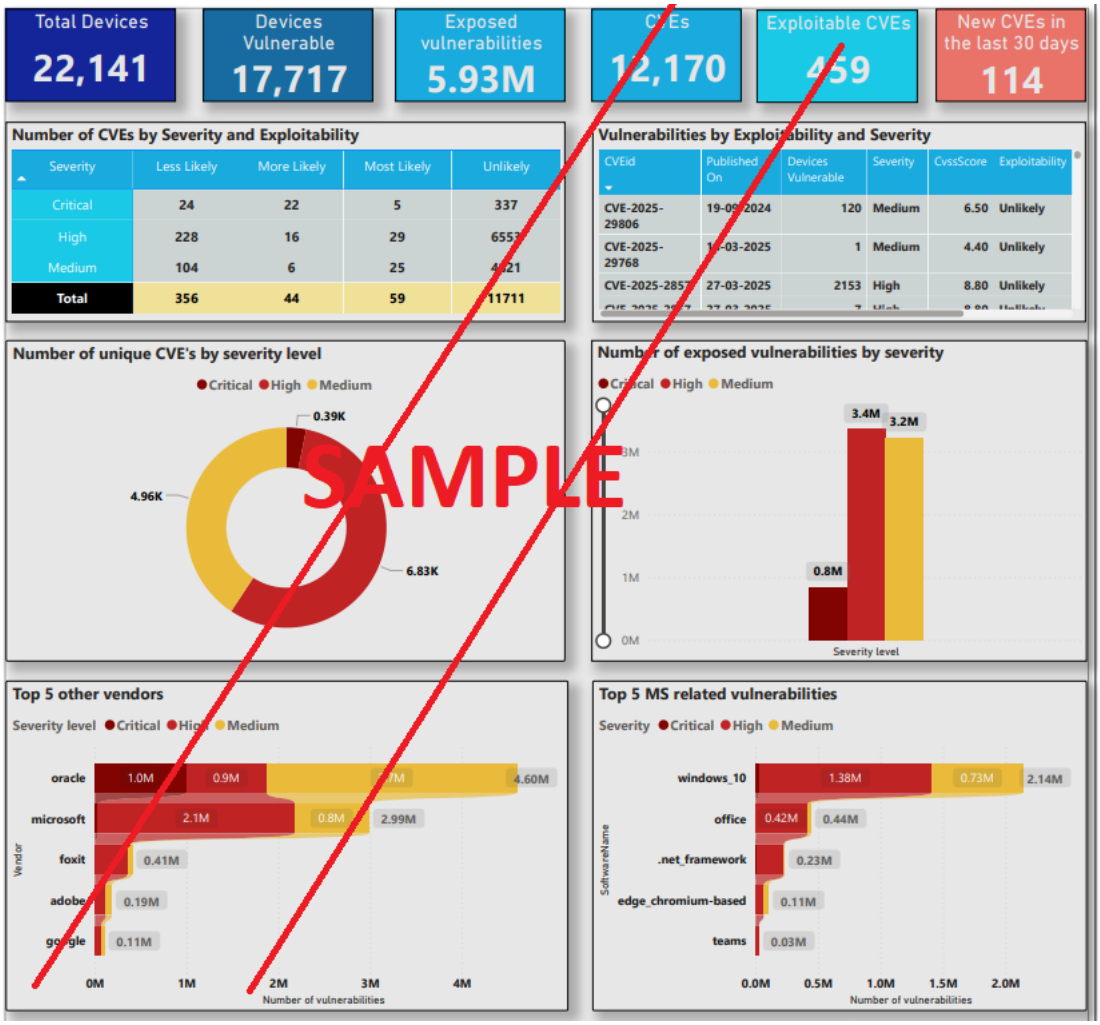
