ChatGPT

Enterprise Software Packaging and Deployment Assessment Questions

Below is a set of diverse assessment questions (multiple-choice, scenario-based, short-answer, etc.) on enterprise application packaging and deployment topics, along with answers and explanations. Each answer is supported by referenced material.

1. Application Packaging Fundamentals

1. **Which of the following is a primary benefit of enterprise application packaging?**
   A. Allows arbitrary user installations without testing.
   B. Ensures a consistent, stable environment across systems.
   C. Eliminates the need for any application updates.
   D. Automatically upgrades hardware drivers.
   Answer: B. Ensures a consistent, stable environment across systems. Explanation: One of the main benefits of application packaging is it "ensures a consistent, stable and reliable standard environment" by providing uniform installations [1]. It also streamlines software management, mitigates security issues, and reduces support costs. Options A, C, and D are not correct; packaging requires testing, still needs updates, and deals with software (not directly hardware drivers).

2. **True or False: Application packaging decreases the risk of business disruptions.**
   Answer: True. Explanation: The packaging process involves thorough testing and standardized deployments, which "decreases risks for business disruption" [1]. By identifying issues before rollout, packaging helps avoid unexpected failures in production.

3. **What tool converts classic Windows installers into the `.intunewin` format for Intune?**
   Answer: The Microsoft Win32 Content Prep Tool (IntuneWinAppUtil). Explanation: Before uploading a Win32 app to Intune, you use the Microsoft Win32 Content Prep Tool to preprocess the installer into a `.intunewin` file [2]. This tool packages all setup files and detection logic into the required format.

4. **Given an enterprise MSI package that must apply registry settings for each user at first logon, what approach can be used?**
   Answer: Use Active Setup in the package to write to the machine (HKLM) registry, so that Windows will execute those settings for each new user. Explanation: Active Setup allows an administrator to write registry values under HKLM that will then run once per user at logon. For example, writing to HKLM\Software\Microsoft\Active Setup\Installed Components with a "StubPath" causes that command to run in each user's profile [3] [4].

5. **Describe the difference between per-user and all-users registration of a VSTO Office add-in in the registry.**
   Answer: A per-user VSTO add-in is registered under HKEY_CURRENT_USER, whereas an all-users

VSTO add-in is registered under HKEY_LOCAL_MACHINE. Explanation: VSTO add-ins can be registered for the current user (registry entries under HKCU) or all users (entries under HKLM) [5]. For example, HKLM is used when deploying via MSI to make the add-in available to everyone on the computer, while HKCU is used when it's installed for just the current user.

2. MSI Driver Handling and SetupAPI Logs

1. **During Windows deployment, an MSI package includes a device driver. The driver installation fails during the Out-Of-Box Experience (OOBE) phase. Which log file should you inspect for driver installation failures?**
   Answer: `C:\Windows\inf\setupapi.dev.log`. Explanation: The file `setupapi.dev.log` contains logs of driver failures occurring during the OOBE phase (user's first logon phase) of setup [6]. (If it were during offline image specialization, you'd check `setupapi.offline.log` [7].)

2. **Which log file captures driver installation issues during the offline (WinPE/specialize) phase of Windows setup?**
   Answer: `C:\Windows\inf\setupapi.offline.log`. Explanation: The `setupapi.offline.log` records driver failures during the component specialization sub-phase of setup (the offline or WinPE phase) [7].

3. **A Task Sequence in SCCM fails to install a specific device driver. Which system log can you use to see which driver match or why installation failed?**
   Answer: Use `setupapi.dev.log` on the target system. Explanation: The `setupapi.dev.log` (under C:\Windows\INF) logs the matching process of device IDs to driver INF files during device installation [8]. You can search that log for the device's Hardware ID to troubleshoot driver selection or failures.

4. **True or False: Windows will automatically use the vendor's driver signature during an MSI installation regardless of the driver's certificate trust status.**
   Answer: False. Explanation: If a driver is unsigned or its certificate is not trusted, installation can fail. Tools like `sigverif` or examining `setupapi.dev.log` would show signature issues, and one might need to disable driver signature enforcement or install the certificate in the Trusted Publishers store.

