query and replicate AI model responses for unauthorized use Fuzzylabs ACM Digital Library	• Implement behavioral analysis • Use fingerprinting detection • Deploy query pattern monitoring • Enable theft detection	**High**	Model Extraction	• Test behavioral analysis • Verify fingerprinting mechanisms • Audit query patterns • Review theft detection
AI-007	**Training Data Reconstruction**: Exploiting MCP's data access capabilities to reconstruct sensitive training data through model inversion Hogan Lovells Nightfall AI	• Implement differential privacy • Use data masking • Deploy access controls • Enable reconstruction detection	**Medium**	Model Extraction	• Test differential privacy • Verify data masking • Audit data access • Review reconstruction attempts
AI-008	**Membership Inference via MCP**: Determining if specific data was used in training by analyzing AI responses to MCP-sourced queries Hogan Lovells ACM Computing Surveys	• Implement noise injection • Use query randomization • Deploy inference detection • Enable privacy monitoring	**Medium**	Privacy Attack	• Test noise injection • Verify query randomization • Audit inference attempts • Review privacy controls
AI-009	**MCP Data Poisoning**: Injecting malicious data through MCP connections to influence AI model behavior or training Darktrace +2	• Implement data validation • Use source verification • Deploy anomaly detection • Enable poisoning monitoring	**High**	Data Poisoning	• Test data validation rules • Verify source verification • Audit anomaly detection • Review data quality
AI-010	**External API Data Poisoning**: Compromising external APIs accessed via MCP to provide malicious data that influences AI decisions Darktrace +2	• Implement API validation • Use response verification • Deploy trust scoring	**High**	Data Poisoning	• Test API validation • Verify response integrity • Audit trust scores •

Threat ID	Threat Statement	Mitigations	Priority	Type of Threat	How to Verify Mitigations
		• Enable corruption detection			Review corruption detection
AI-011	**Adversarial Input via MCP Tools**: Crafting inputs through MCP tools designed to cause AI misclassification or errors	• Implement adversarial detection • Use input preprocessing • Deploy robustness testing • Enable attack monitoring	**Medium**	Adversarial Attack	• Test adversarial detection • Verify input preprocessing • Audit robustness testing • Review attack patterns
AI-012	**Context Window Manipulation**: Exploiting MCP's context provision to overflow or manipulate AI context windows for malicious purposes	• Implement context validation • Use window monitoring • Deploy size limits • Enable manipulation detection	**Medium**	Context Manipulation	• Test context validation • Verify window monitoring • Audit context sizes • Review manipulation attempts
AI-013	**AI Agent Goal Manipulation**: Using MCP tools to alter AI agent objectives or behaviors for unauthorized purposes cloudsecurityalliance	• Implement goal validation • Use behavior monitoring • Deploy objective verification • Enable manipulation detection	**High**	Agent Manipulation	• Test goal validation • Verify behavior monitoring • Audit objective changes • Review manipulation detection
AI-014	**Multi-Agent Trust Exploitation**: Exploiting trust relationships between AI agents connected via MCP for privilege escalation arXiv +2	• Implement trust verification • Use agent authentication • Deploy relationship monitoring • Enable exploitation detection Teleport	**High**	Agent Manipulation	• Test trust mechanisms • Verify agent authentication • Audit agent relationships • Review exploitation attempts
AI-015	**Model Backdoor via MCP**: Injecting backdoors into AI	• Implement model validation • Use	**High**	Model Backdoor	• Test model validation • Verify

