Infosecotb.com with vMeNext Threat Model

Owner: InfoSecOTB
Reviewer: Piotr Kowalczyk
Contributors:
Date Generated: Mon Oct 06 2025

Executive Summary

High level system description

Infosecotb.com is a professional cybersecurity blog hosted on WordPress through BlueHost. The blog serves as a platform for sharing insights, articles, and resources related to information security, targeting cybersecurity professionals and enthusiasts.

Website Structure:
- Content Management System (CMS): Built on WordPress, allowing for easy content creation, management, and publishing.
- User Interaction: Features such as chatbot, comments, contact forms, and newsletter subscriptions that facilitate user engagement.
- Categorized Content: Articles are organized into categories based on topics

Functionality:
- Article Publishing: Regularly updated with new blog posts that include technical guides, best practices, and industry insights.
- Search Functionality: Allows users to search for specific topics or articles.
- Social Media Integration: Links to social media platforms for sharing and promoting content.
- vMeNext AI powered chatbot

User Types:
- Visitors: General users seeking information on cybersecurity topics.
- Administrators: Individuals with backend access for managing content, settings, and website security.

Technical Environment:
- Hosting: Utilizes BlueHost for hosting, which provides shared or dedicated server resources.
- Plugins and Themes: Employs various WordPress plugins for enhanced functionality (e.g., SEO, analytics, security).
- Database: Relies on a MySQL database for storing content, user information, and site settings.
- vMeNext chatbot published using iFrames

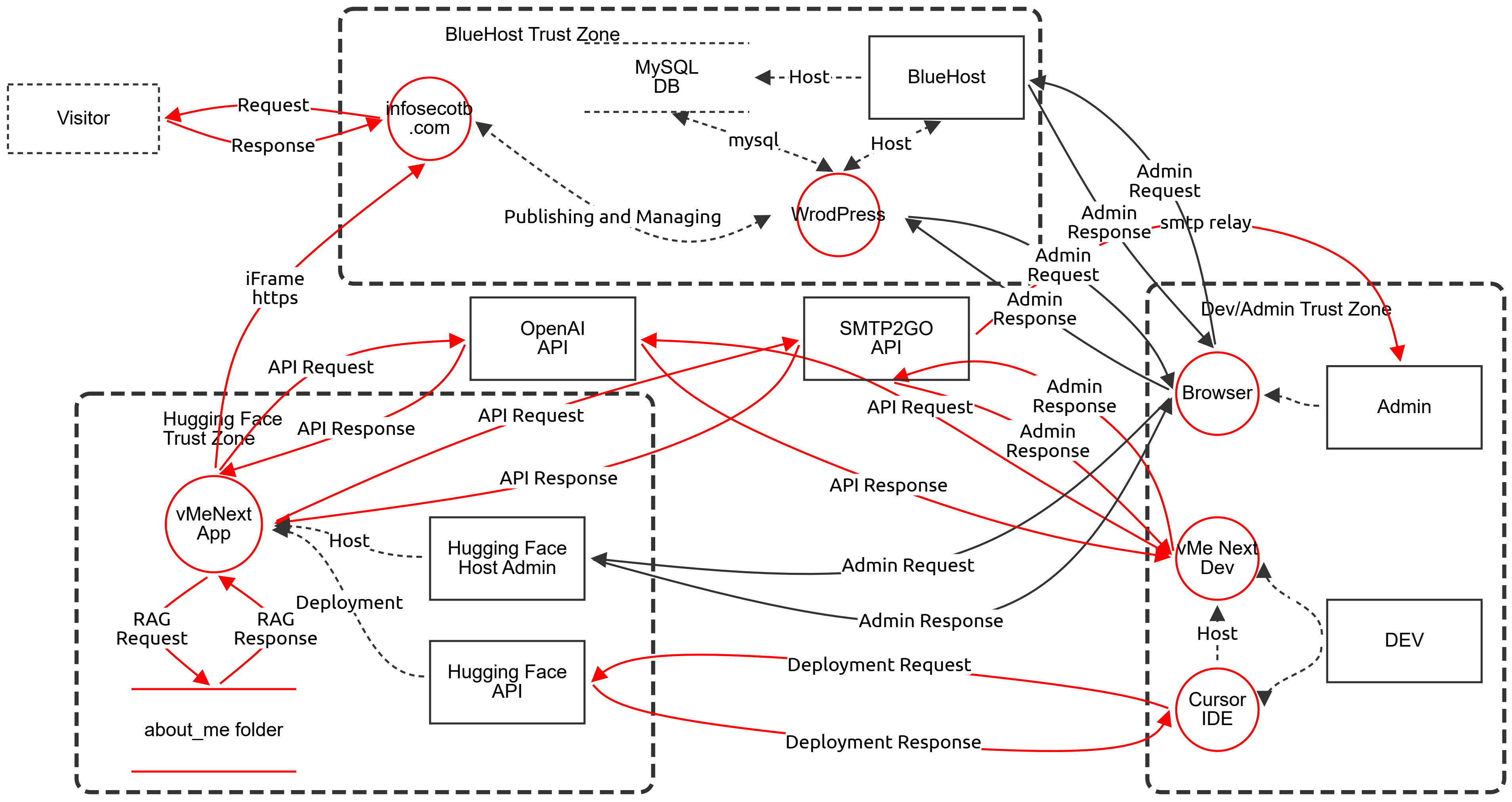
vMeNext is a comprehensive AI-powered chatbot system designed to serve as an intelligent interface for blog content and website management. Built with modern Python technologies, it combines the power of OpenAI's GPT models with automated web scraping, monitoring, and user engagement features.

Key Capabilities:
- Intelligent Conversations: Powered by OpenAI's latest GPT models for natural, context-aware responses
- Blog Content Integration: Automatic scraping, processing, and summarization of blog posts
- Website Monitoring: Continuous availability checking with real-time alerts
- Document Processing: Support for multiple file formats (PDF, DOCX, TXT, MD)
- User Engagement: Automated email notifications and contact management
- Analytics Dashboard: Website uptime statistics with visualizations

Summary

Total Threats	65
Total Mitigated	0
Total Open	65
Open / Critical Severity	0
Open / High Severity	32
Open / Medium Severity	30
Open / Low Severity	3

Infosecotb.com with vMeNext Diagram



Infosecotb.com with vMeNext Diagram

Visitor (Actor) *- Out of Scope*

Reason for out of scope:

Description: Visitor connecting to infosecotb.com using a browser

Number	Title	Type	Severity	Status	Score	Description	Mitigations

vMeNext App (Process)

Description: Gradio ChatBot Python Application with RAG Running on Hugging Face Space