3. COM and Office Add-in Load Behavior

1. **Which LoadBehavior value for a VSTO COM add-in causes it to load at startup by default?**
   A. 0
   B. 1
   C. 2
   D. 3
   Answer: D. 3. Explanation: For VSTO Office add-ins, the `LoadBehavior` registry entry under the Office\Addins key is typically set to 3 by default, which means "load at startup" [9].

2. **If a user disables a VSTO add-in in Office, which registry hive holds the modified LoadBehavior value that will take effect for that user?**

Answer: HKEY_CURRENT_USER. Explanation: Disabling an add-in changes the LoadBehavior value in HKCU for that user, which overrides the HKLM value. The documentation notes that "LoadBehavior value in the HKEY_CURRENT_USER hive overrides the default LoadBehavior in HKEY_LOCAL_MACHINE" [10].

3. **A Word COM add-in fails to load on startup. You find its registry LoadBehavior value is 0 under HKCU. What does a LoadBehavior of 0 mean for COM add-ins?**
   Answer: It means the add-in will not load (Disconnected). Explanation: According to Microsoft documentation, for COM add-ins the LoadBehavior of 0 = Disconnect (the add-in is not loaded) [11]. Setting it to 2 (Bootload) would make it load at startup.

4. **In which registry location would you typically find the LoadBehavior setting for an Excel or Word COM add-in?**
   Answer: Under `HKCU\SOFTWARE\Microsoft\Office\<ApplicationName>\Addins\<AddInName>\LoadBehavior`. Explanation: COM add-ins register under the Office application's Addins key. For example, a Word add-in would be under `HKCU\SOFTWARE\Microsoft\Office\Word\<ProgID>` with a `LoadBehavior` DWORD [12].

5. **Scenario:** A COM add-in you installed via an MSI is not loading for users. You check and see `LoadBehavior` is set to 1 under HKLM for the add-in. What might you do?
   Answer: Ensure `LoadBehavior` is set to a value that loads at startup (such as 3 for VSTO add-ins or 2 for standard COM add-ins) and verify the add-in's manifest/registry under both HKLM (for machine-wide) and HKCU (per-user). Explanation: For VSTO add-ins the default should be 3 (load at startup) [9]. For standard COM add-ins, 2 is "Bootload" (load at startup) [11]. If users still have issues, also check the HKCU hive in case a user-disabled setting is overriding the machine setting [10].

4. Active Setup, Logon/Startup Scripts, and Context

1. **What is Active Setup in Windows?**
   Answer: A mechanism that executes one-time-per-user commands at user logon. Explanation: Active Setup runs commands once per user when they first log on to the system, often used to initialize user-specific settings. It checks for components listed in `HKLM\Software\Microsoft\Active Setup\Installed Components` and runs their `StubPath` commands if not already run for that user [13] [3].

2. **At what point during the logon process is Active Setup executed?**
   Answer: Before the desktop appears and before any Run or RunOnce entries. Explanation: Active Setup runs early during user logon, blocking until it finishes. It executes before any Run or RunOnce registry entries are processed [14].

3. **How can Active Setup be triggered manually during a session?**
   Answer: By running `%windir%\system32\runonce.exe /AlternateShellStartup`. Explanation: Helge Klein notes that you can manually trigger Active Setup by executing this command in a session [15].

4. **Which registry key stores the machine-wide components list for Active Setup?**
   Answer: `HKLM\SOFTWARE\Microsoft\Active Setup\Installed Components`. Explanation: Active Setup components are defined under this key (machine part) and mirrored in each user's `HKCU\Software\Microsoft\Active Setup\Installed Components` (user part) after running [3].

5. **Within an Active Setup component key, what is the purpose of the `StubPath` value?**
   Answer: It is the command line that Active Setup will execute if the component needs to run for a user. Explanation: The `StubPath` registry value contains the command (e.g., an executable or script) to run for that component when Active Setup determines it should run [4].

6. **True or False: A startup script in Group Policy runs under the Local System account, while a user logon script runs under the logged-on user's account.**
   Answer: True. Explanation: By design, a computer startup script runs in the Local System context, whereas a logon script runs with the current user's credentials [16].

7. **Scenario:** You need to deploy a registry fix that writes to each user's HKCU registry. Your deployment tool runs scripts as SYSTEM. Which method can ensure each user gets the fix?
   Answer: Use Active Setup – write the fix in an Active Setup `StubPath` under HKLM so that when each user logs on, Active Setup runs and applies the fix. Explanation: As one expert noted, you "write to Local machine part but Microsoft Active Setup handles the rest," meaning Active Setup will propagate the change to each user's hive [17].