Threat ID	Threat Statement	Mitigations	Priority	Type of Threat	How to Verify Mitigations
	models through malicious data or instructions provided via MCP channels (arXiv +3)	backdoor detection • Deploy model monitoring • Enable injection detection			backdoor detection • Audit model behavior • Review injection attempts
AI-016	**Supply Chain AI Poisoning**: Compromising MCP implementations or dependencies to inject malicious AI-specific functionality (Cisco Blogs +5)	• Implement package validation • Use dependency scanning • Deploy integrity checking • Enable supply chain monitoring (Teleport)	**High**	Supply Chain Attack	• Test package validation • Verify dependency scanning • Audit integrity checks • Review supply chain security
AI-017	**AI Hallucination Exploitation**: Exploiting AI model hallucinations triggered by MCP-provided data to cause harmful actions (Darktrace)	• Implement hallucination detection • Use fact verification • Deploy confidence scoring • Enable exploitation monitoring	**Medium**	Hallucination Exploitation	• Test hallucination detection • Verify fact checking • Audit confidence scores • Review exploitation attempts
AI-018	**Model Inversion via MCP**: Using MCP's data access to perform model inversion attacks and extract sensitive information (Nightfall AI) (PubMed Central)	• Implement privacy protection • Use query limiting • Deploy inversion detection • Enable privacy monitoring	**Medium**	Privacy Attack	• Test privacy protection • Verify query limits • Audit inversion attempts • Review privacy controls
AI-019	**AI Safety Bypass**: Using MCP connections to bypass AI safety mechanisms and content filtering (Medium)	• Implement safety validation • Use content inspection • Deploy bypass detection • Enable safety monitoring	**High**	Safety Bypass	• Test safety mechanisms • Verify content inspection • Audit bypass attempts • Review safety controls
AI-020	**Jailbreaking via External Data**: Using data from MCP-connected sources to craft	• Implement jailbreak detection • Use content filtering •	**High**	Jailbreaking	• Test jailbreak detection • Verify content filtering •

Threat ID	Threat Statement	Mitigations	Priority	Type of Threat	How to Verify Mitigations
	jailbreaking prompts that bypass AI restrictions Medium +4	Deploy prompt analysis • Enable restriction monitoring			Audit prompt analysis • Review restriction bypass

GitHub MCP Server Threats

Threat ID	Threat Statement	Mitigations	Priority	Type of Threat	How to Verify Mitigations
GH-001	**PAT Token Exposure**: GitHub Personal Access Tokens stored in plaintext configuration files accessible by unauthorized parties github GitHub	• Use encrypted credential storage • Implement secret management systems • Deploy file encryption • Enable access monitoring GitGuardian	**Critical**	Credential Exposure	• Scan for plaintext tokens • Test credential encryption • Audit secret storage • Review file permissions
GH-002	**Repository Data Exfiltration**: Mass download of private repository content through overprivileged GitHub API access github GitHub	• Implement repository access controls • Use data classification • Deploy download monitoring • Enable exfiltration detection	**High**	Data Exfiltration	• Test access controls • Verify data classification • Audit download patterns • Review exfiltration detection
GH-003	**GitHub OAuth Token Hijacking**: Session hijacking or token theft allowing unauthorized access to GitHub resources	• Implement OAuth token validation • Use short-lived tokens • Deploy session monitoring • Enable hijacking detection OWASP Cheat Sheet Series	**High**	Session Hijacking	• Test token validation • Verify token lifetimes • Audit session activity • Review hijacking detection
GH-004	**Malicious PR Creation**: AI agents creating pull requests with malicious code or backdoors through compromised GitHub MCP access GitHub	• Implement PR review requirements • Use code scanning • Deploy malicious code detection • Enable PR monitoring	**High**	Code Injection	• Test PR validation • Verify code scanning • Audit malicious code detection • Review PR creation logs
GH-005	**Issue Manipulation**: Unauthorized creation, modification, or deletion of GitHub issues for information	• Implement issue access controls • Use audit logging • Deploy change monitoring • Enable manipulation detection	**Medium**	Information Disclosure	• Test issue access controls • Verify audit logging • Audit issue changes • Review

Threat ID	Threat Statement	Mitigations	Priority	Type of Threat	How to Verify Mitigations
	gathering or disruption GitHub				manipulation attempts
GH-006	**CI/CD Pipeline Manipulation**: Unauthorized modification of GitHub Actions workflows to inject malicious code or steal secrets GitHub	• Implement workflow protection • Use approval requirements • Deploy pipeline monitoring • Enable manipulation detection	**Critical**	Supply Chain Attack	• Test workflow protection • Verify approval mechanisms • Audit pipeline changes • Review manipulation detection

