June 14, 2023

Public Preview : Improve Win32 app security via app isolation

By
David Weston, Vice President Enterprise and OS Security at Microsoft
Sumit Lahiri, Principal PM Microsoft Edge and Platform Security at Microsoft

We are thrilled to announce the public preview launch of Win32 app isolation. This blog post provides an overview of the topic. To learn more about the developer experience and engage with the team, please visit our GitHub page.

It is worth noting that Win32 app isolation is an addition to the family of existing Windows sandbox options, such as Windows Sandbox and Microsoft Defender Application Guard. While these options are based on virtualization based security, Win32 app isolation is built on the foundation of AppContainers (and more). AppContainers are specifically designed to encapsulate and restrict the execution of processes, helping to ensure they operate with limited privileges, commonly referred to as low integrity levels.

Win32 App isolation

The frequency and impact of zero-day vulnerabilities have witnessed a substantial increase over the years. Attackers frequently focus their attention on popular applications, exploiting either unknown or unpatched vulnerabilities. That's why we strongly advocate for the integration of preventive and containment measures.

To this end, we propose using the combined power of Win32 app isolation and cutting-edge technologies like Smart App Control, which together work to effectively block untrusted applications and limit damage if trusted apps are compromised. By implementing this approach, a robust security strategy can be established, significantly mitigating the potential harm caused by zero-day attacks.

Win32 app isolation is a new security feature designed to be the default isolation standard on Windows clients. It is built on AppContainers and offers several added security features to help windows platform defend against attacks that leverage vulnerabilities in the application (this could be 3P libraries as well). To isolate their apps, application developers can update their applications using the tools provided by Microsoft. For more information on Win32 app Isolation developer experience, please visit our GitHub page.

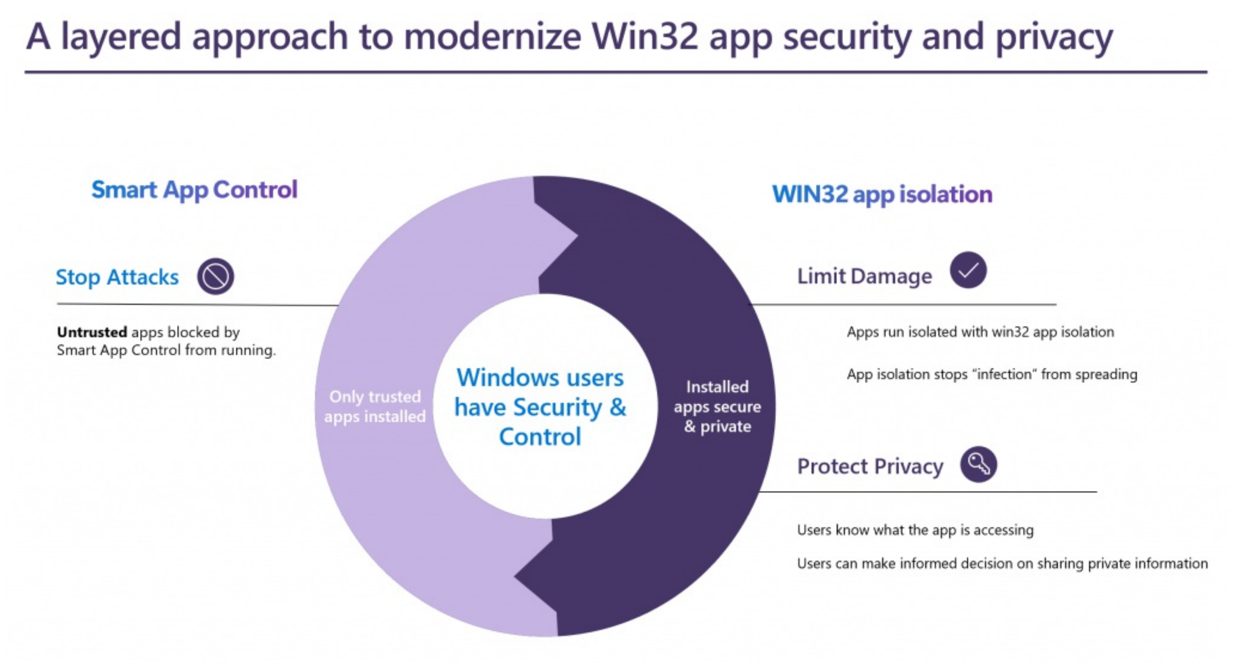
A layered approach to modernize Win32 app security and privacy
Smart App Control
Stop Attacks
Untrusted apps blocked by Smart App Control from running.
Only trusted apps installed
Windows users have Security & Control
Installed apps secure & private
WIN32 app isolation
Limit Damage
Apps run isolated with win32 app isolation
App isolation stops "infection" from spreading
Protect Privacy
Users know what the app is accessing
Users can make informed decision on sharing private information

