APPEON®

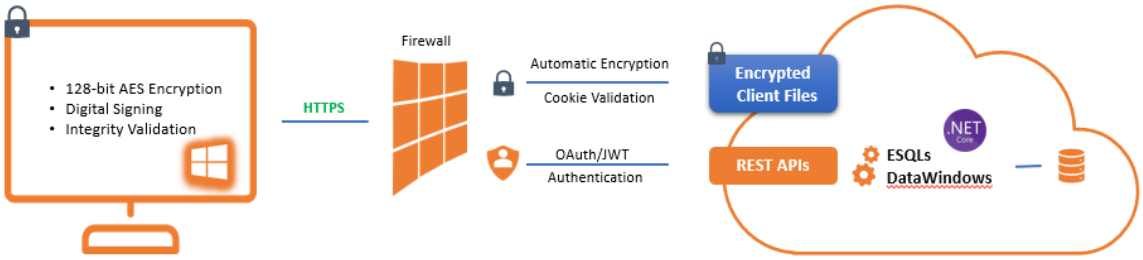
Technical Whitepaper:

POWERSERVER 2025 SECURITY MEASURES

Table of Contents

1 INTRODUCTION

Installable cloud apps deployed from PowerServer 2025 adopt standard cloud-native architecture running on the .NET framework and adhere to industry best practices for security. All SQLs and DataWindows execute behind the firewall. Data is accessed through industry-standard REST APIs, secured by OAuth or JWT. The client app is encrypted, digitally signed, and checked for integrity.



This document explains the detailed security measures that PowerServer 2025 has inherently implemented, or you can undertake, to protect the privacy, integrity, and availability of data and files transmitted for the running of installable cloud apps:

- Automatic Encryption
- Client App Security
- Transport Security
- Cookie Validation
- Token-Based Authentication
- Data Security

2 AUTOMATIC ENCRYPTION

Encryption is an effective way to reduce the probability of a security breach in the apps. PowerServer automatically encrypts the app files and sensitive information, and the encryption algorithm is uniformly AES 128-bit.

- **Encryption of account information in the server configuration**

  If you export a PowerServer project to a .srj file, the database connection information in the file is encrypted. Because web server profiles contain the FTP username and password, the file that stores the profiles is also encrypted.

- **Encryption of the compiled p-code files**

  When compiling a PowerServer project, every SRW, SRD, SRU, etc. file from the application PBLs is compiled into its individual corresponding p-code file (that have new file extensions, such as .dwo, .apl, .fun, .win, .udo) instead of a monolithic PBD file. The PowerServer project configuration settings provide the option “Encrypt all the compiled p-code files”. If the option is selected, the p-code files generated during app compilation are encrypted.

- **Encryption of the app manifest file**

  Each installable cloud app contains an app manifest file that lists every file contained in the installable cloud app and the file hash code. The manifest is encrypted.

3 CLIENT APP INTEGRITY

When a user loads and runs an installable cloud app, the files to be downloaded to the client include a supporting program (Cloud App Launcher) and supporting runtime files (relevant PowerBuilder Runtime files). The supporting program and files can be jointly used by multiple installable cloud apps; and the app executable file, P-code files compiled and granularized from the source code, resource files, OCX files, and other external files used by the app.

It is important to ensure that the client app only uses the files from the original deployment. Therefore, PowerServer is designed with the following mechanisms to enforce the client app integrity.

- **Integrity assurance of the app files**

  Each installable cloud app contains a file manifest that lists every file contained in the installable cloud app and the hash code of each file.

  The PowerServer project configuration settings provide the option “Validate the application integrity before the app runs”. Checking this option will check that the app files to be executed on the client have not been tampered with outside of the PowerServer compile process.

- **Digital signing of the app executable**

  The app executable file (appname.exe) can be digitally signed, including support for Extended Validation (EV) code-signing certificates. A valid digital signature—particularly one issued by a trusted Certificate Authority—helps ensure the authenticity and integrity of the executable file, giving users greater confidence that the app is legitimate and has not been tampered with.

- **Verification by the Cloud App Launcher**

  The Cloud App Launcher incorporates the following verification mechanisms to ensure a secure and trustworthy client app download and upgrade process:

  - **Trusted application validation**

    The Cloud App Launcher only accepts applications that are flagged with a trusted identifier, helping ensure authenticity and prevent unauthorized use.

  - **URL whitelist enforcement**

    Upon launch, the Cloud App Launcher reads a configuration file (apprun.json) to validate a list of approved download URLs. Only files hosted at these trusted locations are permitted for download or execution.

  - **Digital signing of the Cloud App Launcher executab**
    The Cloud App Launcher is signed with an Appeon Extended Validation (EV)

code-signing certificate. EV certificates require rigorous identity verification by a trusted Certificate Authority (CA), providing the highest level of publisher authenticity.

If you choose to deploy a custom Cloud App Launcher, be sure to sign it with your own code-signing certificate—ideally one issued by a reputable CA.