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Spoofing of vMeNext App Identity via API Key Compromise	Spoofing	High	Open		The vMeNext App process communicates with OpenAI API and SMTP2GO API using API keys. If these keys are compromised (e.g., hardcoded in source, exposed in logs, or stolen from environment variables), an attacker could impersonate the application and make unauthorized API calls. The app resides in the Hugging Face Trust Zone and crosses trust boundaries via public network flows to external APIs.	- Store API keys in secure secret management systems (e.g., Hugging Face Secrets, HashiCorp Vault) - Implement key rotation policies - Use environment variables with restricted access - Monitor API usage for anomalies - Implement rate limiting and usage quotas
	Tampering with RAG Context from about_me Folder	Tampering	High	Open		The vMeNext App reads documents from the about_me folder to provide context to the AI chatbot. If an attacker gains write access to this folder (through compromised Hugging Face Space credentials or deployment pipeline), they could inject malicious content, misleading information, or prompt injection attacks that manipulate the chatbot's responses. This could lead to misinformation, reputation damage, or exploitation of users.	- Implement file integrity monitoring on the about_me folder - Use read-only file system permissions where possible - Validate and sanitize document content before processing - Implement version control and audit logging for document changes - Use code signing for deployment packages - Implement content security policies for RAG inputs
	Information Disclosure via Unencrypted iFrame Communication	Information Disclosure	Medium	Open		The vMeNext App is embedded in infosecotb.com via iFrame over HTTPS. However, if the iFrame implementation doesn't properly restrict cross-origin communication or if sensitive data is passed through postMessage without validation, attackers could intercept or exfiltrate user conversations, prompts, or API responses. The flow crosses from Hugging Face Trust Zone to BlueHost Trust Zone.	- Implement Content Security Policy (CSP) with frame-ancestors directive - Use postMessage with strict origin validation - Avoid passing sensitive data through iFrame communication - Implement Subresource Integrity (SRI) for iFrame resources - Use X-Frame-Options headers appropriately

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Denial of Service via Resource Exhaustion	Denial of Service	High	Open		The vMeNext App processes user queries and makes API calls to OpenAI, which can be resource-intensive. An attacker could flood the application with complex queries or rapid requests, exhausting compute resources, API quotas, or causing financial damage through excessive API usage. The app is publicly accessible through the infosecotb.com website.	- Implement rate limiting per user/IP address - Set maximum query length and complexity limits - Configure API usage quotas and budget alerts - Implement request queuing with timeout mechanisms - Use CAPTCHA or proof-of-work for suspicious traffic patterns - Monitor resource utilization and implement auto-scaling limits
	Elevation of Privilege via Prompt Injection	Elevation of Privilege	High	Open		The vMeNext App uses OpenAI API with RAG context from the about_me folder. An attacker could craft malicious prompts that attempt to override system instructions, access restricted information, or manipulate the AI to perform unauthorized actions. This is particularly concerning as the chatbot may have access to sensitive context about the blog owner or system configuration.	- Implement robust prompt filtering and validation - Use OpenAI's moderation API to screen inputs - Separate system prompts from user inputs with clear delimiters - Implement output filtering to prevent sensitive data leakage - Use role-based access controls in prompt design - Regularly audit and test for prompt injection vulnerabilities - Implement conversation context limits
	Repudiation of Malicious Actions via Insufficient Logging	Repudiation	Medium	Open		The vMeNext App processes user queries and makes external API calls, but if logging is insufficient or logs are not properly secured, malicious actors could deny performing harmful actions (e.g., attempting prompt injections, data exfiltration, or abuse). Without comprehensive audit trails, it becomes difficult to investigate security incidents or attribute actions to specific users.	- Implement comprehensive logging of all user interactions, API calls, and system events - Include timestamps, user identifiers (IP, session), query content, and responses - Store logs in tamper-proof, centralized logging system - Implement log retention policies compliant with security requirements - Enable real-time alerting for suspicious patterns - Ensure logs are encrypted at rest and in transit

about_me folder (Store)

Description: Folder with documents read by Python application and provided to AI ChatBot as a prompt context.

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Tampering with RAG Documents in about_me Folder	Tampering	High	Open		The about_me folder stores documents that provide context to the vMeNext App chatbot. If an attacker gains write access to this data store (through compromised deployment credentials, Hugging Face Space vulnerabilities, or insider threats), they could modify, delete, or inject malicious content. This could lead to the chatbot providing incorrect information, executing prompt injection attacks, or damaging the blog's reputation.	- Implement strict file system permissions (read-only for application) - Use file integrity monitoring (FIM) to detect unauthorized changes - Implement version control for all documents with audit trails - Require multi-factor authentication for deployment access - Use code review and approval processes for document updates - Implement automated content validation and scanning
	Information Disclosure of Sensitive Context Data	Information Disclosure	Medium	Open		The about_me folder contains documents with information about the blog owner and system. If these documents include sensitive personal information, credentials, API keys, or system configuration details, and the folder permissions are misconfigured or the Hugging Face Space is compromised, this information could be exposed to unauthorized parties. The data store resides within the Hugging Face Trust Zone but is accessed by the application process.	- Review all documents for sensitive information before deployment - Implement data classification and handling policies - Encrypt sensitive documents at rest - Use secret management systems for credentials (never store in documents) - Implement access controls and audit logging for folder access - Regularly scan for accidentally committed secrets or PII
	Denial of Service via Large or Malformed Documents	Denial of Service	Medium	Open		The about_me folder stores documents that are processed by the vMeNext App for RAG functionality. If an attacker can upload extremely large files, malformed documents, or files with embedded exploits (e.g., XML bombs, zip bombs), the application could experience resource exhaustion, crashes, or processing delays when attempting to read and parse these files.	- Implement file size limits for documents - Validate file formats and structure before processing - Use safe parsing libraries with timeout mechanisms - Implement resource limits for document processing - Scan uploaded files for malware and malicious content - Use sandboxed environments for document parsing

DEV (Actor)

Description: vMeNext Application Developer

Number	Title	Type	Severity	Status	Score	Description	Mitigations

Cursor IDE (Process)

Description: Cursor IDE used for developing and running vMe Next Dev application and deploying on Hugging Face Space