Figure 1: Combining Preventive and Containment Strategies.

Another benefit of isolation is to safeguard end-user privacy choices in the event of a breach. When a Win32 app runs with the same privilege as the user, it is possible to allow itself access to user's information without the user's consent. Consequently, there is a risk of unauthorized access to the user's privacy data by malicious actors without their knowledge or consent.

Goals of Win32 app isolation

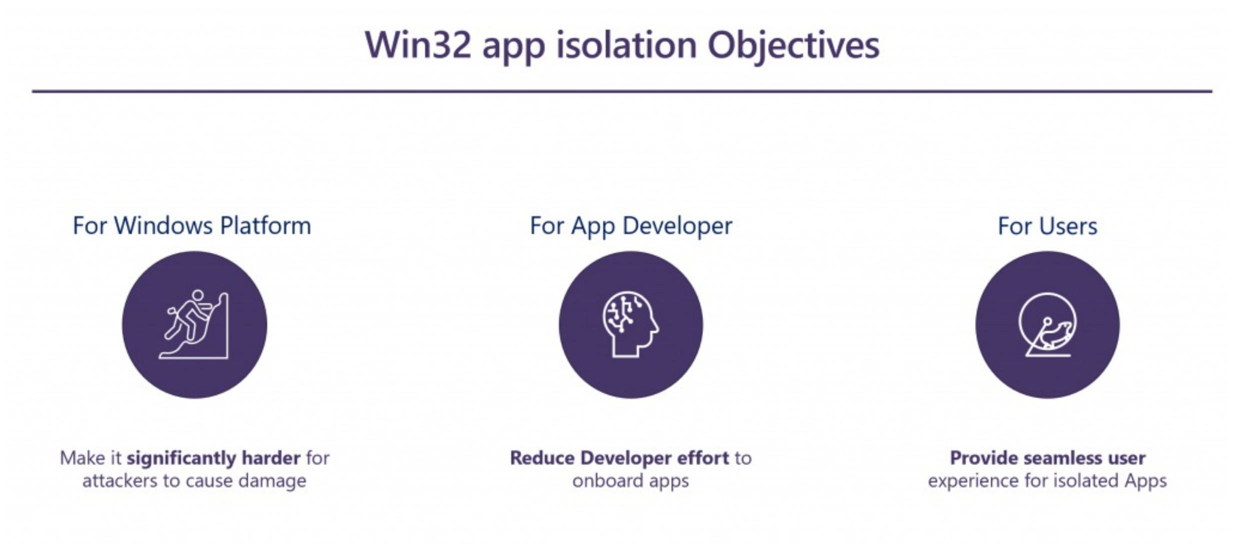
Win32 app isolation Objectives
For Windows Platform
Make it significantly harder for attackers to cause damage
For App Developer
Reduce Developer effort to onboard apps
For Users
Provide seamless user experience for isolated Apps

Figure 2: Key objectives of Win32 app isolation

Limit damage: Win32 app isolation achieves its goal of limiting impact (in the event apps are compromised) by running apps with low privilege, which requires a multi-step attack to break out of the container. Attackers must target a specific capability or vulnerability, compared to having broad access and since the attack must be directed at a specific vulnerability, mitigation patches can be quickly applied, reducing the shelf life of the attack.

Reduce developer effort: To reduce the effort required for developers to update their apps, Microsoft provides developer tools and experiences, with a focus on the MSIX packaging tool and the Application Capability Profiler.

Seamless user experience: Finally, while the focus is on security, it is critical that security decisions are not delegated to end-users via cryptic security prompts, and application compatibility is maintained.