8. **Which runs first at logon: an Active Setup command or a RunOnce registry entry?**
   Answer: Active Setup runs first. Explanation: Active Setup processing happens before any Run or RunOnce commands are evaluated during logon [14].

5. Scheduled Tasks and Reboot Handling

1. **Which registry location can be used to schedule a one-time execution of a command at the next user logon?**
   Answer: `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce`. Explanation: Adding an entry under the `RunOnce` key causes the specified command to run once at the next user logon [18].

2. **What is the difference between the `Run` and `RunOnce` registry keys?**
   Answer: Entries under `Run` execute at every logon, while `RunOnce` entries execute only once per logon session. Explanation: The Advanced Installer guide explains that commands in `Run` always run on logon, whereas `RunOnce` commands run only a single time per user [19].

3. **How can you configure a task that runs an installer at user logon with highest privileges (even if the user has no admin rights)?**
   Answer: Create a Scheduled Task with a trigger of "At logon" and set the task to run as `NT AUTHORITY\SYSTEM` with highest privileges. Explanation: Task Scheduler allows specifying a user account for the task; by using the `NT AUTHORITY\SYSTEM` account (a local system account with full privileges), the installer can run elevated without prompting. This account has the highest system

privileges [20]. You can create such a task via `schtasks.exe` or PowerShell (e.g. `Register-ScheduledTask -TaskName "Name" -Trigger (New-ScheduledTaskTrigger -AtLogon) -User "SYSTEM" -Action (New-ScheduledTaskAction -Execute "msiexec.exe" -Argument "/i your.msi /qn") -RunLevel Highest`).

4. **True or False: Using `schtasks.exe /create /RL HIGHEST /RU "SYSTEM"` creates a task that ignores UAC.**
   Answer: True. Explanation: The `/RU "NT AUTHORITY\SYSTEM"` and `/RL HIGHEST` options tell the task to run under the Local System account with the highest run level, bypassing user context and UAC restrictions [21].

5. **Scenario:** You must perform an application update that requires deleting an in-use folder. The update includes uninstalling the old version, which must occur after a reboot. How can you automate the continuation of installation after reboot?
   Answer: Use a RunOnce key or scheduled task. Explanation: One approach is to place the remaining installation command in the `HKLM\...\RunOnce` registry key, so it executes at the next logon [22]. Alternatively, you can create a scheduled task (triggered at logon) under SYSTEM to run the installer after reboot. Both methods resume installation without manual intervention.

6. **Which PowerShell-based deployment framework is suggested for scripting complex installation sequences (e.g. creating scheduled tasks before reboot)?**
   Answer: PowerShell App Deployment Toolkit (PSADT). Explanation: The Advanced Installer article suggests using a wrapper such as PSAppDeployToolkit to handle advanced scenarios like scheduling tasks and reboot sequencing [23].

6. PowerShell App Deployment Toolkit (PSADT)

1. **Which PSADT function should you use to launch a standard executable (e.g. setup.exe) during a deployment script?**
   Answer: `Start-ADTProcess`. Explanation: `Start-ADTProcess` is the PSADT function designed to execute a typical setup executable with standard logging and control options [24].

2. **If an installer spawns child processes and the parent process exits early, which PSADT parameter ensures the script waits for all related processes to finish?**
   Answer: `-WaitForChildProcesses`. Explanation: This parameter makes PSADT wait for all child processes to complete before proceeding, useful when an installer launches subprocesses [25].

3. **You have a per-user application to install after a system context install. Which PSADT function allows running a process in the context of the logged-in user?**
   Answer: `Start-ADTProcessAsUser`. Explanation: `Start-ADTProcessAsUser` runs a process under the current user account, even when the script itself is running in system context. It is intended for cases where a user-context installation step is needed from a system-run script [26].

4. **How can you hide command-line arguments from appearing in the deployment log when using PSADT?**
   Answer: Use the `-SecureArgumentList` parameter. Explanation: `-SecureArgumentList`

instructs PSADT to hide the argument string in logs, which is useful for concealing sensitive information or long parameters [27].

