W18\_2

RBAC In windows user management system

//make this a table in report

In file properties under security tab

permissions: full control,Modify,R,RX,w,Special permissions

Roles:

authenticated users: n,y,y,y,y,n

system: y,y,y,y,y,n

administrators: y,y,y,y,y,n

users: n,n,y,y,n,n

In settings-> accounts -> family and other users

standard user

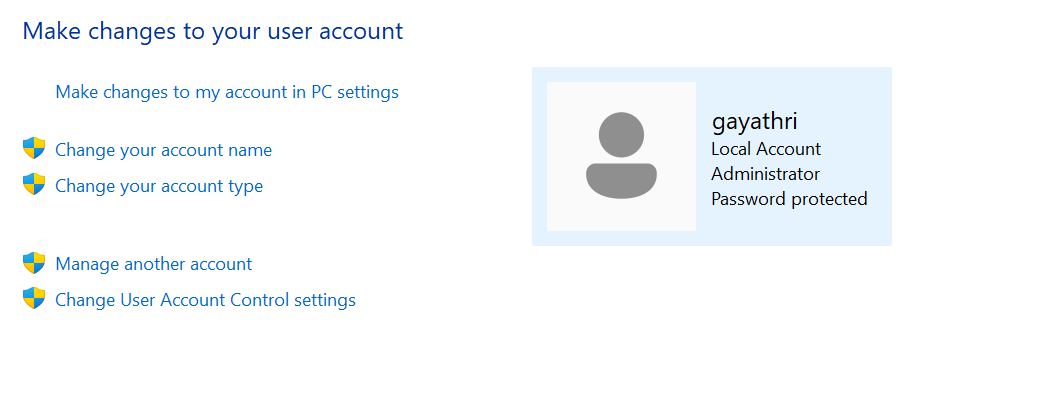
administrator

organiser

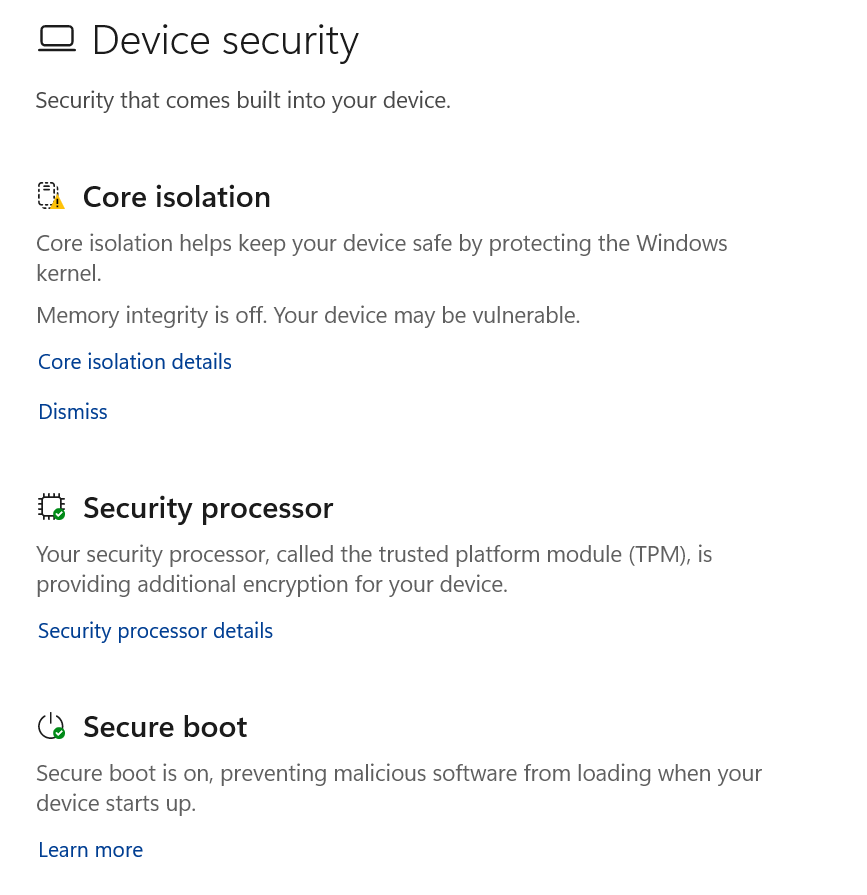
1. **Standard User:**
   * **File and Folder Access:** Standard users typically have read and execute permissions for most files and folders on the system. They can create, modify, and delete files within their user profile folders.
   * **Application Installation:** Standard users usually don't have the permission to install or uninstall applications that affect the entire system. They can install or uninstall programs within their user profile.
   * **System Settings:** Standard users may have limited access to system settings. They can customize personal preferences but may not be able to change critical system settings.
2. **Administrator:**
   * **File and Folder Access:** Administrators have full control over the file system, allowing them to read, write, execute, modify, and delete files and folders throughout the system.
   * **Application Installation:** Administrators have the authority to install and uninstall applications that affect the entire system, including system-wide software installations.
   * **System Settings:** Administrators have complete control over system settings. They can configure security settings, install drivers, and make changes to the system configuration.
3. **Organizer (Assuming Specific Role):**
   * The term "Organizer" doesn't have a specific built-in role in Windows. If you are referring to a specific application or organization-specific role, the permissions would depend on the implementation by that application or organization.
   * Permissions associated with an "Organizer" role would need to be defined by the software or system that uses this role. It could include various levels of access to manage and organize resources.