Limit damage

The protection offered by Win32 App isolation follows a two-step process. In the first step, the Win32 application is launched as a low integrity process using AppContainer, which is recognized as a security boundary by Microsoft. Consequently, the process is limited to a specific set of Windows APIs by default and is unable to inject code into any process operating at a higher integrity level.

In the second step, the least privilege is enforced by granting authorized access to Windows securable objects. This access is determined by capabilities that are added to the application manifest through MSIX packaging. Securable objects in this context refer to Windows resources whose access is safeguarded by capabilities. These capabilities serve as a means to implement a Discretionary Access Control List on Windows.

Reduce developer effort

To help ensure that isolated applications run smoothly, developers must define the access requirements for the application via access capability declarations in the application package manifest.

The Application Capability Profiler **(ACP)** simplifies the entire process by allowing the application to run in "learn-mode" with low privileges. Instead of failing access if the capability is not present, ACP allows access **and logs additional capabilities required for access if the application were to run isolated**.

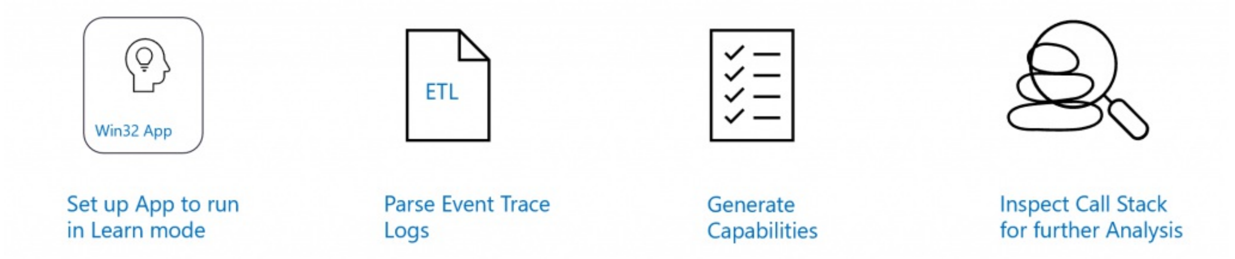
Win32 App
ETL
Set up App to run in Learn mode
Parse Event Trace Logs
Generate Capabilities
Inspect Call Stack for further Analysis

Figure 3: Overview of Application Capability Profiler

Under the hood, **ACP** uses the Windows Performance Analyzer data layer backend (WPA) and parses Event Trace Logs (**ETL**) to provide a list of additional capabilities needed. ETLs are detailed and verbose, and ACP parses them to output missing capabilities as a file. Once the capabilities are output, they can simply be included in the application package manifest.

Finally, ACP provides a WPA profile file "*ACP-StackTrace.wpaProfile*" that allows the user to easily configure WPA to inspect the event trace logs captured (in learn-mode) for relevant call stack information. In most cases, call stack analysis using WPA is not required.

For more information on **ACP**, please refer to the Github documentation page, linked here.

Seamless user experience

To create a smooth user experience that aligns with non-isolated/native Win32 applications, two key factors should be taken into consideration, as outlined below. The first factor relates to implementing methods to manage access to files and privacy information within and outside the isolation boundary (AppContainer). The second factor involves integrating Win32 apps with other Windows interfaces in a way that helps enable seamless functionality without causing perplexing user consent prompts. Now, let's explore these factors in greater detail:

1. Approaches for accessing data and privacy information;
2. Integrating Win32 Apps for compatibility with other Windows interfaces;

Approaches for accessing data and privacy information.

Access to a user's private data, such as camera, microphone, location, images, files, or folders, is not permitted without the user's permission. However, the app may require access to certain program files, such as .NET libraries or protected registry keys. Asking for permission to access user folders is intuitive, but asking for permission to access program files and registry keys can be confusing for users. This confusion can be exploited by a compromised Win32 app to trick the user into allowing access to a registry key or sensitive file, which can then be used to escape the sandbox.