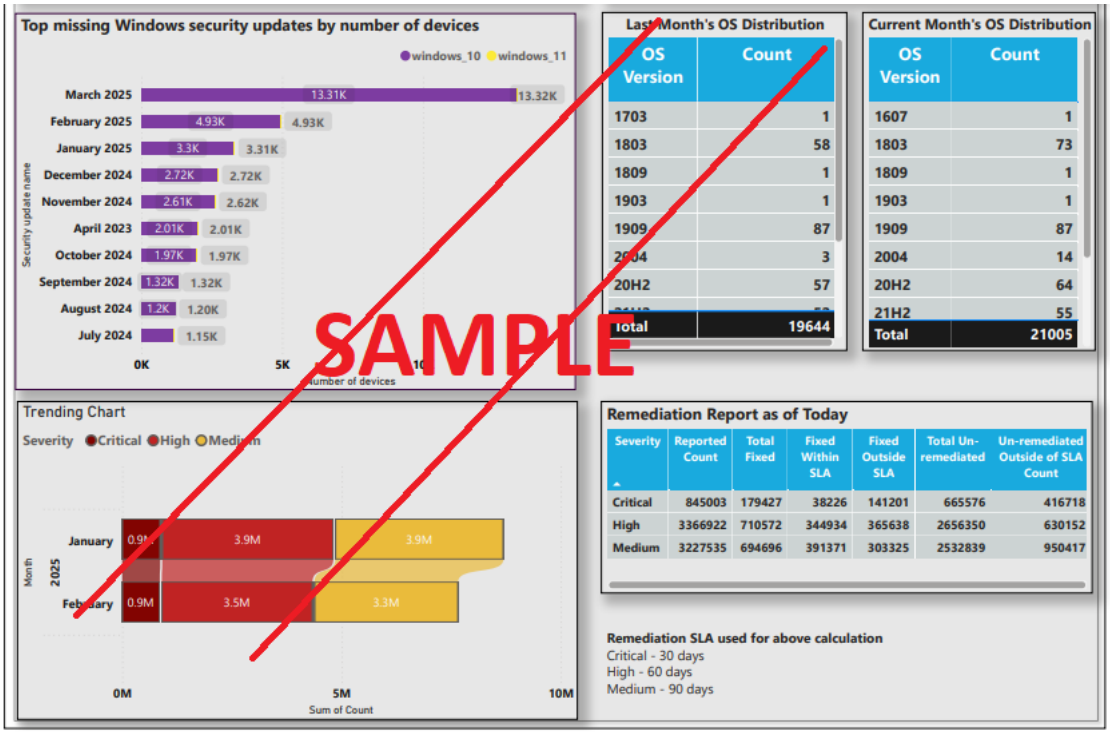
Defender Dashboard