4 TRANSPORT SECURITY

4.1 Securing communication with TLS and HTTPS

Installable cloud apps communicate with two main components:

- The web server, to download application files and runtime resources via HTTP requests
- The PowerServer Web APIs, to access services and exchange data

To protect these communications, it is strongly recommended to enable TLS 1.2 or TLS 1.3 for both channels:

- File downloads between the client and the web server
- REST API calls between the client and the .NET server

Enabling HTTPS is essential for your app's security. It ensures that all data in transit is encrypted, which protects against interception, tampering, or eavesdropping. Once HTTPS is configured:

- All data—including full URLs, cookies, request/response bodies (binary or text), session IDs, tokens, and headers—is encrypted during transmission.
- Sensitive information cannot be viewed or manipulated by third parties during transfer.

Once your server is configured to use HTTPS, both the Cloud App Launcher's download process and the client app's runtime calls to the PowerServer Web API perform strict server certificate validation by default. This ensures that every TLS handshake verifies the server's identity against a trusted root and confirms that the certificate chain is unbroken and unrevoked. By enforcing robust TLS 1.2/1.3 settings and rejecting any certificate errors, you effectively eliminate the risk of man-in-the-middle attacks, data interception, or unauthorized content injection.

4.2 Bypassing certificate validation (for testing only)

In certain controlled, non-production scenarios—such as development environment or testing environment that uses self-signed server certificate—you may temporarily enable the “Ignore Server Certificate” option to bypass validation errors. However, this exception must be scoped strictly to your test environment. Always verify that production builds continue to enforce full certificate validation to maintain end-user trust and compliance with industry best practices.

5 COOKIE VALIDATION

It is possible to add a cookie validation script in the app folder of an installable cloud app at the web server. When a client requests to run the app, the web server gets and validates the cookie from the client against the script, and determines whether to allow the client to download the required files (CloudAppLauncher, the app runtime files etc.) for running the app. The cookie validation is performed at the start of every client session. In addition, with the GetHttpRequestHeader and GetHttpRequestHeaders functions of the Application object, the HTTPClient object in the app can get the cookie from the HTTP request header, and then validate the cookie if necessary.

6 TOKEN-BASED AUTHENTICATION

6.1 Using the traditional authentication method for development only

Communication between the client app and the .NET server is through standard REST APIs. If you select the “Do not use auth service” option in the PowerServer project settings, the generated REST APIs do not perform any authentication and can be readily accessed. During development, it should not be a big concern, but when going into production, especially if the REST APIs are exposed over the Internet, then it is highly recommended to implement some token-based authentication for the REST APIs.

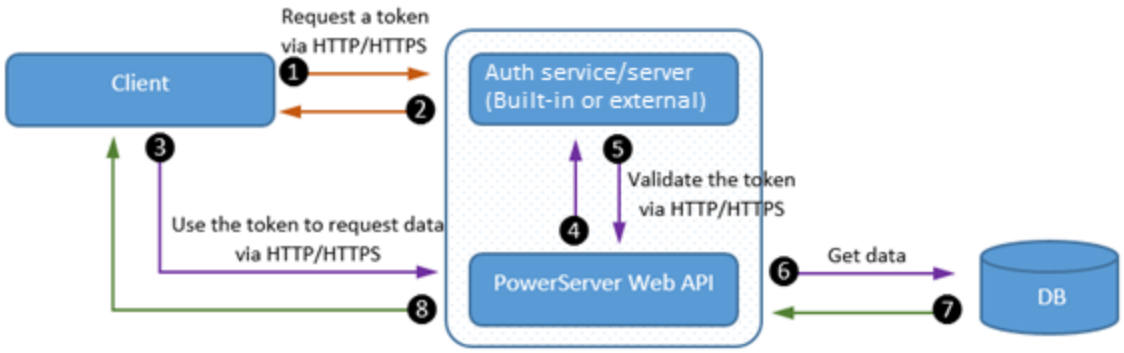
6.2 Implementing token-based authentication for production

Token-based authentication helps to mitigate REST API security risks as follows:

- An app user's access to any resource will be denied unless he/she is granted a token.
- An app user should only have the required set of permissions to perform actions on PowerServer's REST APIs for which they are authorized, and no more.

Below is a sample workflow of token-based authentication. As illustrated in the workflow (Figure 2), the client sends the user credentials to the authentication service/server (built-in or external). The authentication service/server validates the user, and if validation is successful, it authorizes and returns a token to the client. Then, the client will send all the subsequent requests with the token, and the PowerServer's REST APIs will validate the token in the request with the authentication service/server before handing the request.

Figure 2



There are various approaches to configuring an authentication service/server to work with PowerServer's REST APIs. Which one to select would depend on your security requirements and the existing security infrastructure in your company.

6.2.1 Approach 1: Use a built-in auth template to set up built-in authentication

The steps are simple and quick if you decide to implement token-based authentication by utilizing one of the three built-in auth template options in the PowerServer project settings:

- Use built-in JWT server
- Use built-in AWS Cognito server

With this built-in approach, the auth service/server is hosted in the PowerServer solution, and the users and their credentials can be either stored in a .cs file in the solution or the designated authentication database. Follow the instructions in the respective document to implement the built-in authentication for the .NET server: Using JWT, and Using Amazon Cognito.

