

Measure Killer Security Information and Documentation

Version 2.9.0

This document outlines the security details and data handling practices of Measure Killer.

Data Handling and Privacy Assurance

Measure Killer is meticulously designed with user privacy and data security as paramount concerns. It is important for users to understand the scope and way that Measure Killer interacts with Power BI.

Metadata-Only Interaction

Measure Killer only interacts and utilizes metadata from Power BI Desktop or the Power BI Service. At no point does Measure Killer read, store, or process actual data contained within Power BI reports or models. This strict limitation to metadata ensures that sensitive data remains confidential and secure.

What is considered Metadata?

Metadata is considered the following (examples):

- Names or artifacts (report, semantic models, workspaces, users/email addresses)
- DAX and M expressions including comments
- Names of measures, tables, calculated columns, calculated tables, calculation groups, field parameters etc.
- Uncompressed size of columns
- All definitions (e.g. name of reports, pages, visuals, visual titles, visual size and other metadata)

Local Processing of Metadata

All processing and analysis of metadata conducted by Measure Killer is performed locally on the user's machine through the installed client software. This approach ensures that the operations on metadata do not involve or require any external processing or storage facilities. By doing so, Measure Killer upholds a high standard of data protection and minimizes potential security risks. There is no database that is hosted, managed, or used to process any kind of information, everything is done on the user's machine and not transmitted anywhere besides when using XMLA or REST API calls with Microsoft directly.

XMLA Endpoint Connection

To retrieve metadata from semantic models in the Power BI Service, XMLA endpoints are used. The authentication happens via the entered Microsoft account (see adomd client for more specifics on this).

API Calls to Microsoft for Metadata Acquisition

The only other external interaction Measure Killer undertakes involves REST API calls to Microsoft's Power BI Service for the acquisition of metadata. These API calls are made in

compliance with all relevant security protocols. Authentication happens via browser interaction (OAUTH2). Once metadata has been acquired, all subsequent analyses and operations are carried out locally. See the list at the bottom of this document for a list of REST API calls done.

Online / Offline Modes

Measure Killer is designed to be able to operate both online and offline:

- In the offline modes, Measure Killer connects to the local instance of Analysis Services to retrieve the metadata it needs (Modes 1 and 2). In these modes the only time there is an interaction with the internet is upon launch to verify valid licensing (see License Verification below for details)
- Online modes (Modes 3, 4 and 5) are triggered when an active internet connection is detected and when the user has a valid license. In these modes there are external connections happening (only REST API calls and XMLA connections to Microsoft as outlined above).

License Verification and Datetime Check

- Measure Killer uses API calls to acquire the current date and time: GET <https://worldtimeapi.org/api/timezone/Europe/London> or, as a fallback, GET <https://timeapi.io/api/Time/current/zone?timeZone=Europe/London> to ensure the license has not expired.

Version Check and User Notification

- To inform users about available updates, Measure Killer queries GET <https://measurekiller.com/downloads/> to compare the installed version against the most recent version available the website.
- There is one additional GET request to extract text and write it in the main UI window from this URL GET <https://en.brunner.bi/post/measure-killer-feedback-1>

License Data EventStream Endpoint

For the purpose of validating active licenses and ensuring compliance, Measure Killer transmits a minimal set of license-related metadata to a secure EventStream endpoint. The transmitted payload includes the local machine username, license key, installed Measure Killer version, current UTC timestamp, public IP address, and machine name. No Power BI metadata, report content, or user data is ever included in this process. All data is sent exclusively over encrypted channels (HTTPS/TLS) and is handled in accordance with strict security and privacy practices.

API Endpoints and Security Measures

- Secure requests (HTTPS) are made to Power BI API endpoints, ensuring data in transit is encrypted.
- Access tokens are securely passed in request headers, with responses parsed as JSON for internal processing.