In task manager -> users



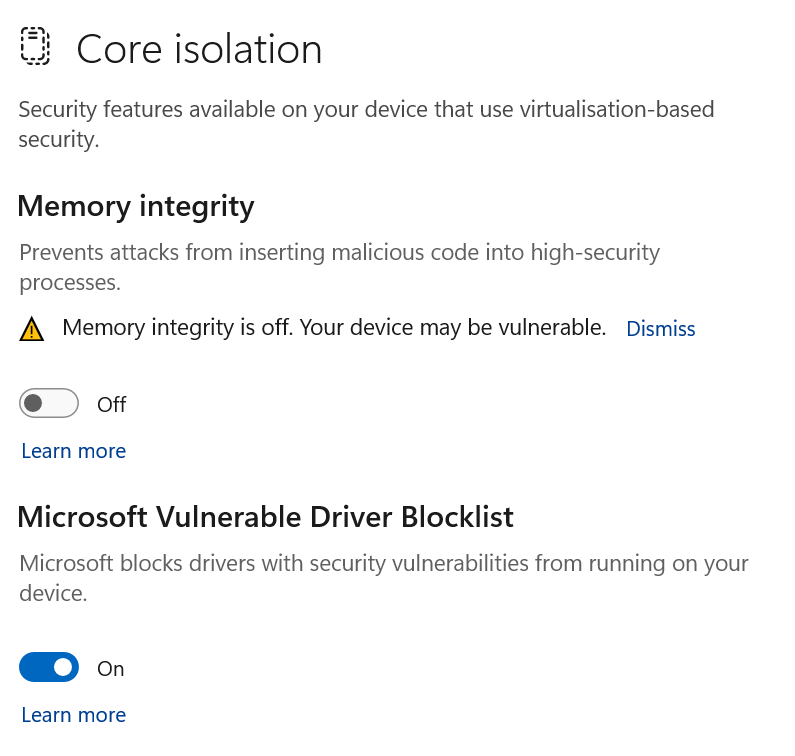


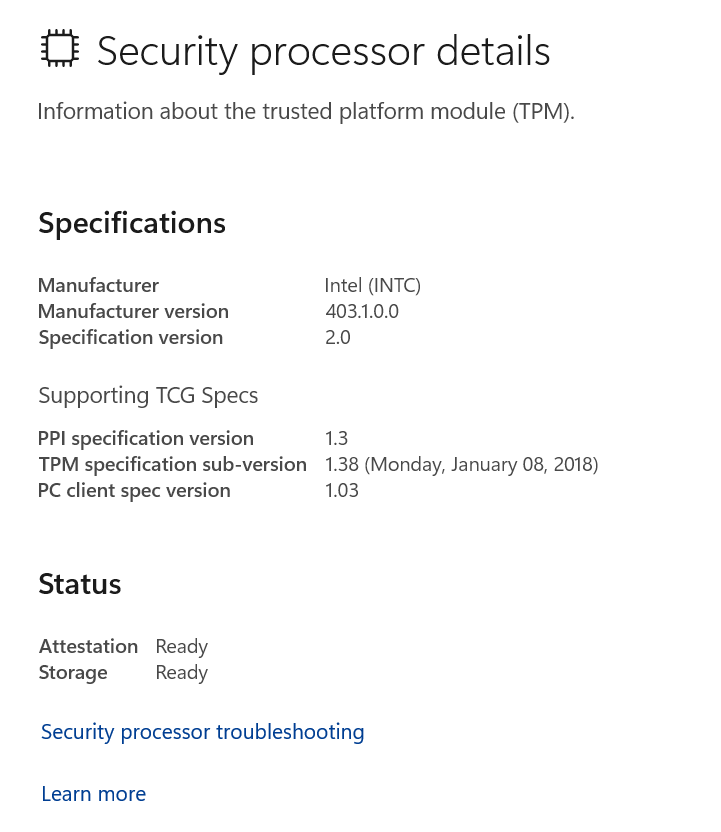
In windows security app-> device security



