# Executive Summary

Microsoft Presidio, an open-source PII detection and anonymization framework, has a generally strong security posture but requires careful deployment to avoid known risks. To date, **no official CVEs or security advisories have been published** for Presidio’s core software[[1]](https://github.com/microsoft/presidio/security#:~:text=Disclosure%20aka). However, recent developments highlight **two key areas of concern**: vulnerabilities in Presidio’s **third-party dependencies** and **configuration/security gaps in container deployments**. In 2025, the Presidio team released updates addressing critical issues in libraries used by Presidio (e.g. fixing known CVEs in Requests, Streamlit, and PyTorch)[[2]](https://github.com/microsoft/presidio/releases#:~:text=), underscoring that outdated Presidio versions may carry inherited vulnerabilities. Additionally, a **zero-day vulnerability** was disclosed in October 2025 that allows crafted inputs to **bypass Presidio’s data masking**, potentially exposing sensitive data until a patch is applied[[3]](https://hoop.dev/blog/microsoft-confirms-zero-day-vulnerability-in-presidio-data-protection-library/#:~:text=Microsoft%20confirmed%20a%20zero%20day,custom%20hardening%20or%20additional%20safeguards). Furthermore, Presidio’s default Docker containers do not enforce authentication or network restrictions on their REST API, meaning a poorly secured deployment could be openly accessible[[4]](https://github.com/microsoft/presidio/discussions/395#:~:text=I%27ve%20deployed%20Presidio%20v1%20at,have%20ideas%2C%20I%27m%20all%20ears). These factors could lead to unauthorized use or data leakage if Presidio is deployed without proper safeguards. This report details the known vulnerabilities and security concerns related to Presidio (especially in Dockerized environments) and provides recommendations to **mitigate risks when running Presidio containers in production**.

# Vulnerability Details

**Lack of Published CVEs for Presidio Core:** As of this report, there are *no publicly disclosed CVEs or GitHub security advisories directly against the Presidio application itself*[[1]](https://github.com/microsoft/presidio/security#:~:text=Disclosure%20aka). Microsoft’s Presidio repository follows a Coordinated Vulnerability Disclosure process, and any security flaws would typically be handled via the Microsoft Security Response Center (MSRC) rather than public issue trackers. The absence of published advisories suggests that **no critical exploitable bug in Presidio’s own code has been publicly confirmed**. Nevertheless, this does **not** mean Presidio is risk-free – vulnerabilities can exist in how it’s used or in its dependencies.

**Dependency Vulnerabilities:** Presidio relies on numerous third-party Python packages (NLP libraries, web frameworks, etc.), and vulnerabilities in those can impact Presidio deployments. In fact, **recent Presidio releases explicitly patched dependency-related vulnerabilities**. For example, version 2.2.360 updated components like *Streamlit* and *PyTorch* to address known CVEs[[2]](https://github.com/microsoft/presidio/releases#:~:text=). The maintainers also upgraded the Requests library to fix a critical issue (GHSA-9hjg-9r4m-mvj7) that could leak credentials via .netrc parsing (Requests’ vulnerability CVE-2024-47081)[[5]](https://github.com/microsoft/presidio/releases#:~:text=,Thanks%20%40SharonHart). These updates imply that **older Presidio Docker images (or installations)** bundled with unpatched versions of these libraries would be vulnerable. Any Presidio container built on an outdated base image or with outdated Python dependencies might contain high-severity CVEs (for instance, an older Streamlit release had a path traversal flaw on Windows[[6]](https://nvd.nist.gov/vuln/detail/CVE-2024-42474#:~:text=Users%20of%20hosted%20Streamlit%20app,file%20sharing%20feature%20is%20enabled), and certain PyTorch versions have had RCE or DoS issues). It is therefore crucial to monitor Presidio’s release notes and upgrade promptly, as **using an outdated Presidio image can leave known exploits in its software stack unpatched**[[2]](https://github.com/microsoft/presidio/releases#:~:text=). In summary, **the primary known vulnerabilities in Presidio are indirect**, coming from its ecosystem: keeping dependencies up-to-date is critical to security.