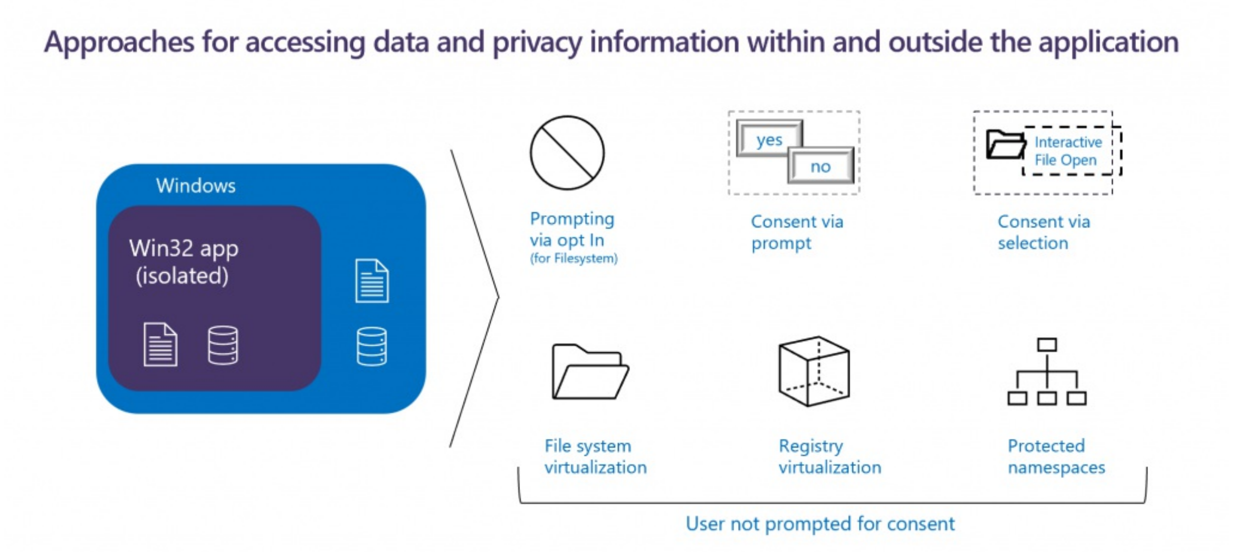
Approaches for accessing data and privacy information within and outside the application
Windows
Win32 app (isolated)
Prompting via opt In (for Filesystem)
yes
no
Consent via prompt
Interactive File Open
Consent via selection
File system virtualization
Registry virtualization
Protected namespaces
User not prompted for consent

Figure 4: Effectively managing user consents.

To help prevent unauthorized access, several instrumentations have been built. Win32 apps need to explicitly include the "*isolatedWin32-promptForAccess*" capability and declare their intent to support prompting. Apps that do not require access to user's data outside the app can opt out of any kind of user prompting for consent. Note, this capability only allows control over filesystem access, access to privacy related information such as camera, location and microphone shall always prompt.

Let us now discuss approaches on how user consents are obtained. User prompts are displayed when user consent is required, and the context of the prompt should be meaningful to the end user, such as accessing the user's documents or camera. The other way to grant consent to files is via user selection such as when the user selects a file via the file dialog or by right clicking the context menu.

When the user grants consent to a specific file for the isolated application, the isolated application interfaces with Windows **Brokering File System** (BFS) and grants access to the files via a mini filter driver. BFS simply opens the file and serves as the interface between the isolated application and BFS.

File and registry virtualization helps ensure that apps continue to work while not updating the base file or registry. This also minimizes any user experience friction while maintaining application compatibility. Protected namespaces are created to allow access only to the app and do not require user consent. For example, access to a folder that has a property only known to the Win32 app and required for app compatibility can be granted.