there are specific security features and configurations in the "Windows Security" app that can be managed based on administrative roles or user account types. Here are some key security configurations and how they are typically managed:

1. **Windows Security Dashboard:**
   * The "Windows Security" dashboard provides an overview of various security features, including antivirus, firewall, device security, and more.
   * Administrators (users with administrative privileges) have access to all features and can make changes as needed.
2. **Virus & Threat Protection:**
   * Administrators have full control over virus and threat protection settings.
   * Standard users have limited options, often restricted to viewing the status and running scans.
3. **Firewall & Network Protection:**
   * Administrators can configure firewall rules and manage network protection settings.
   * Standard users may have limited access, primarily viewing the firewall status.
4. **App & Browser Control:**
   * Administrators can configure settings related to app and browser control, including SmartScreen and exploit protection.
   * Standard users may have limited access, typically viewing the status.
5. **Device Security:**
   * Device Security settings, including features like Core Isolation, are often accessible by administrators.
   * Standard users may not have access to certain advanced security features.
6. **Device Performance & Health:**
   * Administrators can check and manage the health of the device.
   * Standard users may have limited access to health-related information.

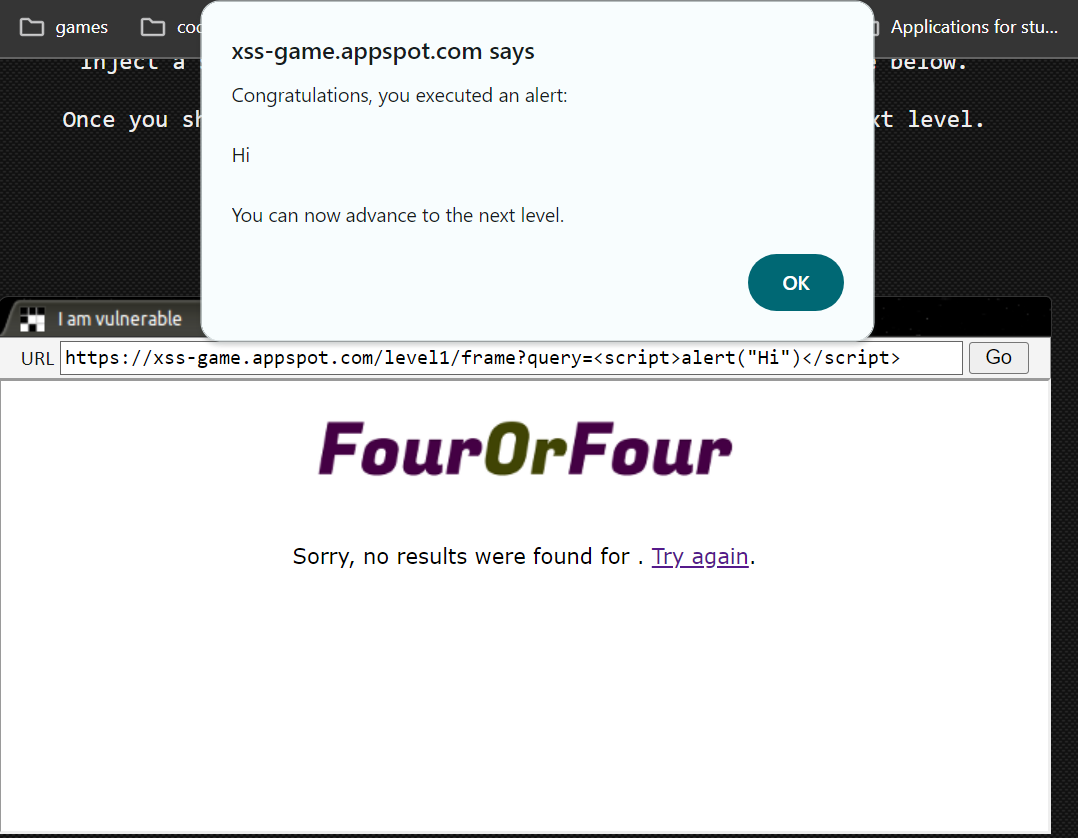




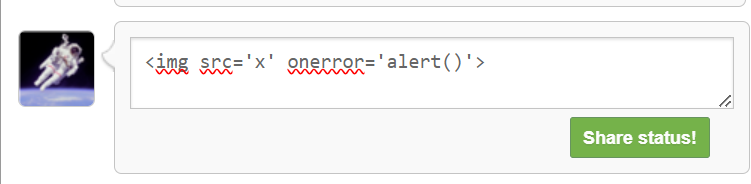
W18\_3

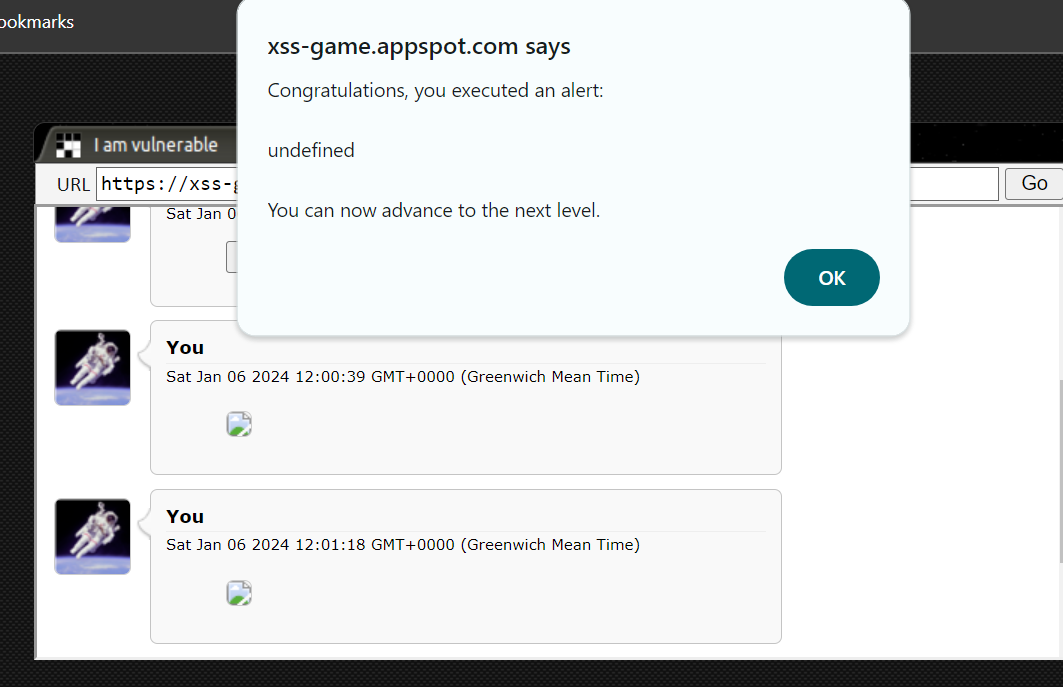
Level 1



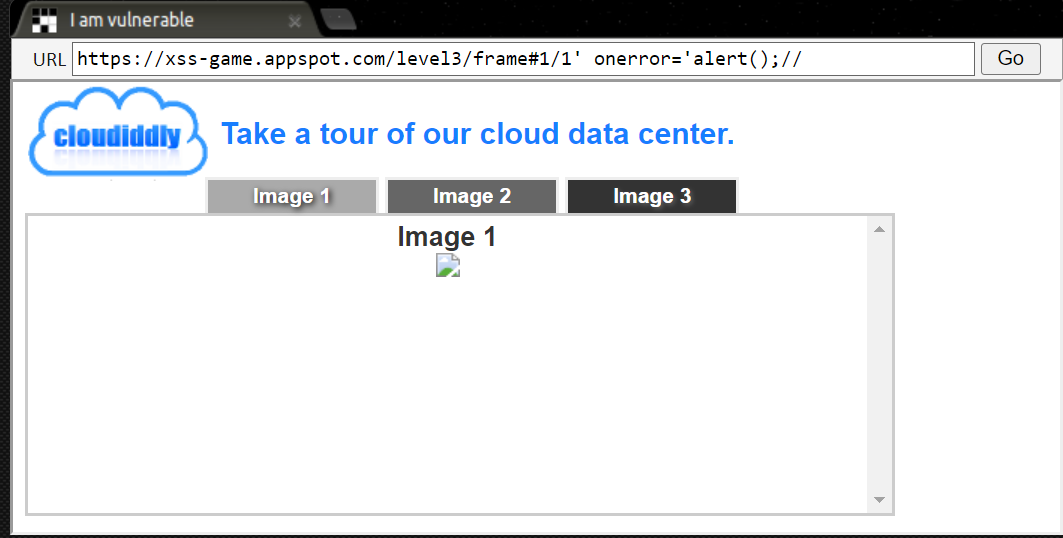


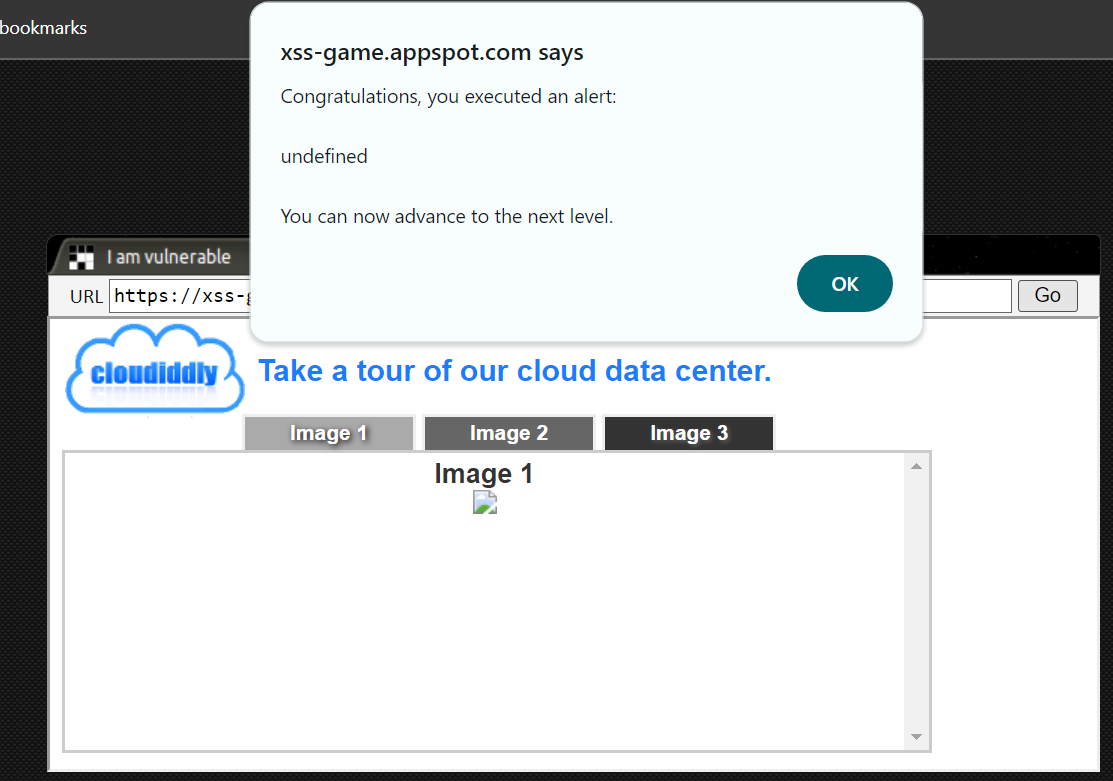
Level 2





Level 3





Here's how XSS attacks can bypass the Same-Origin Policy:

1. **Script Injection:**
   * In a typical XSS attack, an attacker injects malicious scripts into user inputs, such as form fields, URL parameters, or even in the content of the website itself (e.g., comments, user profiles).
2. **Executing in the Victim's Browser:**
   * When a user visits a page that contains the injected script, the browser unknowingly executes the malicious code as part of the page's content.
3. **Context of Origin:**
   * The injected script runs in the context of the victim's origin (the domain of the web application). This allows the script to access and manipulate the content of the web page, interact with cookies, make requests to the same origin, and even perform actions on behalf of the authenticated user.
4. **Access to Sensitive Information:**
   * XSS can be used to steal sensitive information, such as session cookies, user credentials, and other data stored within the victim's browser.
5. **Actions on Behalf of the User:**
   * The attacker-controlled script can perform actions on behalf of the user, such as making requests to the web application with the victim's session credentials, changing user settings, or initiating financial transactions.
6. **Bypassing SOP Restrictions:**
   * Since the injected script runs in the context of the victim's origin, it has the same level of access to resources and functionalities as legitimate scripts on the page. This effectively bypasses the SOP restrictions, allowing the attacker to interact with resources within the same origin.

W18\_4

**Example: Docker Container Sandbox**

**Description:** Docker is a widely used platform for containerization, allowing applications and their dependencies to be packaged and isolated within lightweight containers. Containers provide a form of sandboxing, allowing applications to run in an isolated environment.

**Security Settings and Access Control:**

1. **Namespace Isolation:**
   * Docker uses Linux namespaces to provide isolation for processes, network, users, and other system resources. Each container has its own set of namespaces, preventing interference between containers.
2. **Control Groups (cgroups):**
   * Docker leverages cgroups to limit and control the resource usage (CPU, memory, etc.) of containers. This prevents a misbehaving or compromised container from consuming excessive resources.
3. **Read-Only Filesystem:**
   * Containers often run with a read-only filesystem, limiting the ability of the application inside the container to modify system files.
4. **Seccomp Profiles:**
   * Docker supports Seccomp (secure computing mode) profiles to restrict the system calls that a container can make, reducing the attack surface.
5. **Capabilities:**
   * Docker containers can drop unnecessary Linux capabilities, limiting the privileges of the containerized processes.

**Enforcement:**

* **Docker Daemon Policies:**
  + The Docker daemon enforces security policies, such as access control policies that restrict which users can interact with the Docker daemon.
* **Docker Images and Registry Security:**
  + Docker images often include security settings, and registries may enforce policies to ensure that only authorized images are used.
* **Runtime Security:**
  + Container runtimes, like containerd or runc, enforce security settings specified by the container image and runtime configuration.

**Known Vulnerabilities:** As with any software, Docker has faced vulnerabilities in the past. Common security concerns include vulnerabilities in the Linux kernel that might be exploited to escape containerization. Security patches and updates are regularly released to address such vulnerabilities. Security best practices, including using official images, minimizing the attack surface, and keeping software up to date, are essential to maintaining a secure containerized environment.

W18\_5

**General Overview of Tripwire Implementation:**

1. **Agent-Based Architecture:**
   * Tripwire often employs an agent-based architecture where agents are installed on endpoints or servers to monitor file integrity and system configurations.
2. **File System Monitoring:**
   * The agents continuously monitor specified files, directories, and system configurations for changes. This includes additions, modifications, or deletions of files.
3. **Baseline Creation:**
   * Tripwire establishes a baseline or snapshot of the initial state of the monitored files and configurations. This baseline serves as a reference point for detecting subsequent changes.
4. **Checksums and Hashing:**
   * Cryptographic hash functions, such as SHA-256, are commonly used by Tripwire to generate checksums or hashes for files. These values are compared against the baseline to identify alterations.
5. **Policy Configuration:**
   * Administrators can define policies using Tripwire to specify which files or directories should be monitored, the types of changes that are allowed, and the actions to take in the event of unauthorized modifications.
6. **Alerting and Reporting:**
   * Tripwire generates alerts when unauthorized changes are detected. These alerts can be configured to trigger notifications via email, SNMP, or other methods. Additionally, comprehensive reports on file integrity and system changes are often available through a centralized console.
7. **Integration with SIEM:**
   * Tripwire can integrate with Security Information and Event Management (SIEM) solutions to provide a broader context for security events and facilitate correlation with other security information.
8. **Regulatory Compliance:**
   * Tripwire is often used to meet regulatory compliance requirements, providing a solution for monitoring and documenting changes to critical files and configurations.