5. **Which PSADT function is tailored specifically for executing MSI package actions (install/uninstall/repair)?**
   Answer: `Start-ADTMsiProcess`. Explanation: This function wraps `Start-ADTProcess` with additional MSI-specific parameters (like `-Action` install/uninstall) and is recommended for handling MSI packages within PSADT scripts [28].

6. **Scenario:** A PSADT script uses `Start-ADTProcess -FilePath 'installer.exe' -WindowStyle Hidden -NoWait`. What does this do?
   Answer: It runs `installer.exe` with no visible window and does not wait for its completion. Explanation: `-WindowStyle Hidden` runs the process without showing a window, and `-NoWait` tells the script to continue without waiting for the process to exit [29]. This combination is useful for launching background tasks that should not block the script.

7. Intune Win32 Packaging and Deployment

1. **Which file format is created by the Intune Win32 Content Prep Tool?**
   Answer: `.intunewin`. Explanation: The Win32 Content Prep Tool zips the application files and installer into a single `.intunewin` package that Intune can distribute [30].

2. **What is the maximum supported file size for a Win32 app in Intune?**
   Answer: 30 GB. Explanation: According to Microsoft documentation, the maximum size for a packaged Win32 application in Intune is 30 GB [31].

3. **Name two features of Intune Win32 app deployment that help ensure applications are only installed when needed.**
   Answer: Detection rules and requirement rules. Explanation: Intune Win32 apps allow you to define *detection logic* so the app installs only if it's not already present [32]. You can also set *requirements* (rules) so the app only installs on devices that meet certain criteria (e.g. OS version, hardware specs) [33].

4. **Which log file on the client contains detailed information about the Intune Win32 app deployment process?**
   Answer: `IntuneManagementExtension.log`. Explanation: The Intune Management Extension logs all the Win32 app processing steps to `IntuneManagementExtension.log` (found under `C:\ProgramData\Microsoft\IntuneManagementExtension\Logs` on the client) [34]. Other logs like `AgentExecutor.log` record script executions, but the IME log covers the overall flow.

5. **How can you inspect the contents (such as the `Detection.xml`) of a `.intunewin` file without uploading it?**
   Answer: Rename the `.intunewin` file to `.zip` and unzip it. Explanation: The Intune documentation states that a `.intunewin` is actually a zip file containing two folders (Contents and

Metadata, including `Detection.xml`). Changing the extension to `.zip` lets you open it with any ZIP tool [35].

6. **Scenario:** You assign a Win32 app to a device group but it doesn't install. Which network requirement might block content delivery?
   Answer: Intune endpoints and CDNs need to be reachable. Explanation: For Win32 app delivery, the client must be able to access the Intune endpoints and Azure CDN locations where the content is stored [36]. If those are blocked by firewall or network policy, the app will not download.

7. **True or False: Intune's Win32 app deployment does not support specifying installation dependencies on other apps.**
   Answer: False. Explanation: Intune Win32 app deployment *does* support dependencies. You can specify other Win32 apps that must be installed before this one [37]. This helps enforce installation order and requirements.

8. **What is one advantage of using Win32 app deployment in Intune for critical Windows 10 updates?**
   Answer: It lets you push arbitrary `.exe` files (including updates) using the Win32 mechanism when no other native update deployment is available [38]. Explanation: The Intune Win32 channel can be used to deploy standalone packages (like hotfixes) that aren't offered through the normal Windows Update or built-in mechanisms [38].

8. MSIX Packaging and Conversion

1. **The MSIX Packaging Tool requires a special driver. What can cause the driver installation to fail?**
   Answer: If the Windows Update service is disabled or policies block Feature on Demand (FOD) packages. Explanation: The MSIX Packaging Tool driver is delivered as a Windows Feature-on-Demand via Windows Update. It will fail to install if the Update service is off or WSUS/GPO policies prevent FOD acquisition [39].

2. **If the MSIX Packaging Tool reports that the driver needs to be reinstalled, what should you do?**
   Answer: Reboot the machine and restart the conversion. Explanation: The known issue states that if the driver indicates a restart is needed, performing a reboot and retrying fixes the problem [40].

3. **You receive an "Insufficient system resources" error (0x80131500) during MSIX packaging conversion. How do you resolve it?**
   Answer: Stop some existing ETW (Event Trace) sessions to free resources, then retry. Explanation: This error occurs when the OS's limit on concurrent ETW sessions (default 64) is exceeded. The solution is to open Performance Monitor, navigate to Data Collector Sets → Event Trace Sessions, and stop unnecessary sessions [41].