Integrations with Win32 app isolation for compatibility

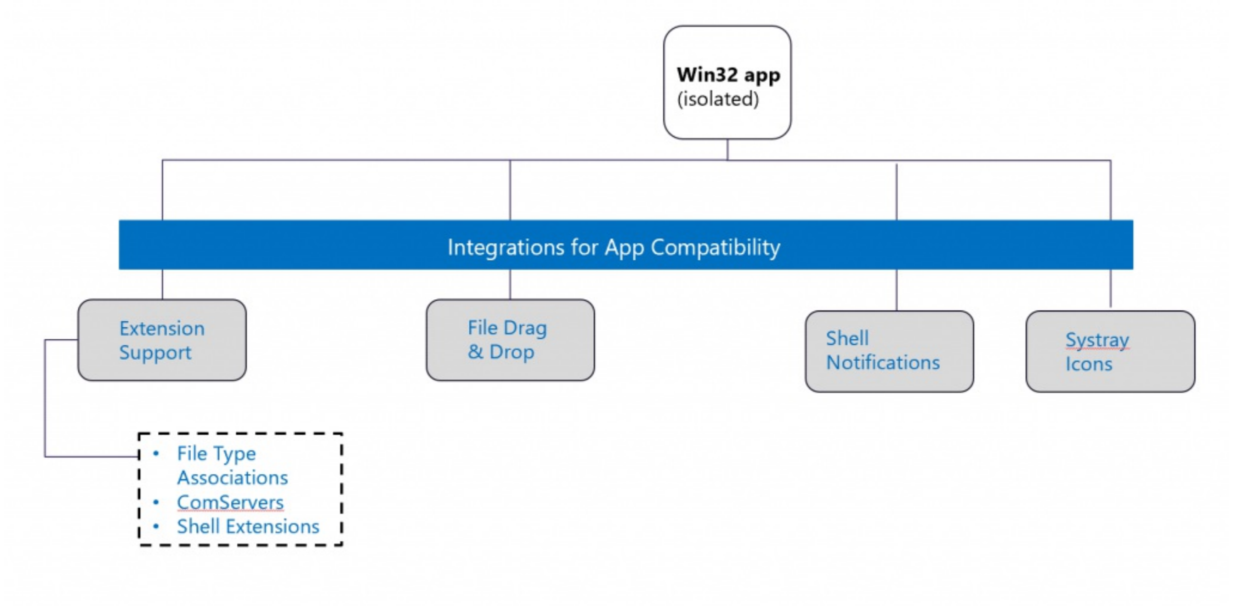

Figure 5: Maintaining app compatibility with isolation.

In order to achieve a high level of similarity and feature parity between isolated and non-isolated Win32 applications, certain allowances have been made. Specifically, Win32 apps operating with low privileges (referred to as low integrity level) are permitted to interact with file systems and various APIs. For example, interactions involving File Type Associations, including the "open with" option, COM servers, and file drag and drop, are facilitated through BFS (File System Broker).

Furthermore, capabilities included in the application manifest enable interactions with other Windows components, such as shell notifications and system tray icons. This ensures that these functionalities can be utilized without compromising the security of these applications.

What's Next!

To summarize, Win32 app isolation enhances security for Windows clients by using AppContainer and additional security controls (as described above) to help reduce the risk of damage from compromised applications and help safeguard user privacy. The approach enforces least privilege through added capabilities and employs various strategies to help prevent unauthorized access, while minimizing developer effort and maintaining application compatibility.

To learn more about isolating your existing or new Win32 Apps, visit the GitHub page at: microsoft/win32-app-isolation (github.com). Win32 App Isolation is currently available for public preview. We look forward to your participation!

Tags:

Feature Documentation

What's new

Surface Laptop Studio 2

Surface Laptop Go 3

Surface Pro 9

Surface Laptop 5

Microsoft Copilot

Copilot in Windows

Explore Microsoft products

Windows 11 apps

Microsoft Store

Account profile

Download Center

Microsoft Store support

Returns

Order tracking

Certified Refurbished

Microsoft Store Promise

Flexible Payments

Education

Microsoft in education

Devices for education

Microsoft Teams for Education

Microsoft 365 Education

How to buy for your school

Educator training and development

Deals for students and parents

Azure for students

Business

Microsoft Cloud

Microsoft Security

Dynamics 365

Microsoft 365

Microsoft Power Platform

Microsoft Teams

Copilot for Microsoft 365

Small Business

Developer & IT

Azure

Developer Center

Documentation

Microsoft Learn

Microsoft Tech Community

Azure Marketplace

AppSource

Visual Studio

Company

Careers

About Microsoft

Company news

Privacy at Microsoft

Investors

Diversity and inclusion

Accessibility

Sustainability

Your Privacy Choices

Consumer Health Privacy

Sitemap Contact Microsoft Privacy Terms of use Trademarks Safety & eco Recycling About our ads © Microsoft 2024