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Tampering with Source Code via Compromised IDE or Extensions	Tampering	High	Open		Cursor IDE is used for developing the vMeNext application. If the IDE itself is compromised (through malicious extensions, supply chain attacks, or malware), an attacker could inject malicious code into the application before deployment. This is particularly concerning as the IDE has direct deployment capabilities to Hugging Face Space, residing in the Dev/Admin Trust Zone.	- Use only trusted and verified IDE extensions - Implement code signing and verification in deployment pipeline - Use separate development and deployment credentials - Implement code review processes before deployment - Scan development environment for malware regularly - Use version control with protected branches - Implement CI/CD pipeline with automated security scanning
	Elevation of Privilege via Stolen Developer Credentials	Elevation of Privilege	High	Open		Cursor IDE has access to deployment credentials for Hugging Face Space and potentially other sensitive systems. If developer credentials are compromised (through phishing, keylogging, or session hijacking), an attacker could gain unauthorized access to deploy malicious code, modify the application, or access production secrets. The IDE resides in the Dev/Admin Trust Zone with privileged access.	- Enforce multi-factor authentication (MFA) for all developer accounts - Use hardware security keys for authentication - Implement credential rotation policies - Use separate credentials for development and production - Monitor and alert on unusual deployment activities - Implement IP whitelisting for deployment access - Use time-limited access tokens instead of long-lived credentials
	Information Disclosure via IDE Telemetry or Logs	Information Disclosure	Medium	Open		Cursor IDE may collect telemetry data, store logs, or cache sensitive information (API keys, credentials, source code) locally. If the developer's workstation is compromised or if telemetry data is transmitted insecurely, sensitive information could be exposed to unauthorized parties.	- Review and configure IDE telemetry settings to minimize data collection - Encrypt local IDE cache and configuration files - Use full disk encryption on developer workstations - Implement data loss prevention (DLP) tools - Regularly clear sensitive data from IDE history and caches - Use secure channels for any telemetry transmission - Implement endpoint detection and response (EDR) on developer machines

infosecotb .com (Process)

Description: InfoSec Outside The Box Cybersecurity Blog created and managed with WordPress CMS with vMeNext AI powered chatbot added using iFrame

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Spoofing via WordPress Admin Account Compromise	Spoofing	High	Open		The infosecotb.com WordPress process manages the blog content and embeds the vMeNext chatbot via iFrame. If WordPress admin credentials are compromised (through brute force, credential stuffing, or phishing), an attacker could impersonate the administrator, modify content, inject malicious scripts, or alter the iFrame to point to a malicious chatbot. The process resides in the BlueHost Trust Zone and receives public network traffic from visitors.	- Enforce strong password policies and multi-factor authentication (MFA) - Implement account lockout policies after failed login attempts - Use security plugins (e.g., Wordfence, Sucuri) for WordPress hardening - Limit login attempts and implement CAPTCHA - Monitor and alert on suspicious login activities - Use IP whitelisting for admin access where possible - Regularly audit user accounts and remove unused accounts
	Tampering with WordPress Core or Plugin Files	Tampering	High	Open		The infosecotb.com WordPress installation includes core files, themes, and plugins. If an attacker gains write access to the file system (through compromised FTP/SSH credentials, vulnerable plugins, or hosting account compromise), they could modify WordPress files to inject backdoors, malware, or malicious iFrames. This could compromise all site visitors and the embedded vMeNext chatbot.	- Implement file integrity monitoring for WordPress core, themes, and plugins - Use read-only file permissions where possible - Keep WordPress core, themes, and plugins updated - Remove unused themes and plugins - Use security plugins to scan for malware and vulnerabilities - Implement Web Application Firewall (WAF) - Use secure FTP (SFTP) with key-based authentication - Regularly backup and verify backup integrity
	Information Disclosure via WordPress Configuration Exposure	Information Disclosure	High	Open		The infosecotb.com WordPress process stores sensitive configuration in wp-config.php, including database credentials, authentication keys, and salts. If this file is exposed through misconfiguration (e.g., directory listing, backup files in web root, or path traversal vulnerabilities), attackers could gain access to the MySQL database and compromise the entire site. The process is publicly accessible over HTTPS.	- Move wp-config.php outside the web root or use .htaccess to deny access - Disable directory listing on the web server - Remove backup files and version control directories from web root - Use strong, unique authentication keys and salts - Implement proper file permissions (e.g., 600 for wp-config.php) - Use environment variables for sensitive configuration where possible - Regularly scan for exposed sensitive files

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Denial of Service via WordPress Resource Exhaustion	Denial of Service	Medium	Open		The infosecotb.com WordPress process is publicly accessible and handles visitor requests. An attacker could launch DDoS attacks, XML-RPC amplification attacks, or exploit resource-intensive plugins to exhaust server resources, making the site unavailable. This would also impact the embedded vMeNext chatbot accessibility.	- Implement rate limiting and request throttling - Disable XML-RPC if not needed or restrict access - Use CDN with DDoS protection (e.g., Cloudflare) - Implement caching mechanisms (e.g., WP Super Cache, W3 Total Cache) - Configure resource limits on the hosting environment - Monitor server resources and implement auto-scaling if possible - Use Web Application Firewall (WAF) to filter malicious traffic
	Elevation of Privilege via WordPress Plugin Vulnerabilities	Elevation of Privilege	High	Open		The infosecotb.com WordPress installation uses various plugins for enhanced functionality. Vulnerable plugins could allow attackers to escalate privileges from unauthenticated visitor to administrator, execute arbitrary code, or gain unauthorized access to the hosting environment. The process resides in the BlueHost Trust Zone with access to the MySQL database.	- Keep all plugins updated to latest versions - Only install plugins from trusted sources with good security track records - Regularly audit installed plugins and remove unused ones - Use security plugins to scan for known vulnerabilities - Implement principle of least privilege for WordPress user roles - Monitor WordPress security advisories and apply patches promptly - Use virtual patching through WAF for zero-day vulnerabilities
	Repudiation via Insufficient WordPress Audit Logging	Repudiation	Medium	Open		The infosecotb.com WordPress process handles content management, user interactions, and administrative actions. If audit logging is insufficient or logs are not properly secured, malicious actors could deny performing harmful actions (e.g., content modification, user account manipulation, or malicious plugin installation). Without comprehensive logs, incident investigation and attribution become difficult.	- Install and configure WordPress activity logging plugins (e.g., WP Activity Log) - Log all administrative actions, login attempts, and content changes - Store logs in centralized, tamper-proof logging system - Implement log retention policies - Enable real-time alerting for critical actions - Ensure logs include timestamps, user identifiers, IP addresses, and action details - Regularly review logs for suspicious activities