Contents

Executive Summary

The Defender Dashboard solution provides a centralized, dynamic view of security alerts and endpoint status across approximately 30,000 devices. **It features a Python backend that fetches data via Microsoft Defender APIs, performs pre-filtering, and maintains historical data for remediation tracking.** This processed data is then visualized using a Power BI frontend. Refreshed twice daily, the system replaces inefficient manual reporting and scattered data, delivering timely, accurate insights with advanced filtering capabilities to stakeholders including the CISO, SOC, IT Admins, and Remediation teams.



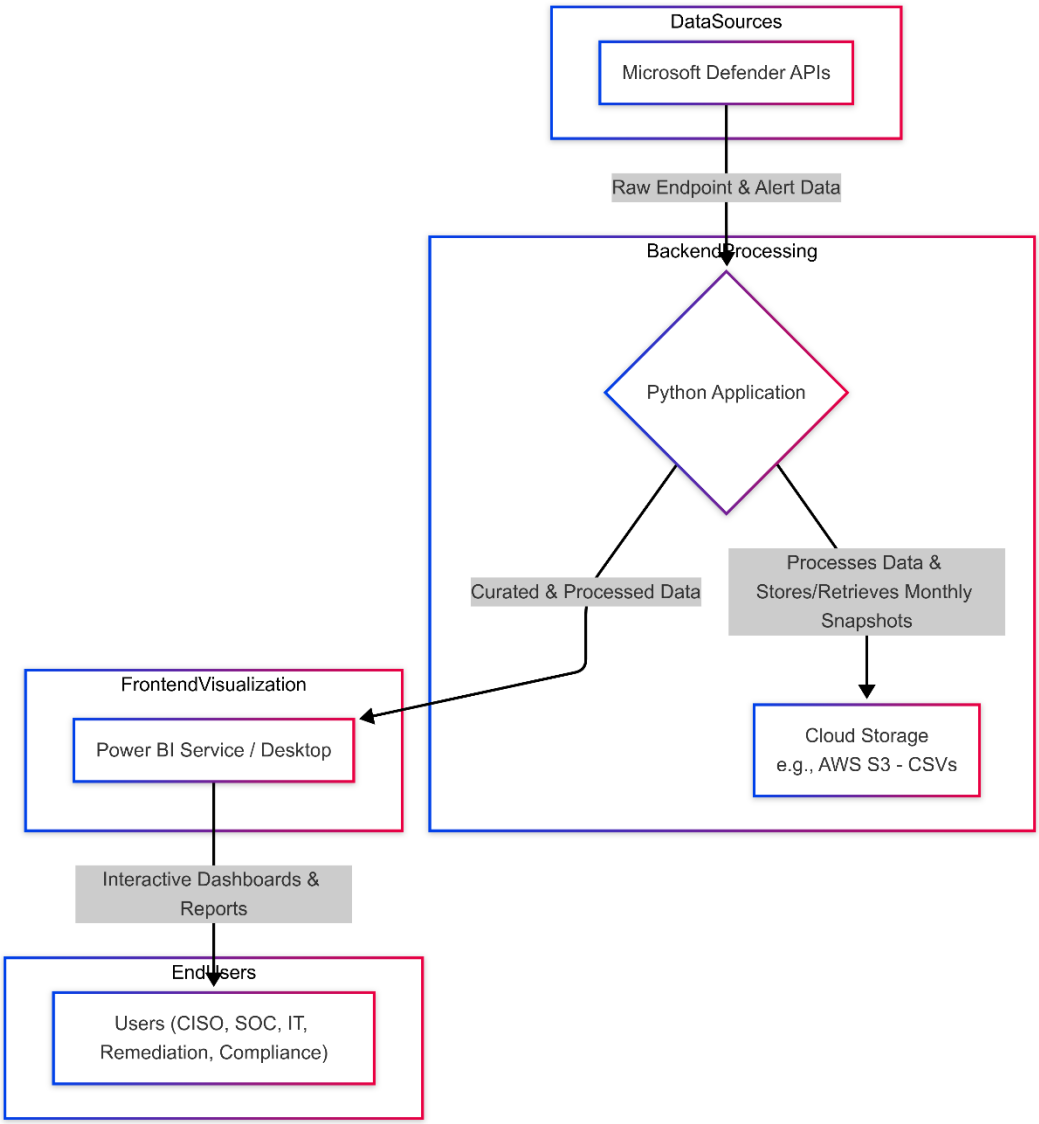

Top missing Windows security updates by number of devices
windows_10
windows_11
March 2025
February 2025
January 2025
December 2024
November 2024
April 2023
October 2024
September 2024
August 2024
July 2024
Security update name
Number of devices
Last Month's OS Distribution
OS Version
Count
Current Month's OS Distribution
Total
19644
21005
SAMPLE
Trending Chart
Severity
Critical
High
Month
2025
January
February
Sum of Count
Remediation Report as of Today
Severity
Reported Count
Total Fixed
Fixed Within SLA
Fixed Outside SLA
Total Un-remediated
Un-remediated Outside of SLA Count
Critical
845003
179427
38226
141201
665576
416718
High
3366922
710572
344934
365638
2656350
630152
Medium
3227535
694696
391371
303325
2532839
950417
Remediation SLA used for above calculation
Critical - 30 days
High - 60 days
Medium - 90 days

The Challenge: Monitoring Defender Data Before the Dashboard

Before the implementation of the Defender Dashboard, monitoring the security posture across approximately 30,000 endpoints using Microsoft Defender faced significant hurdles:

- **Data Fragmentation & Manual Collection:** Critical security data was highly fragmented, stored across various SharePoint locations and requiring analysts to perform laborious, time-consuming manual downloads of reports directly from the Defender UI, often on a per-device or per-report type basis.
- **Static & Delayed Reporting:** The established reporting process relied heavily on manually compiling this downloaded data into Excel spreadsheets and PowerPoint presentations. This typically resulted in only a static, monthly snapshot, offering an outdated view of the security landscape.
- **Lack of Timely & Actionable Insights:** The infrequent, manual reporting cycle made it extremely difficult to get timely insights into emerging threats (including potential zero-days requiring immediate attention), track daily or weekly trends, or quickly correlate alerts with endpoint status for rapid response.
- **Scalability & Efficiency Issues:** Manually handling, filtering, and analyzing gigabytes of data generated by tens of thousands of endpoints was inherently inefficient, prone to errors, and did not scale effectively.
- **Significant Negative Consequences:** This fragmented and manual approach directly led to:
    - **Delayed detection and response** capabilities for security incidents.
    - An **incomplete and often outdated understanding** of the organization's overall security posture.
    - **Inefficient use of valuable analyst time**, with significant effort spent on data gathering and report preparation instead of proactive security analysis and threat mitigation.

High-Level Conceptual Architecture Diagram



The Solution: Building the Defender Dashboard

Conceptual Overview:

The core concept behind the Defender Dashboard was to create a robust, centralized system to overcome the significant limitations of previous manual methods and the complexities of directly connecting visualization tools to the source APIs at scale. The solution was architected as a two-tier system:

1. **Python Backend:** Developed to handle the demanding task of interfacing with various Microsoft Defender APIs across multiple operating companies (~30,000 assets total). This backend automates the fetching and consolidation of data from these diverse sources, addressing API limitations and complexity. Crucially, it also manages historical data, performing monthly backups via scripts to enable trend analysis and comparison over time.
2. **Power BI Frontend:** Leveraged as the visualization and interaction layer. It consumes the processed, centralized data curated by the Python backend. This frontend provides stakeholders with a unified dashboard view, replacing manual reporting. It incorporates advanced filtering capabilities (using DAX) and allows users to download specific reports, and evidence extracts for remediation purposes, tailored to their needs.

This approach was chosen for its ability to reliably handle the scale and complexity of the environment, provide necessary data processing and historical context capabilities, and ultimately deliver far more detailed, timely, and actionable vulnerability insights across all workstations compared to the fragmented and labor-intensive previous processes.

Key Features & Functionality:

The Defender Dashboard system incorporates a range of features across its Python backend and Power BI frontend to deliver comprehensive endpoint security visibility and reporting:

Python Backend Capabilities:

- **Targeted Data Aggregation:** Regularly fetches key data points via Microsoft Defender APIs across all operating companies (~30k assets), including full device inventories (total counts, EOL status), software inventories, and detailed vulnerability information (categorized by CVE, severity).
- **Intelligent Data Processing:** Consolidates data fetched from various sources, applying logic to purge unnecessary information and, critically, performs **deduplication of superseded vulnerabilities** to maintain data accuracy and relevance.

- **Historical Analysis & Inferred Remediation Tracking:** Stores monthly snapshots of device and vulnerability data in cloud storage (AWS S3 using CSVs). By comparing the current month's data against the previous month's snapshot, the system **infers which devices/vulnerabilities have been remediated** (i.e., those no longer appearing). This novel approach overcomes a limitation in direct Defender reporting and enables accurate tracking of remediation progress and historical trending.
- **SLA Compliance Monitoring:** Utilizes the processed vulnerability data and historical context to automatically identify assets that are in breach of predefined remediation Service Level Agreements (SLAs).
- **Data Preparation:** Structures and prepares the consolidated, cleansed, and enriched data specifically for efficient consumption and visualization by the Power BI frontend.

Power BI Frontend Features:

- **Unified Dashboards & KPIs:** Presents key metrics and insights through dedicated dashboard views, including:
    - Overall device counts (total vs. currently vulnerable).
    - Vulnerability breakdowns (by CVE, severity level, unique vulnerability count).
    - Lists of devices missing recent critical patches (e.g., "Patch Tuesday" updates).
    - Identification and tracking of devices running End-of-Life (EOL) operating systems.
- **Advanced Filtering & Focus:** Implements powerful filtering capabilities using DAX queries, allowing diverse stakeholders (CISO, SOC, IT, Remediation teams) to drill down into specific data subsets relevant to their roles (e.g., filtering for non-remediated critical vulnerabilities within a specific OpCo or date range).
- **Trend Visualization:** Displays clear visualizations of historical trends based on the backend's monthly snapshot data, effectively illustrating progress in vulnerability posture and remediation efforts over time.
- **Customizable Data Export:** Enables users to easily export filtered data views, vulnerability lists, and evidence directly from the dashboard interface into **Excel or PDF formats**, facilitating offline analysis, task assignment, and compliance reporting.