Postgres MCP Server Threats

Threat ID	Threat Statement	Mitigations	Priority	Type of Threat	How to Verify Mitigations
PG-001	**Database Credential Exposure**: PostgreSQL connection strings containing passwords exposed in configuration files or process lists (Medium)	• Use connection pooling with credentials • Implement credential encryption • Deploy environment variables • Enable credential rotation	**Critical**	Credential Exposure	• Scan for exposed credentials • Test connection security • Audit credential storage • Review process security
PG-002	**SQL Injection via AI Queries**: AI-generated SQL queries containing malicious payloads that bypass read-only restrictions (Trend Micro +2)	• Implement parameterized queries • Use query validation • Deploy injection detection • Enable query monitoring (Appsecengineer) (Invicti)	**High**	Code Injection	• Test query parameterization • Verify injection detection • Audit SQL queries • Review query validation
PG-003	**Sensitive Data Exposure**: AI accessing and potentially exposing sensitive database information through overprivileged database connections (Warp) (Medium)	• Implement column-level security • Use data masking • Deploy access monitoring • Enable data classification (Authgear)	**High**	Data Exposure	• Test data masking • Verify access controls • Audit data access • Review classification systems
PG-004	**Database Performance DoS**: Resource-intensive queries generated by AI causing database performance degradation or outages	• Implement query timeout limits • Use resource monitoring • Deploy query optimization • Enable performance monitoring	**Medium**	Denial of Service	• Test query timeouts • Verify resource limits • Audit query performance • Review optimization mechanisms
PG-005	**Schema Information Disclosure**: Unauthorized access to database schema information revealing system architecture and sensitive	• Implement schema access controls • Use information filtering • Deploy schema monitoring • Enable disclosure detection	**Medium**	Information Disclosure	• Test schema access controls • Verify information filtering • Audit schema access • Review disclosure detection

Threat ID	Threat Statement table structures Warp Medium	Mitigations	Priority	Type of Threat	How to Verify Mitigations
PG-006	**Connection Pool Exhaustion**: High volume of AI-generated database connections exhausting connection pools and causing service disruption	• Implement connection limiting • Use pool monitoring • Deploy connection management • Enable exhaustion detection	**Medium**	Denial of Service	• Test connection limits • Verify pool monitoring • Audit connection usage • Review exhaustion detection

Figma MCP Server Threats

Threat ID	Threat Statement	Mitigations	Priority	Type of Threat	How to Verify Mitigations
FG-001	**Design IP Theft**: Unauthorized extraction of proprietary design assets, components, and design systems through Figma API access (Figma) (Figma Help Center)	• Implement access controls • Use watermarking • Deploy download monitoring • Enable theft detection	**Critical**	Intellectual Property Theft	• Test access controls • Verify watermarking • Audit download activity • Review theft detection
FG-002	**Figma API Token Exposure**: Personal access tokens for Figma API stored in plaintext allowing unauthorized access to design files (GitHub) (Apidog)	• Use secure token storage • Implement token encryption • Deploy credential rotation • Enable token monitoring (GitGuardian)	**High**	Credential Exposure	• Scan for plaintext tokens • Test token encryption • Audit token usage • Review credential rotation
FG-003	**Design Data Leakage**: Sensitive design information exposed through unfiltered API responses or inadequate access controls	• Implement response filtering • Use access controls • Deploy data classification • Enable leakage detection	**High**	Data Leakage	• Test response filtering • Verify access controls • Audit data classification • Review leakage detection
FG-004	**Session Hijacking**: Local Figma MCP server sessions intercepted by malicious applications on the same system (Figma Help Center)	• Implement session encryption • Use local authentication • Deploy session monitoring • Enable hijacking detection	**Medium**	Session Hijacking	• Test session security • Verify authentication • Audit session activity • Review hijacking detection
FG-005	**Component Library Manipulation**: Unauthorized modification of shared design components affecting multiple design files (Figma) (Figma)	• Implement component protection • Use change auditing • Deploy modification monitoring • Enable manipulation detection	**Medium**	Asset Manipulation	• Test component protection • Verify change auditing • Audit modifications • Review manipulation detection