iFrame https (Data Flow)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Tampering with iFrame Content via Man-in-the-Middle Attack	Tampering	Medium	Open		The iFrame flow connects vMeNext App to infosecotb.com over HTTPS, crossing from Hugging Face Trust Zone to BlueHost Trust Zone. While HTTPS provides encryption, if certificate validation is weak or if there are TLS downgrade vulnerabilities, an attacker could perform a man-in-the-middle attack to modify the iFrame content, inject malicious scripts, or redirect users to phishing sites.	- Enforce TLS 1.2 or higher with strong cipher suites - Implement HTTP Strict Transport Security (HSTS) with preloading - Use certificate pinning where feasible - Implement Subresource Integrity (SRI) for iFrame resources - Monitor for TLS certificate changes and anomalies - Use Content Security Policy (CSP) to restrict iFrame sources
	Information Disclosure via iFrame Clickjacking	Information Disclosure	Medium	Open		The iFrame flow embeds the vMeNext App into infosecotb.com. If proper frame protection headers are not implemented, an attacker could embed the chatbot in a malicious site and use clickjacking techniques to trick users into revealing sensitive information or performing unintended actions. The flow crosses trust boundaries between Hugging Face and BlueHost zones.	- Implement X-Frame-Options: SAMEORIGIN or DENY headers - Use Content Security Policy (CSP) frame-ancestors directive - Implement frame-busting JavaScript as defense-in-depth - Validate and restrict iFrame embedding to trusted domains only - Educate users about clickjacking risks - Monitor for unauthorized iFrame embedding attempts
	Denial of Service via iFrame Resource Loading	Denial of Service	Low	Open		The iFrame flow loads the vMeNext App into infosecotb.com. If the Hugging Face Space becomes unavailable or experiences performance issues, the iFrame will fail to load or cause delays, degrading the user experience on the main website. This creates a dependency where the availability of infosecotb.com is partially dependent on Hugging Face infrastructure.	- Implement lazy loading for iFrame to prevent blocking page load - Set appropriate timeout values for iFrame loading - Provide fallback content or error messages when iFrame fails to load - Monitor Hugging Face Space availability and performance - Consider implementing a backup chatbot solution - Use asynchronous loading techniques - Implement graceful degradation strategies

(Data Flow) *- Out of Scope*

Reason for out of scope:

Number	Title	Type	Severity	Status	Score	Description	Mitigations

(Data Flow) *- Out of Scope*

Reason for out of scope:

Number	Title	Type	Severity	Status	Score	Description	Mitigations

(Data Flow) *- Out of Scope*

Reason for out of scope:

Number	Title	Type	Severity	Status	Score	Description	Mitigations

Host (Data Flow) *- Out of Scope*

Reason for out of scope:

Number	Title	Type	Severity	Status	Score	Description	Mitigations

RAG Request (Data Flow)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Information Disclosure via Unencrypted RAG Request	Information Disclosure	Medium	Open		The RAG Request flow from vMeNext App to about_me folder occurs within the Hugging Face Trust Zone. However, if the internal communication is not properly secured or if the Hugging Face Space environment is compromised, an attacker could intercept these requests to understand the application's context retrieval patterns, potentially revealing sensitive information about the RAG implementation or document structure.	- Implement encryption for internal data flows even within trust zones - Use secure file access APIs with proper authentication - Implement access controls and audit logging for folder access - Monitor for unusual file access patterns - Use principle of least privilege for application file system access - Implement runtime application self-protection (RASP)
	Tampering with RAG Request to Access Unauthorized Files	Tampering	High	Open		The RAG Request flow allows vMeNext App to read documents from the about_me folder. If input validation is insufficient, an attacker could manipulate the request to perform path traversal attacks, accessing files outside the intended directory. This could expose sensitive configuration files, credentials, or other application data within the Hugging Face Space environment.	- Implement strict input validation and sanitization for file paths - Use whitelisting for allowed file names and paths - Implement path canonicalization to prevent traversal attacks - Use chroot or containerization to restrict file system access - Implement file access audit logging - Use secure file access APIs that prevent path traversal - Regularly test for path traversal vulnerabilities

RAG Response (Data Flow)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Information Disclosure via RAG Response Interception	Information Disclosure	Medium	Open		The RAG Response flow returns document content from about_me folder to vMeNext App within the Hugging Face Trust Zone. If the Hugging Face Space environment is compromised or if logging is excessive, an attacker could intercept these responses to extract sensitive information from the documents, understand the RAG context, or identify vulnerabilities in the content processing logic.	- Implement encryption for internal data flows - Minimize logging of RAG response content - Use secure memory handling to prevent data leakage - Implement data loss prevention (DLP) controls - Monitor for unusual data access patterns - Use runtime application self-protection (RASP) - Implement secure coding practices to prevent memory dumps
	Tampering with RAG Response to Inject Malicious Content	Tampering	High	Open		The RAG Response flow delivers document content to vMeNext App for processing. If an attacker compromises the about_me folder or the file reading mechanism, they could inject malicious content, prompt injection payloads, or misleading information into the response. This could manipulate the chatbot's behavior, spread misinformation, or exploit downstream processing vulnerabilities.	- Implement file integrity monitoring for source documents - Validate and sanitize RAG response content before processing - Use content security policies for RAG data - Implement anomaly detection for unusual response patterns - Use read-only file system permissions - Implement digital signatures for documents - Regularly audit document content for malicious patterns

mysql (Data Flow) *- Out of Scope*

Reason for out of scope: Managed by BlueHost

Description: Managed and secured by BlueHost

Number	Title	Type	Severity	Status	Score	Description	Mitigations

smtp relay (Data Flow)

Description: E-mail sent to administrator

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Spoofing via SMTP Relay Impersonation	Spoofing	Medium	Open		The smtp relay flow sends emails from vMe Next Dev to Admin over public network with encryption. However, if SMTP authentication is weak or if SPF/DKIM/DMARC records are not properly configured, an attacker could spoof emails appearing to come from the vMeNext system, potentially phishing the administrator or sending malicious notifications. The flow crosses from Dev/Admin Trust Zone to external email infrastructure.	- Implement SPF, DKIM, and DMARC records for email authentication - Use strong SMTP authentication credentials - Implement TLS for SMTP connections (STARTTLS or SMTPS) - Monitor for unauthorized email sending attempts - Use dedicated email sending service with authentication (SMTP2GO) - Implement email rate limiting - Educate administrators about email spoofing risks
	Information Disclosure via Email Interception	Information Disclosure	Medium	Open		The smtp relay flow transmits emails over public network, though marked as encrypted. If TLS is not properly enforced or if there are downgrade attacks, email content could be intercepted in transit. Emails may contain sensitive information about system status, errors, or user activities that could aid attackers in reconnaissance.	- Enforce TLS 1.2+ for all SMTP connections - Implement certificate validation for SMTP servers - Avoid including sensitive information in email bodies - Use secure email gateways with encryption - Monitor for TLS downgrade attempts - Implement email encryption (S/MIME or PGP) for sensitive content - Use secure channels for critical notifications
	Denial of Service via Email Flooding	Denial of Service	Low	Open		The smtp relay flow sends notifications to the administrator. If the application generates excessive emails due to errors, monitoring alerts, or malicious triggering, it could flood the administrator's inbox, causing important notifications to be missed or overwhelming the email service. This could also incur costs if using a paid email relay service.	- Implement email rate limiting and throttling - Use email aggregation for multiple similar events - Configure alert thresholds to prevent notification storms - Implement email quotas and budget alerts - Use alternative notification channels for critical alerts (SMS, push) - Monitor email sending patterns for anomalies - Implement circuit breakers for email sending