6.2.2 Approach 2: Use an authentication server that already exists

PowerServer 2022 R3 provides templates that can be easily extended to support an existing authentication server that works with the OAuth 2.0 flows or JWT, such as Azure AD, Azure AD B2C, Visual Guard, and Okta OIDC (OpenID Connect). If you have already implemented such an authentication server, you can select the "Use external Azure Active Directory service" (for Azure AD and Azure AD B2C) or "Use external auth service" (for Visual Guard and Okta OIDC) option in the PowerServer project settings, and follow the instructions in this document to implement it for the .NET server: Using external Active Directory service; and Using other authentication servers.

6.2.3 Approach 3: Set up a new authentication server

Sooner or later you may want to set up your own authentication server so that you can share it among multiple projects that may be developed with other tools and languages. The requirement for the new authentication server is that it must comply with the OAuth 2.0 flows or JWT. There is a long list of notable OAuth providers, select whichever one you like, and then follow the relevant documentation to set up the server.

To have the new authentication server working with a PowerServer project, please select the "Use external auth service" option in the PowerServer project settings, and follow the instructions in this document: Using other authentication servers.

6.3 Implementing authorization for management APIs and other custom extensions

PowerServer Management APIs are disabled by default for security reasons and must be explicitly enabled via app.UsePowerServerManagementAPI(). Once Management APIs

are enabled—or if you extend or expose any custom Web API endpoints—you must enforce strict authorization policies. We recommend defining role- or claim-based requirements in your AuthenticationExtensions.cs (for example, requiring an Admin role for management API access) and applying equivalent policies to all custom APIs.

By combining token-based authentication with rigorous authorization checks, you ensure that only properly privileged users can view or modify sensitive resources, greatly strengthening your security posture.

7 DATA SECURITY

Data security is a core focus of cloud app security. PowerServer has taken a number of measures to enhance the data security in installable cloud apps, assuming that you follow the best practices recommended in this section.

7.1 Hosting the .NET server and database in the same LAN

It is important to publish the PowerServer's REST APIs to a server that resides in the same LAN as the database server. This way, the data operations of the app will be executed in the LAN behind the firewall, which reduces the vulnerability to cyber-attacks.

Specifically speaking, in installable cloud apps,

- Static DataWindows/DataStores are all .NET models and hosted on the .NET server;
- Embedded SQLs are all SQL strings hosted on the .NET server.
- Dynamic DataWindows/DataStores will be created at the client and have its SQL sent as strings, encrypted, to the .NET server during runtime.
- Dynamic SQLs will be created at the client and sent as strings, encrypted, to the .NET server during runtime.

When the client app performs a data request, it calls the corresponding REST API, and then the REST API is executed by the .NET server using the corresponding .NET model or SQL strings. After that, the .NET server returns the result set in JSON format to the client.

During the process, the data accessing and handling functions are all restricted, and under control, on the server side (.NET server and database server).

7.2 Using a firewall

Firewalls have long been the first line of defense in network security. In installable cloud apps, data is exchanged between the client, and the servers (the .NET server and the database server), so it is important to implement a firewall solution for the apps that adhere to best practices, such as configuring a whitelist for web intrusion prevention.

7.3 Setting up database connections using caches

An inherent security feature of installable cloud apps is that you no longer need to keep the database connection information (including user ID, password, DB connection string, etc.) on the client side. To implement a database connection, you configure the connection information in the PowerServer REST API project (that resides on the server) and then map each transaction object from the client app to a server cache. This way, the cache settings are deployed and stored in the server APIs, and the connection information will be only used by the .NET server for the database connection.

7.4 Session and transaction management

Session and transaction timeouts can be easily applied to installable cloud apps by simply configuring a setting in the PowerServer project configuration. This feature helps safeguard the application from unauthorized access when authorized users have stepped away momentarily or forgotten to log out from the system.

7.5 Message-level integrity (HMAC)

PowerServer offers an application-level client message authentication feature based on HMAC (Hash-based Message Authentication Code). When enabled, the client generates a unique authentication code for each request using the request data and a shared secret key. This code helps ensure the integrity of the message — if the data is tampered with during transmission, the code will no longer match.

On the server side, PowerServer verifies the HMAC to ensure that the request has not been altered during transmission. While HTTPS encrypts data in transit, it does not guarantee that the client itself is trustworthy — for example, when running on

unmanaged or potentially compromised devices. In such cases, HMAC adds a critical layer of protection by ensuring that only properly signed requests are accepted, effectively detecting and rejecting any unauthorized modifications — even from compromised clients.

About Appeon

Appeon helps software developers build faster, better, business apps. PowerBuilder, the flagship product of the company, has been used by over 18,000 organizations worldwide, and has a long history of being relied on for mission-critical systems. Appeon's products simplify and accelerate the development of data-rich cloud apps for Windows.

For more information visit www.appeon.com, or follow Appeon on social media: Facebook, YouTube, Twitter, LinkedIn.

For customers interested in learning more about Appeon products:
Global Customer Service Center: +1-877-3APPEON (+1-877-327-7366), sales@appeon.com