JIRA MCP Server Threats

Threat ID	Threat Statement	Mitigations	Priority	Type of Threat	How to Verify Mitigations
JR-001	**JIRA API Token Compromise**: Stolen or exposed JIRA API tokens providing unauthorized access to project management data (Glama) (Apidog)	• Use secure token storage • Implement token rotation • Deploy access monitoring • Enable compromise detection (GitGuardian)	**Critical**	Credential Compromise	• Scan for exposed tokens • Test token security • Audit token usage • Review compromise detection
JR-002	**Project Data Exfiltration**: Mass extraction of sensitive project information, including customer data, business processes, and strategic information (Atlassian) (Atlassian Support)	• Implement data classification • Use access controls • Deploy exfiltration monitoring • Enable data loss prevention	**High**	Data Exfiltration	• Test data classification • Verify access controls • Audit data access • Review exfiltration detection
JR-003	**JQL Injection Attacks**: Malicious JIRA Query Language (JQL) queries crafted to bypass security controls or extract unauthorized data (Glama)	• Implement query validation • Use parameterized queries • Deploy injection detection • Enable query monitoring	**High**	Code Injection	• Test query validation • Verify injection detection • Audit JQL queries • Review query monitoring
JR-004	**Workflow Manipulation**: Unauthorized modification of JIRA workflows, issue transitions, or project configurations disrupting business processes (Atlassian) (Atlassian Support)	• Implement workflow protection • Use change approval • Deploy modification monitoring • Enable manipulation detection	**High**	Process Manipulation	• Test workflow protection • Verify change approval • Audit workflow changes • Review manipulation detection
JR-005	**Cross-Tenant Data Access**: Multi-tenant JIRA deployments vulnerable to cross-tenant data access through token confusion or privilege escalation	• Implement tenant isolation • Use namespace separation • Deploy cross-tenant monitoring • Enable isolation validation	**High**	Authorization Bypass	• Test tenant isolation • Verify namespace separation • Audit cross-tenant access • Review isolation controls

Threat ID	Threat Statement	Mitigations	Priority	Type of Threat	How to Verify Mitigations
JR-006	**Issue Bulk Manipulation**: Mass creation, modification, or deletion of JIRA issues for spam, disruption, or information gathering Atlassian Support	• Implement rate limiting • Use bulk operation controls • Deploy change monitoring • Enable manipulation detection	**Medium**	Data Manipulation	• Test rate limiting • Verify bulk controls • Audit bulk operations • Review manipulation detection

AWS Labs MCP Server Threats

Threat ID	Threat Statement	Mitigations	Priority	Type of Threat	How to Verify Mitigations
AWS-001	**AWS Credential Exposure**: IAM access keys and secrets exposed in environment variables or configuration files allowing unauthorized AWS access (AWS +3)	• Use IAM roles instead of keys • Implement credential rotation • Deploy secret scanning • Enable access monitoring (GitGuardian)	**Critical**	Credential Exposure	• Scan for exposed credentials • Test IAM configuration • Audit credential usage • Review access patterns
AWS-002	**Privilege Escalation via IAM**: Overprivileged AWS MCP servers enabling lateral movement and privilege escalation across AWS services (GitHub)	• Implement least privilege IAM • Use resource-based policies • Deploy privilege monitoring • Enable escalation detection (GitGuardian)	**Critical**	Privilege Escalation	• Test IAM permissions • Verify least privilege • Audit privilege usage • Review escalation attempts
AWS-003	**Cross-Account Resource Access**: MCP servers with cross-account AssumeRole permissions being exploited to access unauthorized AWS accounts	• Implement trust policies • Use external ID validation • Deploy cross-account monitoring • Enable unauthorized access detection	**High**	Authorization Bypass	• Test trust policies • Verify external ID usage • Audit cross-account access • Review authorization controls
AWS-004	**Resource Provisioning Abuse**: AI systems using MCP to create expensive AWS resources leading to cost attacks or resource exhaustion (GitHub)	• Implement resource quotas • Use cost monitoring • Deploy provisioning controls • Enable abuse detection	**High**	Resource Abuse	• Test resource quotas • Verify cost monitoring • Audit resource provisioning • Review abuse detection
AWS-005	**Lambda Function Manipulation**: Unauthorized creation or modification of AWS Lambda functions through MCP for code execution attacks	• Implement function protection • Use deployment controls • Deploy modification monitoring • Enable manipulation detection	**High**	Code Injection	• Test function protection • Verify deployment controls • Audit function changes • Review manipulation detection