API Response (Data Flow)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Information Disclosure via API Response Interception	Information Disclosure	Medium	Open		The API Response flow from SMTP2GO API to vMeNext App crosses trust boundaries over public network with HTTPS encryption. If TLS is compromised or if the application logs API responses excessively, sensitive information about email delivery status, API keys, or system configuration could be exposed. The flow enters the Hugging Face Trust Zone from external infrastructure.	- Enforce TLS 1.2+ with strong cipher suites - Implement certificate pinning for API connections - Minimize logging of API response content - Sanitize logs to remove sensitive data - Use secure memory handling for API responses - Implement data loss prevention (DLP) controls - Monitor for unusual API response patterns
	Tampering with API Response to Cause False Status	Tampering	Medium	Open		The API Response flow delivers email delivery status from SMTP2GO API to vMeNext App. If an attacker performs a man-in-the-middle attack or compromises the API endpoint, they could modify responses to indicate successful delivery when emails failed, or vice versa. This could mask security incidents or cause operational confusion.	- Implement response signature verification - Use mutual TLS (mTLS) for API authentication - Validate API response structure and content - Implement anomaly detection for unexpected responses - Use API response caching with integrity checks - Monitor for API response tampering indicators - Implement end-to-end encryption for critical data

API Request (Data Flow)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Spoofing via API Key Theft in Transit	Spoofing	High	Open		The API Request flow from vMeNext App to SMTP2GO API crosses trust boundaries over public network with HTTPS. The request includes API authentication credentials. If TLS is compromised or if the application inadvertently exposes the API key (in logs, error messages, or client-side code), an attacker could steal the key and impersonate the application to send unauthorized emails.	- Store API keys in secure secret management systems - Never log or expose API keys in error messages - Implement API key rotation policies - Use environment variables with restricted access - Monitor API usage for anomalies and unauthorized access - Implement IP whitelisting for API access - Use short-lived tokens instead of long-lived API keys where possible

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Tampering with API Request to Send Malicious Emails	Tampering	High	Open		The API Request flow sends email instructions from vMeNext App to SMTP2GO API. If the application doesn't properly validate and sanitize email content before sending, an attacker could exploit input validation vulnerabilities to inject malicious content, phishing links, or spam into emails sent to administrators. This crosses from Hugging Face Trust Zone to external email infrastructure.	- Implement strict input validation for all email fields (to, from, subject, body) - Sanitize email content to prevent injection attacks - Use email templates with parameterized content - Implement content security policies for emails - Monitor sent emails for suspicious patterns - Implement rate limiting and approval workflows for sensitive emails - Use email security scanning before sending
	Information Disclosure via API Request Logging	Information Disclosure	Medium	Open		The API Request flow contains email content and potentially sensitive information about system events or user activities. If the application or network infrastructure logs these requests excessively, sensitive information could be exposed through log files, monitoring systems, or compromised logging infrastructure.	- Minimize logging of API request content - Sanitize logs to remove sensitive data (PII, credentials) - Encrypt logs at rest and in transit - Implement access controls for log files - Use secure logging infrastructure with audit trails - Implement log retention policies - Regularly review and audit logging practices

Admin Response (Data Flow)

Description: SMTP2GO Administration

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Information Disclosure via Admin Response Interception	Information Disclosure	Medium	Open		The Admin Response flow from vMe Next Dev to SMTP2GO API crosses trust boundaries over public network with HTTPS encryption. This flow contains SMTP2GO administration data that could reveal API configuration, usage statistics, or account details. If TLS is compromised or if the response is logged excessively, sensitive administrative information could be exposed.	- Enforce TLS 1.2+ with strong cipher suites - Implement certificate pinning for administrative connections - Minimize logging of administrative response data - Use secure session management for admin interfaces - Implement multi-factor authentication for admin access - Monitor for unusual administrative activities - Use VPN or private networks for administrative access where possible

Admin Response (Data Flow)

Description: SMTP2GO Administration

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Tampering with Admin Response to Manipulate Configuration	Tampering	High	Open		The Admin Response flow from SMTP2GO API to vMe Next Dev delivers administrative configuration and status information over public network. If an attacker performs a man-in-the-middle attack, they could modify the response to provide false configuration data, hide security issues, or manipulate administrative settings. This could lead to misconfiguration or security vulnerabilities.	- Implement response signature verification - Use mutual TLS (mTLS) for administrative connections - Validate response integrity using checksums or digital signatures - Implement anomaly detection for unexpected administrative responses - Use out-of-band verification for critical configuration changes - Monitor for administrative response tampering indicators - Implement configuration version control and audit trails

API Request (Data Flow)

Description: OpenAI API Request

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Spoofing via Stolen OpenAI API Key	Spoofing	High	Open		The API Request flow from vMe Next Dev to OpenAI API crosses trust boundaries over public network with HTTPS. The request includes OpenAI API authentication credentials. If the API key is compromised through code exposure, logging, or developer workstation compromise, an attacker could impersonate the application to make unauthorized API calls, incurring costs or accessing sensitive AI capabilities.	- Store API keys in secure secret management systems (e.g., environment variables, vault) - Never commit API keys to version control - Implement API key rotation policies - Monitor API usage for anomalies and unauthorized access - Implement usage quotas and budget alerts - Use IP whitelisting for API access where possible - Implement least privilege access for API keys
	Tampering with API Request for Prompt Injection	Tampering	High	Open		The API Request flow sends prompts from vMe Next Dev to OpenAI API during development and testing. If the development environment is compromised or if input validation is insufficient, an attacker could inject malicious prompts to test vulnerabilities, extract training data, or manipulate the AI model's behavior. This could reveal security weaknesses before they're fixed in production.	- Implement strict input validation and sanitization for all prompts - Use separate API keys for development and production - Implement prompt filtering and content moderation - Monitor API requests for suspicious patterns - Use OpenAI's moderation API to screen inputs - Implement rate limiting for development API calls - Regularly audit and review API usage logs

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Information Disclosure via Development API Requests	Information Disclosure	Medium	Open		The API Request flow from vMe Next Dev to OpenAI API may contain sensitive information during development, including test data, system prompts, or configuration details. If these requests are logged excessively or if the development environment is compromised, sensitive information about the application's AI implementation could be exposed.	- Minimize logging of API request content in development - Use synthetic or anonymized data for testing - Sanitize logs to remove sensitive information - Implement access controls for development environments - Use separate development and production API keys - Encrypt development environment data at rest - Regularly audit development practices for security

API Request (Data Flow)