**Zero-Day Flaw in Analyzer (2025):** In mid-October 2025, a report surfaced of a **zero-day vulnerability in Presidio’s analyzer module**[[3]](https://hoop.dev/blog/microsoft-confirms-zero-day-vulnerability-in-presidio-data-protection-library/#:~:text=Microsoft%20confirmed%20a%20zero%20day,custom%20hardening%20or%20additional%20safeguards). This flaw allows attackers to **bypass certain input validation checks**, meaning carefully crafted text payloads can trick Presidio into failing to detect or redact sensitive data. In effect, an attacker could input data that *Presidio should mask*, yet the vulnerability causes the data to pass through unredacted – leading to **unintentional exposure of PII** in logs or API responses[[3]](https://hoop.dev/blog/microsoft-confirms-zero-day-vulnerability-in-presidio-data-protection-library/#:~:text=Microsoft%20confirmed%20a%20zero%20day,custom%20hardening%20or%20additional%20safeguards). Since this was a true “zero-day,” no patch was immediately available at the time of disclosure. Microsoft’s advisory (via MSRC) urged users to watch for unusual analyzer behavior and to apply the forthcoming patch as soon as it’s released[[7]](https://hoop.dev/blog/microsoft-confirms-zero-day-vulnerability-in-presidio-data-protection-library/#:~:text=system%20fails%20to%20mask%20or,logs%2C%20outputs%2C%20or%20API%20responses). The zero-day is a stark reminder that **logic vulnerabilities in Presidio’s detection/anonymization pipeline can have serious security implications**, effectively defeating the tool’s purpose. While details on this specific exploit are limited publicly, it reinforces the need for defense-in-depth: monitoring Presidio’s output for anomalies and not solely relying on it as a single gate for sensitive data protection. Until patched, mitigations included isolating the Presidio service, inspecting inputs, and adding custom validation or post-processing to catch data that might slip through[[7]](https://hoop.dev/blog/microsoft-confirms-zero-day-vulnerability-in-presidio-data-protection-library/#:~:text=system%20fails%20to%20mask%20or,logs%2C%20outputs%2C%20or%20API%20responses). (***Note:*** As this was a recent issue, users should check for any Presidio updates or MSRC bulletins post-October 2025 that address this vulnerability.)

**Docker Image and Deployment Security Issues:** Deploying Presidio as a Docker container introduces additional security considerations. The **Presidio Analyzer and Anonymizer services run as REST API microservices** by default (e.g. via Flask/FastAPI), listening on a network port. **By design, these containers do not include built-in authentication or encryption** – they are intended to be run in a protected environment. If a Presidio container is started on a server without network restrictions, *anyone who can reach the API port could send data to be analyzed or even modify the system’s configuration*. A 2021 discussion noted that the **admin APIs for managing custom recognizers were “open to anyone with access to the API,”** with no API keys or auth required[[4]](https://github.com/microsoft/presidio/discussions/395#:~:text=I%27ve%20deployed%20Presidio%20v1%20at,have%20ideas%2C%20I%27m%20all%20ears). This means that in a default deployment, malicious insiders or attackers who find the service reachable could potentially feed it data or change recognizer settings. While Presidio itself may not be exploitable in the classic sense (it doesn’t have an explicit remote code execution backdoor), **exposing it improperly could lead to sensitive data exfiltration or misuse** – for instance, an attacker could intentionally submit data that Presidio doesn’t handle well (as in the zero-day above) to cause leaks, or overwhelm the service to impact availability. Moreover, **container images can contain known OS-level vulnerabilities** if not regularly rebuilt. The Presidio Docker images are based on Python and Linux; without frequent updates, they might carry outdated system libraries. Security scans of generic official images (e.g. base Python images) often reveal many CVEs if updates aren’t applied. In line with this, Microsoft’s documentation emphasizes **container image scanning and regular updates to address CVEs**[[8]](https://hoop.dev/blog/microsoft-presidio-security-model-a-detailed-review/#:~:text=environment,maintain%20alignment%20with%20CVE%20patches). There have been no specific CVEs tied to “Presidio’s Docker image” itself, but it’s understood that **the risk grows over time as an image ages**. Finally, consider that Presidio processes highly sensitive data by nature – any weakness in container isolation or network configuration could amplify the impact of a breach. Running the container with default settings (e.g. as root user or with broad filesystem access) is not recommended, as it could worsen the consequences if the service were compromised.