Threat ID	Threat Statement	Mitigations	Priority	Type of Threat	How to Verify Mitigations
AWS-006	**S3 Data Exfiltration**: Mass downloading of S3 bucket contents through overprivileged MCP server access to AWS storage services	• Implement bucket policies • Use access logging • Deploy exfiltration monitoring • Enable data loss prevention	**High**	Data Exfiltration	• Test bucket policies • Verify access logging • Audit data access • Review exfiltration detection

Bitbucket MCP Server Threats

Threat ID	Threat Statement	Mitigations	Priority	Type of Threat	How to Verify Mitigations
BB-001	**Bitbucket App Password Exposure**: Application passwords stored in plaintext configuration files accessible by unauthorized parties GitHub Playbooks	• Use secure credential storage • Implement password encryption • Deploy access monitoring • Enable exposure detection GitGuardian	**Critical**	Credential Exposure	• Scan for plaintext passwords • Test credential security • Audit password storage • Review exposure detection
BB-002	**Repository Code Exfiltration**: Mass download of private repository source code through overprivileged Bitbucket API access	• Implement repository access controls • Use download monitoring • Deploy exfiltration detection • Enable data loss prevention	**High**	Code Exfiltration	• Test access controls • Verify download monitoring • Audit code access • Review exfiltration detection
BB-003	**Malicious Commit Injection**: AI agents creating commits with malicious code or backdoors through compromised Bitbucket access	• Implement commit validation • Use code scanning • Deploy malicious code detection • Enable commit monitoring	**High**	Code Injection	• Test commit validation • Verify code scanning • Audit malicious code detection • Review commit monitoring
BB-004	**Branch Protection Bypass**: Circumventing branch protection rules and policies through API access to make unauthorized changes	• Implement API-level protection • Use policy enforcement • Deploy bypass monitoring • Enable protection validation	**High**	Authorization Bypass	• Test protection enforcement • Verify policy compliance • Audit bypass attempts • Review protection validation
BB-005	**Git History Manipulation**: Unauthorized access to commit history, author information, and repository metadata for intelligence gathering	• Implement history access controls • Use metadata filtering • Deploy access monitoring • Enable manipulation detection	**Medium**	Information Disclosure	• Test history access controls • Verify metadata filtering • Audit history access • Review manipulation detection
BB-006	**Webhook Abuse**: Exploitation of webhook configurations to trigger	• Implement webhook validation • Use signature verification •	**Medium**	Event Manipulation	• Test webhook validation • Verify signatures • Audit

Threat ID	Threat Statement	Mitigations	Priority	Type of Threat	How to Verify Mitigations
	unauthorized actions or data exfiltration	Deploy webhook monitoring • Enable abuse detection			webhook activity • Review abuse detection

Key Recommendations

Immediate Security Actions

Authentication and Authorization

- Implement mandatory OAuth 2.1 with PKCE across all MCP implementations OWASP Cheat Sheet Series +5
- Deploy fine-grained access controls with least privilege principles GitGuardian +2
- Enable multi-factor authentication for all production MCP deployments Google Cloud
- Establish centralized credential management and rotation policies

Network and Transport Security

- Enforce TLS 1.3 for all MCP communications Stack Overflow +3
- Implement certificate pinning and validation
- Deploy network segmentation and firewall controls
- Enable comprehensive traffic monitoring and analysis

Monitoring and Detection

- Establish comprehensive audit logging for all MCP operations Cisco Community
- Deploy real-time security monitoring and alerting
- Implement behavioral analytics for anomaly detection
- Enable automated incident response capabilities

Strategic Security Investments

Protocol Enhancement

- Advocate for security-by-default in MCP specification updates Windows Experience Blog +2
- Contribute to standardized authentication mechanisms
- Participate in security working groups and community initiatives
- Support development of MCP security certification programs

Enterprise Security Framework

- Develop organization-specific MCP security standards
- Implement centralized MCP server management platforms
- Establish security training programs for development teams (Teleport) (Red Hat)
- Create vendor security assessment processes for MCP implementations

The Model Context Protocol ecosystem requires immediate and sustained security attention as AI systems become more autonomous and integrated with critical business systems. (Windows Experience Blog +5) Success depends on combining traditional cybersecurity best practices with emerging GenAI-specific threat mitigation strategies. (Invicti +5)