Description: OpenAI API Request

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Spoofing via API Key Compromise in Production	Spoofing	High	Open		The API Request flow from vMeNext App to OpenAI API crosses trust boundaries over public network with HTTPS encryption. The request includes OpenAI API authentication credentials. If the API key is exposed through application vulnerabilities, Hugging Face Space compromise, or logging, an attacker could impersonate the application to make unauthorized API calls, causing financial damage and potential data exposure.	- Store API keys in Hugging Face Secrets or secure secret management - Implement API key rotation policies - Monitor API usage for anomalies and set budget alerts - Use IP whitelisting for API access - Implement rate limiting and usage quotas - Never log or expose API keys in application code - Use environment variables with restricted access
	Tampering with API Request for Malicious Prompts	Tampering	High	Open		The API Request flow sends user prompts from vMeNext App to OpenAI API. If input validation is insufficient, an attacker could inject malicious prompts to manipulate the AI's responses, extract sensitive information from the model, perform prompt injection attacks, or cause the application to generate harmful content. The flow crosses from Hugging Face Trust Zone to external AI infrastructure.	- Implement robust input validation and sanitization - Use OpenAI's moderation API to screen all inputs - Implement prompt filtering with blacklists and whitelists - Separate system prompts from user inputs with clear delimiters - Implement output filtering to prevent sensitive data leakage - Monitor for prompt injection patterns - Implement rate limiting per user/session - Regularly test for prompt injection vulnerabilities

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Information Disclosure via API Request Logging	Information Disclosure	Medium	Open		The API Request flow contains user prompts and potentially sensitive information. If the application logs these requests excessively or if Hugging Face Space logging is misconfigured, user conversations, personal information, or system prompts could be exposed through log files or monitoring systems.	- Minimize logging of user prompts and API request content - Sanitize logs to remove PII and sensitive data - Encrypt logs at rest and in transit - Implement access controls for log files - Use secure logging infrastructure with audit trails - Implement log retention policies compliant with privacy regulations - Regularly audit logging practices
	Denial of Service via API Rate Limit Exhaustion	Denial of Service	High	Open		The API Request flow to OpenAI API is subject to rate limits and usage quotas. An attacker could flood the application with requests to exhaust API quotas, causing legitimate users to be denied service. This could also result in significant financial costs due to excessive API usage.	- Implement application-level rate limiting per user/IP - Set API usage quotas and budget alerts - Implement request queuing with timeout mechanisms - Use CAPTCHA or proof-of-work for suspicious traffic - Monitor API usage patterns and implement anomaly detection - Implement circuit breakers to prevent quota exhaustion - Use caching to reduce API calls where appropriate

API Response (Data Flow)

Description: OpenAI API Response

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Information Disclosure via API Response Interception	Information Disclosure	Medium	Open		The API Response flow from OpenAI API to vMeNext App delivers AI-generated responses over public network with HTTPS encryption. If TLS is compromised or if the application logs responses excessively, sensitive information from AI responses, user conversations, or system prompts could be exposed. The flow enters the Hugging Face Trust Zone from external AI infrastructure.	- Enforce TLS 1.2+ with strong cipher suites - Implement certificate pinning for API connections - Minimize logging of API response content - Sanitize logs to remove sensitive data - Use secure memory handling for API responses - Implement data loss prevention (DLP) controls - Encrypt sensitive data in responses before storage

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Tampering with API Response to Inject Malicious Content	Tampering	High	Open		The API Response flow delivers AI-generated content from OpenAI API to vMeNext App. If an attacker performs a man-in-the-middle attack or compromises the API endpoint, they could modify responses to inject malicious content, phishing links, misinformation, or XSS payloads that could be rendered in the user's browser through the chatbot interface.	- Implement response signature verification - Use mutual TLS (mTLS) for API authentication - Validate and sanitize all API response content before rendering - Implement Content Security Policy (CSP) for chatbot interface - Use output encoding to prevent XSS - Implement anomaly detection for unexpected response patterns - Monitor for API response tampering indicators

Deployment (Data Flow) *- Out of Scope*

Reason for out of scope:

Number	Title	Type	Severity	Status	Score	Description	Mitigations

Host (Data Flow) *- Out of Scope*

Reason for out of scope:

Number	Title	Type	Severity	Status	Score	Description	Mitigations

Response (Data Flow)

Description: Response from infosecotb.com website including vMeNext chatbot

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Information Disclosure via Unencrypted Response Content	Information Disclosure	Low	Open		The Response flow from infosecotb.com to Visitor delivers website content including the embedded vMeNext chatbot over HTTPS on public network. While HTTPS provides encryption, if TLS configuration is weak or if sensitive information is inadvertently included in responses (e.g., debug information, internal paths, API keys in client-side code), it could be exposed to visitors or attackers.	- Enforce TLS 1.2+ with strong cipher suites - Implement HTTP Strict Transport Security (HSTS) - Remove debug information and verbose error messages from production - Sanitize responses to prevent information leakage - Use security headers (X-Content-Type-Options, X-Frame-Options) - Regularly scan for exposed sensitive information - Implement Content Security Policy (CSP)

Publishing and Managing (Data Flow) *- Out of Scope*

Reason for out of scope: Managed and secured by BlueHost

Number	Title	Type	Severity	Status	Score	Description	Mitigations

Request (Data Flow)

Description: Request to infosecotb.com website including vMeNext chatbot

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Tampering with Request to Exploit WordPress Vulnerabilities	Tampering	High	Open		The Request flow from Visitor to infosecotb.com crosses trust boundaries over public network with HTTPS. Visitors can send malicious requests to exploit WordPress vulnerabilities, including SQL injection, XSS, CSRF, or plugin vulnerabilities. The flow enters the BlueHost Trust Zone from untrusted public internet.	- Keep WordPress core, themes, and plugins updated - Implement Web Application Firewall (WAF) with WordPress-specific rules - Use security plugins (e.g., Wordfence, Sucuri) - Implement input validation and sanitization - Use parameterized queries to prevent SQL injection - Implement CSRF tokens for all forms - Use Content Security Policy (CSP) to prevent XSS - Regularly scan for vulnerabilities
	Denial of Service via Request Flooding	Denial of Service	Medium	Open		The Request flow from Visitor to infosecotb.com is publicly accessible. An attacker could launch DDoS attacks, send resource-intensive requests, or exploit WordPress vulnerabilities to exhaust server resources, making the website unavailable to legitimate users. This also impacts the embedded vMeNext chatbot.	- Implement rate limiting and request throttling - Use CDN with DDoS protection (e.g., Cloudflare) - Implement Web Application Firewall (WAF) - Configure resource limits on hosting environment - Implement caching mechanisms - Monitor traffic patterns and implement anomaly detection - Use CAPTCHA for suspicious traffic - Implement IP-based blocking for malicious sources
	Spoofing via Session Hijacking	Spoofing	Medium	Open		The Request flow from Visitor to infosecotb.com includes session cookies for authenticated users. If session management is weak or if cookies are not properly secured, an attacker could hijack user sessions through XSS, network sniffing, or session fixation attacks to impersonate legitimate users or administrators.	- Use secure and HttpOnly flags for all cookies - Implement SameSite cookie attribute - Use strong session ID generation - Implement session timeout and renewal - Regenerate session IDs after authentication - Implement IP-based session validation - Use HTTPS for all pages (not just login) - Implement logout functionality that invalidates sessions

Host (Data Flow) *- Out of Scope*

Reason for out of scope:

Description: Managed by BlueHost

Number	Title	Type	Severity	Status	Score	Description	Mitigations

Host (Data Flow) *- Out of Scope*

Reason for out of scope: Managed and secured by BlueHost

Number	Title	Type	Severity	Status	Score	Description	Mitigations

Admin Response (Data Flow)

Description: WordPress Administration

Number	Title	Type	Severity	Status	Score	Description	Mitigations

Admin Request (Data Flow)

Description: WordPress Administration

Number	Title	Type	Severity	Status	Score	Description	Mitigations

Admin Request (Data Flow)

Description: BlueHost Administration

Number	Title	Type	Severity	Status	Score	Description	Mitigations

Admin Response (Data Flow)

Description: BlueHost Administration

Number	Title	Type	Severity	Status	Score	Description	Mitigations

Admin Response (Data Flow)

Description: Hugging Face Administration

Number	Title	Type	Severity	Status	Score	Description	Mitigations

Admin Request (Data Flow)

Description: Hugging Face Administration

Number	Title	Type	Severity	Status	Score	Description	Mitigations

Deployment Request (Data Flow)

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Spoofing via Compromised Deployment Credentials	Spoofing	High	Open		The Deployment Request flow from Cursor IDE to Hugging Face API crosses trust boundaries over public network with HTTPS encryption. The request includes deployment authentication credentials. If these credentials are compromised through IDE vulnerabilities, developer workstation attacks, or credential theft, an attacker could impersonate the developer to deploy malicious code to production.	- Use short-lived deployment tokens instead of long-lived credentials - Implement multi-factor authentication for deployment - Use hardware security keys for authentication - Store credentials in secure credential managers - Implement IP whitelisting for deployment access - Monitor deployment activities for anomalies - Use separate credentials per developer with audit trails - Implement credential rotation policies
	Tampering with Deployment Request to Deploy Malicious Code	Tampering	High	Open		The Deployment Request flow sends application code from Cursor IDE to Hugging Face API. If the developer's workstation is compromised or if the IDE has malicious extensions, an attacker could modify the deployment payload to inject backdoors, malware, or vulnerable code before it reaches production. The flow crosses from Dev/Admin Trust Zone to Hugging Face Trust Zone.	- Implement code signing and verification - Use automated security scanning in deployment pipeline - Implement deployment approval workflows - Use version control with protected branches - Scan developer workstations for malware regularly - Use only trusted IDE extensions - Implement integrity checks for deployment packages - Use immutable infrastructure and infrastructure-as-code
	Information Disclosure via Deployment Request Logging	Information Disclosure	Medium	Open		The Deployment Request flow may contain sensitive information including source code, configuration files, API keys, or secrets. If the deployment process logs this information excessively or if logs are not properly secured, sensitive data could be exposed through log files, monitoring systems, or compromised infrastructure.	- Minimize logging of deployment payload content - Use secret management systems instead of embedding secrets in code - Sanitize logs to remove sensitive data - Encrypt logs at rest and in transit - Implement access controls for deployment logs - Use .gitignore and secret scanning to prevent credential commits - Regularly audit deployment logs for exposed secrets

Deployment Response (Data Flow)

Description: Hugging Face Space Application Deployment

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Information Disclosure via Deployment Response Logging	Information Disclosure	Medium	Open		The Deployment Response flow from Hugging Face API to Cursor IDE delivers deployment status and potentially sensitive information about the production environment over public network with HTTPS. If this response is logged excessively or if the developer workstation is compromised, information about the production infrastructure, configuration, or vulnerabilities could be exposed.	- Minimize logging of deployment response details - Sanitize logs to remove sensitive infrastructure information - Encrypt logs on developer workstations - Implement access controls for deployment logs - Use secure workstation configurations with full disk encryption - Monitor for unusual deployment activities - Implement data loss prevention (DLP) on developer machines
	Tampering with Deployment Response to Hide Failures	Tampering	Medium	Open		The Deployment Response flow provides feedback about deployment success or failure. If an attacker performs a man-in-the-middle attack or compromises the communication channel, they could modify responses to indicate successful deployment when it actually failed, or vice versa. This could mask security issues or cause operational confusion.	- Implement response signature verification - Use mutual TLS (mTLS) for deployment API connections - Validate deployment status through independent verification - Implement anomaly detection for unexpected responses - Monitor production environment independently of deployment responses - Use out-of-band verification for critical deployments - Implement deployment health checks and rollback mechanisms

API Response (Data Flow)

Description: OpenAI API Response

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Information Disclosure via API Response Interception in Development	Information Disclosure	Medium	Open		The API Response flow from OpenAI API to vMe Next Dev delivers AI-generated responses during development over public network with HTTPS. If the developer workstation is compromised or if responses are logged excessively, sensitive information about the AI implementation, test prompts, or system behavior could be exposed.	- Use secure workstation configurations with endpoint protection - Minimize logging of API responses in development - Use synthetic or anonymized data for testing - Encrypt development environment data at rest - Implement access controls for development logs - Use separate API keys for development and production - Regularly audit development practices for security

MySQL DB (Store) *- Out of Scope*

Reason for out of scope: Managed by BlueHost

Description: MySQL Database used for WordPress website

Number	Title	Type	Severity	Status	Score	Description	Mitigations

Admin (Actor)

Description: System Administrator

Number	Title	Type	Severity	Status	Score	Description	Mitigations

vMe Next Dev (Process)

Description: Gradio ChatBot Python Application Development

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Tampering with Development Code to Inject Vulnerabilities	Tampering	High	Open		The vMe Next Dev process is used for developing the vMeNext application in the Dev/Admin Trust Zone. If the development environment is compromised through malware, malicious IDE extensions, or supply chain attacks, an attacker could inject vulnerabilities, backdoors, or malicious code that would be deployed to production. The process has access to deployment credentials and production APIs.	- Use secure development workstations with endpoint protection - Implement code review processes before deployment - Use static and dynamic code analysis tools - Scan for vulnerabilities in dependencies - Use only trusted IDE extensions and tools - Implement version control with protected branches - Use separate development and production environments - Implement CI/CD pipeline with automated security testing
	Elevation of Privilege via Stolen Development Credentials	Elevation of Privilege	High	Open		The vMe Next Dev process has access to sensitive credentials including OpenAI API keys, SMTP2GO API keys, and Hugging Face deployment credentials. If the development environment is compromised through phishing, malware, or insider threats, an attacker could steal these credentials to gain unauthorized access to production systems, APIs, or deploy malicious code.	- Use secure credential management systems (e.g., password managers, vaults) - Implement multi-factor authentication for all accounts - Use hardware security keys for authentication - Implement principle of least privilege for credentials - Use separate credentials for development and production - Implement credential rotation policies - Monitor for unusual credential usage - Use endpoint detection and response (EDR) on development machines