# Mitigations and Best Practices for Secure Deployment

Deploying Microsoft Presidio in a containerized environment should follow strict security best practices to mitigate the above concerns:

* **Keep Presidio Updated:** Always use the **latest Presidio release** or Docker image to ensure you have all recent security patches. The development team often releases fixes for vulnerabilities in underlying packages (such as the Requests credential leak fix and other CVE patches)[[2]](https://github.com/microsoft/presidio/releases#:~:text=). Regularly monitor the Presidio GitHub releases and update your containers especially after any announced security fix. If a zero-day vulnerability is reported (like the 2025 case), plan to apply the patched version as soon as it’s available and consider temporarily isolating or limiting Presidio’s use until then.
* **Scan and Harden Container Images:** Incorporate container vulnerability scanning into your CI/CD pipeline for Presidio. Microsoft’s guidance stresses using image scanning tools to catch outdated packages and aligning your images with the latest CVE fixes[[8]](https://hoop.dev/blog/microsoft-presidio-security-model-a-detailed-review/#:~:text=environment,maintain%20alignment%20with%20CVE%20patches). If the official image is used, watch for updates on Docker Hub or the GitHub repo. If you build a custom Presidio image (e.g. to include additional recognizers or models), base it on a minimal, up-to-date OS image and update system packages to eliminate known vulnerabilities. Remove any unnecessary utilities or libraries from the image to reduce the attack surface. Also, **run the Presidio container with a non-root user** if possible (the Dockerfile or runtime options should be adjusted to drop privileges), and consider using Docker/Kubernetes security settings (like read-only file systems and limited Linux capabilities) to sandbox the process.
* **Network Segmentation and Access Control:** **Never expose the Presidio API directly to the public internet.** Deploy Presidio containers in an internal network or behind an API gateway/proxy. In Kubernetes, use Network Policies to restrict which pods or services can communicate with the Presidio service[[8]](https://hoop.dev/blog/microsoft-presidio-security-model-a-detailed-review/#:~:text=environment,maintain%20alignment%20with%20CVE%20patches). At the very least, bind the container’s service to a localhost or private interface if running standalone. For production use, front the Presidio API with an authentication layer – for example, an Nginx/Traefik proxy that requires an API key or JWT token, or integrate it into an existing auth scheme. This ensures that only authorized applications or users can call the analyzer/anonymizer endpoints. Without such a gate, **anyone who can reach the Presidio port could submit data or commands** (as highlighted by the lack of default auth for certain API calls[[4]](https://github.com/microsoft/presidio/discussions/395#:~:text=I%27ve%20deployed%20Presidio%20v1%20at,have%20ideas%2C%20I%27m%20all%20ears)). By implementing an API key or other auth mechanism, you reduce the risk of unauthorized use. Additionally, enable TLS encryption for any network traffic to Presidio. Even if you deploy it internally, use HTTPS for calling the Presidio API or leverage mutual TLS in a service mesh – this prevents eavesdropping or man-in-the-middle attacks on sensitive data in transit[[8]](https://hoop.dev/blog/microsoft-presidio-security-model-a-detailed-review/#:~:text=environment,maintain%20alignment%20with%20CVE%20patches).
* **Configuration and Data Handling:** Configure Presidio with security in mind. Ensure that **no sensitive data is logged in plaintext** – review logging levels and messages (Presidio by default shouldn’t log PII, but if you wrap it in your own app, be cautious). Where possible, use Presidio’s abilities to detect secrets and PII in logs (for example, using Presidio itself to scan logs is suggested as a privacy measure[[9]](https://github.com/microsoft/code-with-engineering-playbook/blob/016770e43d8a75be87b98c000c049f07c4a6e6f8/docs/observability/logs-privacy.md#L8-L16)[[10]](https://github.com/microsoft/code-with-engineering-playbook/blob/016770e43d8a75be87b98c000c049f07c4a6e6f8/docs/observability/logs-privacy.md#L54-L62)). If Presidio is writing any telemetry or audit data, direct it to secure storage. Consider enabling audit logs for who/what is calling the Presidio API (this may be handled by your API gateway or by custom code around Presidio’s service). Monitor these logs for anomalies, such as bursts of requests or unusual input patterns, which could indicate abuse or an attempt to exploit a flaw.