My Contribution & Development Process:

As the **Lead Developer** for the Defender Dashboard project, I had end-to-end responsibility for its design, development, and implementation, encompassing both the Python backend and the Power BI frontend. My key contributions included:

- **Requirements & Design:** Collaborated directly with stakeholders, including the CISO, to define the specific requirements and visualizations needed for the final dashboard, ensuring it met leadership and operational needs.
- **API Research & Architecture:** Conducted thorough research into the Microsoft Defender API documentation to identify the optimal endpoints and data extraction methods. Based on this research and the project requirements, I designed the overall system architecture, defining the data flow, the Python processing logic, and the use of AWS S3 for historical data storage.
- **Backend Development (Python):** Personally, developed the Python scripts responsible for:
    - Fetching data from Defender APIs using the requests library.
    - Processing, consolidating, and filtering large datasets efficiently using pandas.
    - Interacting with AWS S3 using boto3 for storing and retrieving the historical monthly CSV snapshots crucial for trend analysis and remediation tracking.
- **Cloud Deployment & Automation:** Utilized **AWS Lambda** to deploy and schedule the Python backend scripts, ensuring automated data refreshes occurred reliably twice daily.
- **Frontend Development (Power BI):** Developed the interactive dashboards and reports within Power BI, implementing the required visualizations, DAX measures for advanced filtering, and data export functionalities.
- **Best Practices & Testing:** Employed standard development practices, including **version control (presumably Git)** for the Python codebase. The system also underwent stress testing to validate its performance and scalability handling data from ~30,000 assets.

Impact & Organizational Benefits

The implementation of the Defender Dashboard delivered significant and measurable improvements to the organization's endpoint security monitoring, reporting, and overall posture management across approximately 30,000 assets:

- **Unified & Timely Visibility:** By replacing fragmented data sources (SharePoint, manual downloads) and static monthly reports, the dashboard provided a **single, centralized source of truth**. With twice-daily refreshes, it offered **near real-time visibility** into the security posture, dramatically improving situational awareness for all stakeholders (CISO, SOC, IT Admins, Remediation, Compliance).
- **Significant Efficiency Gains:** The automation driven by the Python backend **eliminated hours of laborious manual effort** previously spent by analysts on data collection, consolidation using Excel, and report preparation using PowerPoint. This freed up significant analyst time, allowing them to focus on higher-value activities like threat analysis and proactive security measures.
- **Faster Detection & Response:** Access to timely, consolidated data and advanced filtering capabilities enabled the SOC and relevant teams to **more quickly identify, prioritize, and respond** to critical alerts and potential emerging threats like zero-days, reducing overall risk exposure.
- **Accurate Remediation Tracking & Trend Analysis:** The custom historical data comparison logic provided **reliable tracking of remediation progress** – a capability previously lacking – and enabled meaningful **trend analysis** of the vulnerability landscape over time. Automated SLA breach identification also improved accountability.
- **Improved Data Handling & Accessibility:** The system proved capable of efficiently processing **large volumes of data (Gigabytes)**. The Power BI frontend made complex security information easily accessible and digestible for diverse audiences through interactive visualizations and tailored filtering, overcoming previous scalability issues and reporting limitations. The ability to export filtered data provided necessary evidence for remediation and compliance tasks.

Conclusion

In summary, the Defender Dashboard project successfully transformed the organization's approach to endpoint security monitoring by replacing fragmented, manual processes with an automated, scalable solution. Leveraging a Python backend hosted on AWS (Lambda, S3) for robust data engineering and a Power BI frontend for intuitive visualization, the system provides timely, actionable insights across approximately 30,000 assets. This initiative demonstrably improved efficiency, visibility, and response capabilities. It highlights my expertise in system architecture design, API integration, Python development for data processing, cloud automation (AWS), and utilizing Business Intelligence tools (Power BI, DAX) to meet complex security and reporting requirements for diverse stakeholders, from technical teams to the CISO.