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Information Disclosure via Development Environment Exposure	Information Disclosure	Medium	Open		The vMe Next Dev process handles sensitive information including source code, API keys, test data, and system architecture. If the development workstation is compromised, lost, or stolen, or if data is inadvertently exposed through cloud synchronization or backup services, sensitive information could be disclosed to unauthorized parties.	- Use full disk encryption on development workstations - Implement data loss prevention (DLP) tools - Use secure cloud storage with encryption - Implement access controls for development resources - Regularly audit data handling practices - Use secure backup solutions with encryption - Implement remote wipe capabilities for lost devices - Avoid storing sensitive data in cloud-synced folders

Browser (Process)

Description: Browser used by System Administrator

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Spoofing via Browser Session Hijacking	Spoofing	High	Open		The Browser process is used by the Admin to access WordPress, BlueHost, and Hugging Face administration interfaces in the Dev/Admin Trust Zone. If browser sessions are hijacked through XSS, malicious extensions, or network attacks, an attacker could impersonate the administrator to gain unauthorized access to all managed systems. The browser handles multiple administrative sessions simultaneously.	- Use browser with strong security features and keep it updated - Implement multi-factor authentication for all admin interfaces - Use separate browser profiles for administrative tasks - Disable or carefully vet browser extensions - Clear cookies and sessions after administrative tasks - Use VPN or secure networks for administrative access - Implement session timeout policies - Monitor for unusual administrative activities
	Tampering with Browser to Inject Malicious Scripts	Tampering	High	Open		The Browser process renders administrative interfaces and handles sensitive operations. If the browser is compromised through malicious extensions, man-in-the-browser attacks, or malware, an attacker could inject malicious scripts to modify administrative actions, steal credentials, or manipulate system configurations across WordPress, BlueHost, and Hugging Face platforms.	- Use browser with strong security features and sandbox - Keep browser and extensions updated - Use only trusted extensions from official stores - Implement Content Security Policy (CSP) on admin interfaces - Use browser isolation or virtual machines for high-risk tasks - Implement endpoint protection on administrator workstations - Regularly scan for malware and malicious extensions - Use separate browsers for administrative and general use

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Information Disclosure via Browser Cache and History	Information Disclosure	Medium	Open		The Browser process caches administrative pages, stores credentials in password managers, and maintains browsing history. If the administrator workstation is compromised or if browser data is not properly secured, sensitive information including credentials, session tokens, and administrative data could be exposed to unauthorized parties.	- Use private/incognito mode for administrative tasks - Clear browser cache and history regularly - Use secure password managers with encryption - Implement full disk encryption on administrator workstations - Use browser with strong privacy features - Disable browser sync for administrative profiles - Implement data loss prevention (DLP) tools - Use remote wipe capabilities for lost devices

OpenAI API (Actor)

Description: Artificial Intelligence API secured with a key

Number	Title	Type	Severity	Status	Score	Description	Mitigations

SMTP2GO API (Actor)

Description: E-mail relay hosted system API secured with key

Number	Title	Type	Severity	Status	Score	Description	Mitigations

Hugging Face Host Admin (Actor)

Description: Hugging Face Hosting Administrator Control Panel

Number	Title	Type	Severity	Status	Score	Description	Mitigations

Hugging Face API (Actor)

Description: Hugging Face Deployment API

Number	Title	Type	Severity	Status	Score	Description	Mitigations

BlueHost (Actor)

Description: Administrator access to BlueHost

Number	Title	Type	Severity	Status	Score	Description	Mitigations

WrodPress (Process)

Description: WordPress Content Management System

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Spoofing via WordPress Authentication Bypass	Spoofing	High	Open		The WordPress process manages authentication for the infosecotb.com website in the BlueHost Trust Zone. If authentication mechanisms are weak or vulnerable (e.g., weak passwords, lack of MFA, vulnerable plugins), an attacker could bypass authentication to gain unauthorized administrative access. The process is accessible from public network and handles admin requests from the Browser.	- Enforce strong password policies with complexity requirements - Implement multi-factor authentication (MFA) for all admin accounts - Use security plugins to monitor and block brute force attacks - Implement account lockout policies - Use CAPTCHA for login forms - Limit login attempts and implement progressive delays - Monitor for suspicious login activities - Use IP whitelisting for admin access where possible
	Tampering with WordPress Database via SQL Injection	Tampering	High	Open		The WordPress process interacts with MySQL database for content management. If input validation is insufficient or if vulnerable plugins are used, an attacker could exploit SQL injection vulnerabilities to modify database content, steal data, or gain unauthorized access. The process handles public requests and has direct database access.	- Use parameterized queries and prepared statements - Keep WordPress core and plugins updated - Implement input validation and sanitization - Use Web Application Firewall (WAF) with SQL injection rules - Implement database user with minimal privileges - Use security plugins to scan for vulnerabilities - Regularly audit database queries in custom code - Implement database activity monitoring
	Information Disclosure via WordPress Information Leakage	Information Disclosure	Medium	Open		The WordPress process may expose sensitive information through error messages, debug logs, version information, or directory listings. If not properly configured, attackers could gather reconnaissance information about the system, installed plugins, file structure, or vulnerabilities to plan targeted attacks. The process is publicly accessible.	- Disable debug mode in production - Remove version information from headers and meta tags - Disable directory listing on web server - Implement custom error pages without detailed information - Remove readme.html and other default files - Use security headers to prevent information leakage - Regularly scan for exposed sensitive files - Implement proper file permissions

Number	Title	Type	Severity	Status	Score	Description	Mitigations
	Elevation of Privilege via WordPress Plugin Vulnerabilities	Elevation of Privilege	High	Open		The WordPress process uses various plugins for functionality. Vulnerable plugins could allow attackers to escalate privileges from unauthenticated user to administrator, execute arbitrary code, or gain unauthorized access to the hosting environment. The process has access to the MySQL database and file system in the BlueHost Trust Zone.	- Keep all plugins updated to latest versions - Only install plugins from trusted sources - Regularly audit installed plugins and remove unused ones - Use security plugins to scan for vulnerabilities - Implement principle of least privilege for WordPress user roles - Monitor WordPress security advisories - Use virtual patching through WAF for zero-day vulnerabilities - Implement file integrity monitoring
	Repudiation via Insufficient WordPress Audit Logging	Repudiation	Medium	Open		The WordPress process handles administrative actions, content changes, and user activities. If audit logging is insufficient or logs are not properly secured, malicious actors could deny performing harmful actions. Without comprehensive logs, it becomes difficult to investigate security incidents, attribute actions to specific users, or maintain compliance.	- Install WordPress activity logging plugins (e.g., WP Activity Log) - Log all administrative actions, content changes, and login attempts - Store logs in centralized, tamper-proof logging system - Implement log retention policies - Enable real-time alerting for critical actions - Ensure logs include timestamps, user identifiers, IP addresses - Regularly review logs for suspicious activities - Implement log integrity verification