**Hackworld Comprehensive Upgrade and Transfer Management Script**

This script automates the management of upgrades and transfers in the Hackworld game. It includes features such as upgrade target control, stealing mode, and most recent victim tracking. The script ensures that resources are efficiently utilized and upgrades are prioritized based on user-defined targets.

**Standard Logic (When No Checkboxes Are Active)**

**Upgrade Logic**

1. **Priority Order**:
   * The script upgrades components in a specific priority order, focusing first on the miner, then the vault, and finally on the scanner and stealer if their upgrade costs are below a certain threshold.
2. **Conditions for Upgrades**:
   * **Miner**: The script checks if there are enough coins in the vault to cover the cost of a miner upgrade. If so, it initiates the upgrade.
   * **Vault**: If the miner upgrade cost is greater than the current vault capacity, the script prioritizes upgrading the vault to handle more coins. This is done to ensure future miner upgrades are feasible.
   * **Scanner and Stealer**: These components are upgraded only if their costs are below a calculated transfer threshold (25% of the miner upgrade cost by default).

**Transfer Logic**

1. **Threshold-Based Transfers**:
   * The script transfers coins to the vault based on a threshold, which is calculated as 25% of the miner upgrade cost by default. This ensures that there are always enough coins available for critical upgrades.
2. **Transfer Conditions**:
   * Coins are transferred to the vault if the current coin amount exceeds the transfer threshold, ensuring that resources are efficiently managed and available for upgrades.

**Key Features**

1. **Upgrade Target Control**
2. **Stealing Mode**
3. **Most Recent Victim Tracking**
4. **Inactivity Detection**
5. **Persistent State Management**
6. **Console Logging**

**Feature Details and Instructions**

**1. Upgrade Target Control**

**Description**: Allows users to set desired target levels for each component and dynamically compares them with current levels to prioritize upgrades.

**Components**:

* Firewall (fw)
* Scanner (sc)
* Miner (mi)
* Stealer (st)
* Vault (va)

**How It Works**:

* The script continuously checks the current levels of the components.
* Compares current levels with target levels.
* Prioritizes upgrades for components that are furthest from their target levels.

**Instructions**:

* Open the control panel at the bottom-left of the screen.
* Check the "Activate Upgrade Target Control" checkbox to enable the feature.
* Set target levels for each component using the input fields.
* The script will automatically manage upgrades to reach the target levels.

**2. Stealing Mode**

**Description**: Ensures a buffer is maintained in the vault to allow for scanning and stealing from other players. Overrides other upgrade logic when active.

**How It Works**:

* When enabled, the script reserves a buffer equal to the cost of one scanner upgrade plus one stealer upgrade.
* Ensures that this buffer is not used by other upgrades.
* Uses single upgrade buttons instead of max upgrade buttons to avoid depleting the buffer.

**Instructions**:

* Check the "Activate Stealing Mode" checkbox in the control panel to enable stealing mode.
* The script will automatically maintain the required buffer and adjust upgrade behavior.

**3. Most Recent Victim Tracking**

**Description**: Tracks the most recent player who was stolen from by monitoring the event log.

**How It Works**:

* Observes the event log for messages indicating successful steals.
* Updates the most recent victim display in the control panel.

**Instructions**:

* No additional configuration is needed.
* The most recent victim will be displayed in the "Most Recent Victim" section of the control panel.

**4. Inactivity Detection**

**Description**: Detects periods of inactivity and refreshes the page to ensure the script remains active.

**How It Works**:

* Monitors the coins and vault values.
* If there is no change for an extended period, the script reloads the page.

**Instructions**:

* No additional configuration is needed.
* The script will automatically detect inactivity and refresh the page if necessary.

**5. Persistent State Management**

**Description**: Saves the state of the control panel and other settings to localStorage to preserve them across page reloads.

**How It Works**:

* Saves settings such as the active state of upgrade target control, stealing mode, transfer threshold multiplier, and target levels.
* Restores these settings when the page is reloaded.

**Instructions**:

* Settings are automatically saved and restored.
* Simply configure the settings in the control panel, and they will persist across sessions.

**Script Overview**

Here’s a detailed overview of the script's main functions and their purposes:

**Initialization and Main Loop**

* **initAutomation**: Initializes the script, sets up the control panel, starts the event log monitor, and sets the interval for performing strategic upgrades and transfers.
* **waitForJQuery**: Ensures jQuery is loaded before initializing the script.

**Upgrade and Transfer Logic**

* **performStrategicUpgradesAndTransfers**: Main function that coordinates upgrades and transfers based on the active mode (standard or upgrade target control).
* **performUpgradeTargetControl**: Handles upgrades based on user-defined target levels.
* **performTransfers**: Manages transfers of coins to the vault based on the transfer threshold.

**Upgrade Functions**

* **upgradeComponentToTarget**: Checks if a component should be upgraded to its target level.
* **upgradeComponentIfNeeded**: Determines if an upgrade should be performed and initiates it.
* **performSingleUpgrade**: Executes a single upgrade action.
* **performMultipleUpgrades**: Executes multiple upgrades with a delay between each.

**Utility Functions**

* **getCurrentLevels**: Retrieves the current levels of all components.
* **getUpgradeCosts**: Retrieves the upgrade costs of all components.
* **getDiscrepancies**: Calculates the discrepancies between current and target levels.
* **getComponentId**: Maps component keys to their respective IDs.
* **updateCurrentLevelDisplay**: Updates the displayed current levels in the control panel.
* **saveState**: Saves the current state to localStorage.
* **loadState**: Loads the saved state from localStorage.
* **checkForInactivity**: Monitors for inactivity and forces a page refresh if necessary.