4. **After converting an application to MSIX, it fails to run on Windows 10 version 1709 or later due to versioning. What manifest change is needed?**
   Answer: Set the `TargetDeviceFamily` `MinVersion` to at least `10.0.16299.0`. Explanation: The MSIX Packaging Tool versions prior to late 2019 enforced store versioning, meaning packages

had a MinVersion of 10.0.16299.0 as required for Windows 10 1709+. If you see an error about version, editing the MSIX manifest to `<TargetDeviceFamily Name="Windows.Desktop" MinVersion="10.0.16299.0" .../>` fixes the issue [42].

5. **True or False: An MSIX package containing a Windows service can be installed on Windows 10 versions earlier than 1903.**
   Answer: False. Explanation: MSIX packages with services automatically require a minimum OS version of 10.0.19025.0 (Windows 10 2004). Even though you can create it on older OS, it will not install on anything earlier than 2004 [43].

9. Sysinternals Tool Usage

1. **Which Sysinternals tool can monitor registry, file system, and process/thread activity in real time?**
   Answer: Process Monitor (Procmon). Explanation: Process Monitor is an advanced monitoring utility that shows real-time registry, file system, and process/thread events. It combines the older FileMon and RegMon tools and adds powerful filtering [44].

2. **What tool would you use to identify which process has a particular file or registry key open?**
   Answer: Process Explorer. Explanation: Process Explorer can list all handles and DLLs for processes. Its Find Handle or DLL capability shows which processes have a given file or key open [45].

3. **Which Sysinternals utility lists all configured autostart programs, drivers, and services on a system?**
   Answer: Autoruns. Explanation: Autoruns shows the most comprehensive list of auto-start locations (startup folder, Run/RunOnce keys, services, drivers, etc.) and highlights third-party entries [46].

4. **How can Autoruns help troubleshoot unwanted applications loading at boot?**
   Answer: It can disable or delete entries in registry or file locations where autostart items are configured. Explanation: Autoruns displays all automatic startup entries; you can uncheck or delete items (e.g. in Run/RunOnce, Scheduled Tasks, services) to prevent them from loading [47].

5. **Scenario:** An MSI installation fails with an “access denied” because a file is in use. Which Sysinternals tool could help identify which process is locking that file?
   Answer: Process Explorer (using its Find Handle feature) or the `handle.exe` command-line tool. Explanation: Process Explorer can search for a specific file handle and show which process is using it [45]. Similarly, the Sysinternals Handle utility can find the lock on a file.

10. Compatibility Testing, Event Viewer, Dependency Walker

1. **You suspect a missing dependency in a 32-bit DLL on a 64-bit system. Which tool can help identify missing DLLs or functions?**
   Answer: Dependency Walker. Explanation: Dependency Walker analyzes module dependencies. It highlights missing import functions or missing modules in red or yellow. For example, it flags potential problems like unresolved imports in red, indicating missing DLLs or exports [48].

2. **When using Dependency Walker, what does it mean if a system DLL appears with missing functions (highlighted) but reports a warning about delay-load?**
   Answer: Often it means the missing function is delay-loaded and can be safely ignored. Explanation: Dependency Walker warns about missing delay-load functions (e.g. WNetRestoreConnectionA in XP) because it can't know if the application handles them gracefully. In many cases this is normal and ignorable, as documented [48] [49].

3. **How can you use Dependency Walker to troubleshoot a COM component registration failure?**
   Answer: Run `Regsvr32.exe` under Dependency Walker profiling. Explanation: If registering a DLL fails due to a missing dependency, you can open `Regsvr32.exe` in Dependency Walker, then use the profiling feature (F7) with your DLL as the argument. This runs the DLL registration in context and exposes any runtime errors or missing dependencies [50].

4. **Where would you look in Event Viewer for additional error information if an MSI-based install fails?**
   Answer: The Application log under the "MsiInstaller" source or the System log. Explanation: Windows Installer logs key events to the Application log (source MsiInstaller) when it encounters errors. Checking Event Viewer's Application log often shows error codes or messages from MSIEXEC. (General practice, though not directly cited here.)

