Mitigation provided by the Cyber Knowledge Graph

1. Perform proper forward and reverse DNS lookup to detect DNS spoofing. Run your code using the lowest privileges that are required to accomplish the necessary tasks [REF-76]. If possible, create isolated accounts with limited privileges that are only used for a single task. That way, a successful attack will not immediately give the attacker access to the rest of the software or its environment. For example, database applications rarely need to run as the database administrator, especially in day-to-day operations.

2. Have a strict approval policy for use of deployment systems.

3. Ensure proper system and access isolation for critical network systems through use of multi-factor authentication

4. Ensure proper system isolation for critical network systems through use of firewalls.

5. Ensure proper system and access isolation for critical network systems through use of group policy.

6. Patch deployment systems regularly to prevent potential remote access through [Exploitation for Privilege Escalation.

7. Verify that account credentials that may be used to access deployment systems are unique and not used throughout the enterprise network.

8. Grant access to application deployment systems only to a limited number of authorized administrators.

9. If the application deployment system can be configured to deploy only signed binaries, then ensure that the trusted signing certificates are not co-located with the application deployment system and are instead located on a system that cannot be accessed remotely or to which remote access is tightly controlled.

10. Ensure that any accounts used by third-party providers to access these systems are traceable to the third-party and are not used throughout the network or used by other third-party providers in the same environment. Ensure there are regular reviews of accounts provisioned to these systems to verify continued business need, and ensure there is governance to trace de-provisioning of access that is no longer required. Ensure proper system and access isolation for critical network systems through use of account privilege separation.

Mitigation and Steps provided by Agentic AI System

The mitigation plan focuses on addressing the root causes (CWE-494, CAPEC-187) and exploitation vectors (CVE-2023-22635, T1072).

Phase 1: Firmware Update (Addressing CVE-2023-22635)

1. Identify Affected Systems:
Inventory all systems using InsydeH2O firmware versions 5.0 through 5.5. This requires checking the BIOS/UEFI version on each system. Use system information tools (e.g., msinfo32 on Windows, dmidecode on Linux) or the system's BIOS/UEFI setup interface.

2. Obtain Updated Firmware:
Contact the device manufacturer (e.g., Dell, HP, Lenovo) for the appropriate firmware update. Do not download firmware from unofficial sources. Check the Insyde website for any available security advisories and updates related to CVE-2023-22635. Search for "CVE-2023-22635" on their site.

3. Test Firmware Update:
Before widespread deployment, test the firmware update on a representative sample of affected systems to ensure compatibility and stability. Verify that the updated firmware version is no longer vulnerable to CVE-2023-22635 according to the device manufacturer's advisory.

4. Deploy Firmware Update:
Use a centralized management tool (e.g., SCCM, Intune) to deploy the firmware update across all affected systems, if available. For systems not managed by a central tool, provide clear instructions to users on how to update their firmware. Monitor the update process to ensure successful deployment and address any issues that arise.

5. Verification: After the firmware update, verify the new firmware version is installed correctly on all targeted machines.

Phase 2: Addressing CWE-494 and CAPEC-187 (Code Download Integrity)

This phase focuses on preventing attackers from injecting malicious code and downloading/executing it.

1. Implement Code Signing:
Ensure that all software and scripts downloaded and executed on the system are digitally signed by a trusted authority.
Configure systems to only execute code with valid signatures. This can be achieved through Group Policy on Windows or similar mechanisms on other operating systems.

2. Implement Integrity Checks:
Use hash-based integrity checks (e.g., SHA-256) to verify the integrity of downloaded files before execution. Compare the downloaded file's hash against a known good hash from a trusted source (e.g., the vendor's website).

3. Secure Download Channels (HTTPS):
Always use HTTPS for downloading software and scripts. This helps prevent man-in-the-middle attacks. Verify the SSL/TLS certificate of the download server to ensure you are communicating with the legitimate source.

4. Input Validation and Sanitization:
Implement robust input validation and sanitization to prevent code injection vulnerabilities. This includes validating all user inputs and escaping special characters. Use parameterized queries or prepared statements to prevent SQL injection. Avoid using dynamic code evaluation functions (e.g., eval() in JavaScript, exec() in Python) unless absolutely necessary and with extreme caution.

5. Least Privilege:
Run applications and services with the least privileges necessary to perform their functions. This limits the damage an attacker can cause if they compromise the application. Implement privilege separation to isolate different parts of the system.

Phase 3: Addressing T1072 (Software Deployment Tool Abuse)

This phase hardens software deployment infrastructure.

1. Secure Software Deployment Tools:

Restrict access to software deployment tools (e.g., SCCM, GPO, Ansible) to authorized personnel only. Implement multi-factor authentication (MFA) for all accounts with access to these tools.

2. Monitor Software Deployment Activities:

Implement logging and monitoring of all software deployment activities. Alert on any unusual or suspicious activity, such as deployments outside of normal business hours or to unexpected systems.

3. Regular Security Audits:

Conduct regular security audits of the software deployment infrastructure to identify and address vulnerabilities. Review the configurations of software deployment tools to ensure they are following security best practices.

4. Patch Management:

Keep software deployment tools and their underlying infrastructure up-to-date with the latest security patches. Prioritize patching vulnerabilities that could be exploited to gain unauthorized access to the deployment tools.

5. Principle of Least Privilege (Again):

Ensure that the accounts used to deploy software have only the minimum privileges required. Avoid using domain administrator accounts for software deployment.

Step-by-Step Instructions

Step 1: Inventory Systems: Use asset management tools or manual methods to identify all systems using InsydeH2O firmware and record their versions.
Step 2: Acquire Firmware Updates: Visit the device manufacturer's website and search for updates related to CVE-2023-22635. Download the appropriate update for each system model.
Step 3: Test Updates: Apply the update to a small subset of systems and monitor for any issues.
Step 4: Deploy Updates: Use a centralized management tool or provide instructions for manual updates.
Step 5: Implement Code Signing and Integrity Checks: Configure operating systems and applications to enforce code signing and integrity checks.
Step 6: Secure Download Channels: Ensure all software downloads use HTTPS and verify SSL/TLS certificates.
Step 7: Harden Software Deployment Tools: Restrict access, implement MFA, and monitor deployment activities.
Step 8: Regular Audits and Patching: Conduct regular security audits and patch all systems and tools promptly.