List of REST API Calls

- Datasets - Get Datasets In Group: GET
<https://api.powerbi.com/v1.0/myorg/groups/{workspace.id}/datasets>
- Datasets - Get Refresh History In Group: GET
<https://api.powerbi.com/v1.0/myorg/groups/{workspace.id}/datasets/{dataset.id}/refreshes>
- Datasets - Get Refresh History: GET
<https://api.powerbi.com/v1.0/myorg/datasets/{dataset.id}/refreshes>
- Datasets - Execute Queries In Group: GET
<https://api.powerbi.com/v1.0/myorg/groups/{workspace.id}/datasets/{dataset.id}/executeQueries>
- Reports - Get Reports In Group: GET
<https://api.powerbi.com/v1.0/myorg/groups/{workspace.id}/reports>
- Reports - Get Datasources In Group: GET
<https://api.powerbi.com/v1.0/myorg/groups/{workspace.id}/reports/{report.id}/datasources>
- Reports - Get Datasources: GET
<https://api.powerbi.com/v1.0/myorg/reports/{report.id}/datasources>
- Reports - Export Report In Group: GET
<https://api.powerbi.com/v1.0/myorg/groups/{workspace.id}/reports/{report.id}/Export>
- Reports - Export Report: GET
<https://api.powerbi.com/v1.0/myorg/reports/{report.id}/Export>
- Groups - Get Groups: GET
<https://api.powerbi.com/v1.0/myorg/groups/{workspace.id}/users>
- Groups - Get Groups: GET <https://api.powerbi.com/v1.0/myorg/groups>

List of REST API Admin Calls

- Admin - Get Capacities As Admin: GET
<https://api.powerbi.com/v1.0/myorg/admin/capacities>
- Admin - WorkspaceInfo GetScanStatus: GET
<https://api.powerbi.com/v1.0/myorg/admin/workspaces/scanstatus/{scan.id}>
- Admin - WorkspaceInfo GetScanResult: GET
<https://api.powerbi.com/v1.0/myorg/admin/workspaces/scanresult/{scan.id}>
- Admin - WorkspaceInfo PostWorkspaceInfo: POST
<https://api.powerbi.com/v1.0/myorg/admin/workspaces/getinfo?lineage=true>
- Admin - Get Activity Events: POST
<https://api.powerbi.com/v1.0/myorg/admin/activityevents>
- Admin - Groups GetGroupsAsAdmin: GET
<https://api.powerbi.com/v1.0/myorg/admin/groups>

- Admin - Groups AddUserAsAdmin: POST
<https://api.powerbi.com/v1.0/myorg/admin/groups/{workspace.id}/users>
- Admin - Groups DeleteUserAsAdmin: DELETE
<https://api.powerbi.com/v1.0/myorg/admin/groups/{workspace.id}/users/{user.email}>
- Admin - Users GetUserArtifactAccessAsAdmin: GET
<https://api.powerbi.com/v1.0/myorg/admin/users/{user.email}/artifactAccess>
- AAdmin - Reports GetReportSubscriptionsAsAdmin: GET
<https://api.powerbi.com/v1.0/myorg/admin/reports/{reportId}/subscriptions>

List of Fabric API Calls

- Items - Get Item Definition: POST
<https://api.fabric.microsoft.com/v1/workspaces/{workspace.id}/items/{item.id}/getDefinition>
- Long Running Operations - Get Operation State: GET
<https://api.fabric.microsoft.com/v1/operations/{operation.id}>
- Long Running Operations - Get Operation Result: GET
<https://api.fabric.microsoft.com/v1/operations/{operation.id}/result>
- Domains - List Domain Workspaces GET
<https://api.fabric.microsoft.com/v1/admin/domains/{domainId}/workspaces>
- Domains - List Domains GET <https://api.fabric.microsoft.com/v1/admin/domains>

WIX

- Create Contact: POST <https://www.wixapis.com/contacts/v4/contacts>

License check and network time Protocol (NTP)

Measure Killer uses NTP via the ntplib library in python to enforce license expiration. We consult the time.windows.com server and as a fallback pool.ntp.org

MK License verification uses the Azure Event Hubs libraries for python . To make this work, we need outbound connectivity to the following:

- **Domain:** *.servicebus.windows.net
- **Ports:** 5671 (AMQP over TLS), 443 (AMQP over WebSockets) and as a fallback if 5671 cannot be opened.
- **Protocol:** TLS-encrypted AMQP traffic (not plain HTTP).