* **Isolation and Least Privilege:** Run Presidio in an isolated environment following the principle of least privilege. If using Docker, deploy it on a host with up-to-date Docker daemon security patches. Do not run other high-risk services on the same host that could allow cross-container attacks. In Kubernetes, run Presidio in its own namespace with appropriate RBAC rules – limit the container’s permissions and deny it access to cloud instance metadata or other services it doesn’t need. Microsoft’s documentation recommends using **Kubernetes with proper RBAC and network policy for Presidio**[[8]](https://hoop.dev/blog/microsoft-presidio-security-model-a-detailed-review/#:~:text=environment,maintain%20alignment%20with%20CVE%20patches). Also, ensure the container only has access to the data it needs. Mount directories as read-only if Presidio only needs to read (for example, if you mount model files). Avoid mounting any host sensitive directories into the container. By tightly scoping Presidio’s environment, even if an attacker managed to break out of the Presidio application, they would have minimal further access.
* **Bundle Models & Avoid External Dependencies at Runtime:** For **air-gapped or high-security deployments**, it’s recommended to **package all necessary NLP models and resources inside the Presidio container** and use a local model registry. This prevents the container from reaching out to external sources (like downloading spaCy models or transformers from the internet) during startup or execution[[11]](https://hoop.dev/blog/deploying-microsoft-presidio-in-isolated-environments/#:~:text=Deploying%20Presidio%20in%20isolation%20requires,pulling%20code%20from%20unverified%20sources). By running Presidio in a completely self-contained manner (no outbound network calls), you **eliminate supply-chain risks** where an attacker might hijack an external dependency. In isolated environments, organizations have successfully run Presidio with local spaCy pipelines and offline resources to achieve full functionality without internet access[[11]](https://hoop.dev/blog/deploying-microsoft-presidio-in-isolated-environments/#:~:text=Deploying%20Presidio%20in%20isolation%20requires,pulling%20code%20from%20unverified%20sources). This not only improves security but also ensures consistency (the same models/versions are used in each deployment, reducing unpredictability).
* **Apply Defense-in-Depth for Data Protection:** Recognize that Presidio is one layer of defense. Its accuracy in detecting PII is high but not perfect; moreover, as the zero-day incident showed, it’s possible for certain inputs to slip through. Therefore, treat Presidio’s output with scrutiny in critical workflows. For example, if Presidio is anonymizing data that will be stored or sent out, consider adding a secondary validation step – e.g., scanning the “anonymized” result with a different pattern matcher to confirm nothing sensitive remains. Additionally, **monitor the Presidio service** for performance and unusual activity. A denial-of-service against Presidio (feeding extremely large or complex inputs, for instance) could render your data pipeline ineffective. Use rate limiting on the API and timeouts to mitigate resource exhaustion attacks. If using container orchestration, enable auto-scaling or restarting of the Presidio pod in case of crashes.
* **Follow Container Security Benchmarks:** Adhere to general Docker/Kubernetes security best practices. Some recommendations include: using minimal base images (alpine or distroless if possible), verifying image signatures or checksums (to ensure you’re running the official Microsoft Presidio image and not a tampered one), and restricting outbound network access from the Presidio container (it typically shouldn’t need to call out, unless using external OCR or NER services). Tools like Azure Defender for Containers or other container security platforms can be leveraged to continuously assess Presidio containers for vulnerabilities and misconfigurations. Microsoft’s OpenSSF badge and removal of false-positive tools in the CI (e.g., Terrascan)[[12]](https://github.com/microsoft/presidio/releases#:~:text=,IAC%20repository%20%28%231691%29%20%28Thanks%20%40Copilot) indicate an ongoing effort to align with security best practices; as users, you should do the same in your environment.