5. **During compatibility testing, an application crashes on launch. What Event Viewer logs are most relevant to check?**
   Answer: The Application log (for crash or .NET runtime errors) and System log (for driver or system component issues). Explanation: Application crashes often log errors under the Application log (with source like Application Error, .NET Runtime, etc.), while System errors could indicate driver or service faults. Reviewing these logs can point to missing libraries or permission issues.

6. **What key pieces of information can Dependency Walker provide about a DLL/executable?**
   Answer: It lists all static dependencies (DLLs), imported/exported functions, and highlights missing modules/functions. It can also profile execution to catch dynamic dependencies. Explanation: Dependency Walker shows the full dependency tree of a module. Missing or unresolved imports are flagged in red/yellow, and using profiling it can reveal dynamically-loaded dependencies [51] [48].

7. **Scenario:** An older application fails on a new OS. Which compatibility troubleshooting steps could you take?
   Answer:** Use Application Compatibility Toolkit or set compatibility modes; examine Event Viewer for errors; use Dependency Walker to find missing DLLs; ensure required runtimes are installed. Explanation: Compatibility testing may involve trying legacy modes, checking events for clues, and using tools like Dependency Walker to identify incompatible or missing components.

Each question above is designed to test understanding of enterprise packaging and deployment concepts. The answers include brief explanations and references to authoritative sources.

---

[1] The Main Benefits of Application Packaging - A Business Perspective
https://www.advancedinstaller.com/app-packaging-benefits.html

2 30 31 Prepare a Win32 app to be uploaded to Microsoft Intune | Microsoft Learn
https://learn.microsoft.com/en-us/intune/intune-service/apps/apps-win32-prepare

3 4 13 14 15 Active Setup Explained
https://helgeklein.com/blog/active-setup-explained/

5 9 10 Registry entries for VSTO Add-ins - Visual Studio (Windows) | Microsoft Learn
https://learn.microsoft.com/en-us/visualstudio/vsto/registry-entries-for-vsto-add-ins?view=vs-2022

6 7 Deployment Troubleshooting and Log Files | Microsoft Learn
https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/deployment-troubleshooting-and-log-files?view=windows-11

8 Where to troubleshoot failed OSD driver installs : r/SCCM
https://www.reddit.com/r/SCCM/comments/5rnnwq/where_to_troubleshoot_failed_osd_driver_installs/

11 12 The COM add-in that you added to the Templates and Add-Ins list is not loaded when you restart Word - Microsoft Support
https://support.microsoft.com/en-us/topic/the-com-add-in-that-you-added-to-the-templates-and-add-ins-list-is-not-loaded-when-you-restart-word-6173a3dd-713a-8d10-e512-40f1f7c4ca81

16 active directory - Windows Startup and Logon scripts proper permissions - Server Fault
https://serverfault.com/questions/571382/windows-startup-and-logon-scripts-proper-permissions

17 Deploy Registry Fix Using Intune Win32 App HTMD Blog
https://www.anoopcnair.com/deploy-registry-fix-using-intune-win32-app/

18 19 20 21 22 23 How to continue with an installation after reboot?
https://www.advancedinstaller.com/how-to-continue-installation-after-reboot.html

24 25 26 27 28 29 Installing Applications · PSAppDeployToolkit
https://psappdeploytoolkit.com/docs/usage/installing%20applications

32 33 34 35 36 37 38 Support Tip - Understanding the flow behind deployment, delivery, and processing of a Win32 application through Intune - Intune | Microsoft Learn
https://learn.microsoft.com/en-us/troubleshoot/mem/intune/app-management/develop-deliver-working-win32-app-via-intune

39 40 41 42 43 MSIX Packaging Tool Known Issues and Troubleshooting Tips - MSIX | Microsoft Learn
https://learn.microsoft.com/en-us/windows/msix/packaging-tool/tool-known-issues

44 Process Monitor - Sysinternals | Microsoft Learn
https://learn.microsoft.com/en-us/sysinternals/downloads/procmon

45 Process Explorer - Sysinternals | Microsoft Learn
https://learn.microsoft.com/en-us/sysinternals/downloads/process-explorer

46 47 Autoruns - Sysinternals | Microsoft Learn
https://learn.microsoft.com/en-us/sysinternals/downloads/autoruns

48 49 50 51 Dependency Walker Frequently Asked Questions (FAQ)
https://www.dependencywalker.com/faq.html