Additional Security and Testing Approach

We use a combination of automated scanning and testing along with manual reviews to ensure Measure Killer's security before and after each release. Below are our main practices:

- **Software composition analysis:** When we begin development on a new version, we check all open-source dependencies for known vulnerabilities using pip-audit and OSS Index. Because many issues can be resolved by library updates, we track each new release. We then review any security fixes and confirm compatibility with Measure Killer. If it is stable and relevant, we incorporate it into our codebase.

- **Static code analysis:** We run automated static analysis with tools such as Bandit (a Python security analyzer) and PyLint (for code quality and style). These tools flag potential security weaknesses. Our developers then review each flagged item to determine if it represents a real vulnerability or a false positive.
- **Automated Testing and Code Review:** We use pytest for extensive unit and integration testing. Security-wise, our tests focus on:
 - Validation and Parsing Logic: Ensuring .pbix and .rdl files are parsed securely.
 - User Authentication: Verifying usage of the azure.identity library and tabular connections.
 - Error-Handling and Fail-Safe Behavior: Simulating network or parsing failures using mocks, and testing responses to Power BI API errors.
 - Regression Testing: Checking that known issues from previous Measure Killer versions remain fixed.

Rapid response to new vulnerabilities

If a vulnerability is discovered, our team follows a quick release process to track the issue, apply a prompt fix, and release an updated version of Measure Killer. We typically release bug-fix or patch versions on a monthly basis.

These measures, combined with our architecture that avoids storing or processing actual data, are designed to protect sensitive information and maintain a high level of security for all Measure Killer users.

Known CVEs

V8

Regarding the sandbox escape vulnerabilities associated with V8: In Measure Killer, V8 is bundled with the Qt WebEngine library, but our application does not load any external web content or scripts. It only renders internal HTML charts (for example, in the “Plot results” tab). Because no untrusted content is ever processed, **these vulnerabilities cannot be exploited in Measure Killer.**

LLVM

This library is pulled in by PyInstaller when converting our Python code into an executable. Measure Killer does not compile or handle external code, and PyInstaller is only used during development to build the .exe (never at runtime). Thus, there is no way for an LLVM-Based attack in Measure Killer.

Python (3.11.8)

CVEs in Python:

- CVE-2024-8088 (CWE-835)
- CVE-2024-12254 (CWE-400)
- CVE-2017-20052 (CWE-427)
- CVE-2024-7592 and CVE-2024-6232 (CWE-1333 or CWE-400)
- CVE-2015-5652 (Broad CWE category)
- CVE-2025-0938 (CWE-20)
- CVE-2024-6923 (CWE-94)

- CVE-2024-3219 (CWE-306)

These vulnerabilities arise when an application can run or process arbitrary code or malicious inputs. **Measure Killer does not allow external code execution**, nor does it ingest untrusted data. We only pull metadata from Microsoft's Power BI Service (over HTTPS) and parse local files that the user explicitly opens.

libxml2 and Expat

CVEs associated with libxml2 and Expat (such as CVE-2016-4616, CVE-2016-4615, CVE-2016-4614, CVE-2021-3517, CVE-2024-25062, and more) that typically become relevant if an application processes malformed or hostile XML from unknown sources. In Measure Killer, we only parse .rdl files created by Power BI Report Builder, Microsoft Report Builder, or generated via Power BI's "Export Report In Group" API. **These .rdl files have a defined structure and come from trusted sources, making the known XML-related exploits not applicable.**

libvpx and libaom

These are part of Qt's multimedia modules. Measure Killer does not process or render video files, so potential vulnerabilities in these libraries (usually triggered by malformed video content) do not pose a risk in our application.

Qt (version 6.7.2)

Qt provides the main user interface framework. While Qt can load external HTML or JavaScript, Measure Killer does not permit loading unknown URLs or scripts. Similar to V8, we only use Qt WebEngine for charts generated internally by the tool.

Skia (version 118)

Skia is an image rendering engine used indirectly by Qt WebEngine. We do not load or display arbitrary images in Measure Killer; only predefined icons and resources stored locally. **With no untrusted image handling, Skia-related exploits are not a concern.**

SQLite3 (version 3.42.0)

Measure Killer uses SQLite to store activity logs of the Power BI Tenant, such as report views, and Excel activities. **We do not open or process any external .sqlite files, thereby avoiding typical exploitation paths for SQLite vulnerabilities.**