By implementing the above measures, organizations can significantly reduce the attack surface and impact of potential vulnerabilities when running Microsoft Presidio in Docker. In summary, **secure your Presidio deployment as you would any sensitive microservice**: keep it patched, limit access, run it with least privilege, and monitor it continuously. These steps will help protect against known issues (like dependency CVEs and open APIs) and also bolster defenses against future unknown threats.

# References

1. **Presidio GitHub Security Status** – Official repository’s security page confirming no published advisories for Presidio[[1]](https://github.com/microsoft/presidio/security#:~:text=Disclosure%20aka).
2. **Presidio Release Notes v2.2.360** – Highlights of dependency updates (Streamlit, PyTorch, Requests) to fix CVE-related vulnerabilities[[2]](https://github.com/microsoft/presidio/releases#:~:text=).
3. **Hoop.dev – *Presidio Security Model Review*** – Blog detailing Presidio’s security architecture and recommending container security practices like network policies, RBAC, TLS, and image scanning[[8]](https://hoop.dev/blog/microsoft-presidio-security-model-a-detailed-review/#:~:text=environment,maintain%20alignment%20with%20CVE%20patches).
4. **Hoop.dev – *Zero-Day Vulnerability in Presidio*** – Report on an Oct 2025 zero-day in Presidio’s analyzer that bypasses data masking (input validation flaw) and guidance for mitigation pending a patch[[3]](https://hoop.dev/blog/microsoft-confirms-zero-day-vulnerability-in-presidio-data-protection-library/#:~:text=Microsoft%20confirmed%20a%20zero%20day,custom%20hardening%20or%20additional%20safeguards)[[7]](https://hoop.dev/blog/microsoft-confirms-zero-day-vulnerability-in-presidio-data-protection-library/#:~:text=system%20fails%20to%20mask%20or,logs%2C%20outputs%2C%20or%20API%20responses).
5. **GitHub Discussion – *Presidio API Auth Key*** – User discussion pointing out the lack of default authentication on Presidio’s API (e.g. open access to custom recognizer endpoints) and the need to implement access control externally[[4]](https://github.com/microsoft/presidio/discussions/395#:~:text=I%27ve%20deployed%20Presidio%20v1%20at,have%20ideas%2C%20I%27m%20all%20ears).
6. **Hoop.dev – *Deploying Presidio in Isolated Environments*** – Article advising on running Presidio in offline/secure environments by containerizing with all dependencies bundled and no external calls, to eliminate supply-chain and network risks[[11]](https://hoop.dev/blog/deploying-microsoft-presidio-in-isolated-environments/#:~:text=Deploying%20Presidio%20in%20isolation%20requires,pulling%20code%20from%20unverified%20sources).

[[1]](https://github.com/microsoft/presidio/security#:~:text=Disclosure%20aka) Security Overview · microsoft/presidio · GitHub

<https://github.com/microsoft/presidio/security>

[[2]](https://github.com/microsoft/presidio/releases#:~:text=) [[5]](https://github.com/microsoft/presidio/releases#:~:text=,Thanks%20%40SharonHart) [[12]](https://github.com/microsoft/presidio/releases#:~:text=,IAC%20repository%20%28%231691%29%20%28Thanks%20%40Copilot) Releases · microsoft/presidio · GitHub

<https://github.com/microsoft/presidio/releases>

[[3]](https://hoop.dev/blog/microsoft-confirms-zero-day-vulnerability-in-presidio-data-protection-library/#:~:text=Microsoft%20confirmed%20a%20zero%20day,custom%20hardening%20or%20additional%20safeguards) [[7]](https://hoop.dev/blog/microsoft-confirms-zero-day-vulnerability-in-presidio-data-protection-library/#:~:text=system%20fails%20to%20mask%20or,logs%2C%20outputs%2C%20or%20API%20responses) Microsoft Confirms Zero Day Vulnerability in Presidio Data Protection Library

<https://hoop.dev/blog/microsoft-confirms-zero-day-vulnerability-in-presidio-data-protection-library/>

[[4]](https://github.com/microsoft/presidio/discussions/395#:~:text=I%27ve%20deployed%20Presidio%20v1%20at,have%20ideas%2C%20I%27m%20all%20ears) API Auth Key · microsoft presidio · Discussion #395 · GitHub

<https://github.com/microsoft/presidio/discussions/395>

[[6]](https://nvd.nist.gov/vuln/detail/CVE-2024-42474#:~:text=Users%20of%20hosted%20Streamlit%20app,file%20sharing%20feature%20is%20enabled) CVE-2024-42474 Detail - NVD

<https://nvd.nist.gov/vuln/detail/CVE-2024-42474>

[[8]](https://hoop.dev/blog/microsoft-presidio-security-model-a-detailed-review/#:~:text=environment,maintain%20alignment%20with%20CVE%20patches) Microsoft Presidio Security Model: A Detailed Review

<https://hoop.dev/blog/microsoft-presidio-security-model-a-detailed-review/>

[[9]](https://github.com/microsoft/code-with-engineering-playbook/blob/016770e43d8a75be87b98c000c049f07c4a6e6f8/docs/observability/logs-privacy.md#L8-L16) [[10]](https://github.com/microsoft/code-with-engineering-playbook/blob/016770e43d8a75be87b98c000c049f07c4a6e6f8/docs/observability/logs-privacy.md#L54-L62) logs-privacy.md

<https://github.com/microsoft/code-with-engineering-playbook/blob/016770e43d8a75be87b98c000c049f07c4a6e6f8/docs/observability/logs-privacy.md>

[[11]](https://hoop.dev/blog/deploying-microsoft-presidio-in-isolated-environments/#:~:text=Deploying%20Presidio%20in%20isolation%20requires,pulling%20code%20from%20unverified%20sources) Deploying Microsoft Presidio in Isolated Environments

<https://hoop.dev/blog/deploying-microsoft-presidio-in-isolated-environments/>