

Z powyższego ćwiczenia wynikają następujące wnioski:

1. **Zabezpieczenie oprogramowania za pomocą sum kontrolnych:** Stosowanie sum kontrolnych, takich jak SHA256, jest skutecznym sposobem na weryfikację integralności pliku. Jest to popularna praktyka w celu zapewnienia, że pliki nie zostały zmienione lub uszkodzone podczas przesyłania lub przechowywania.
2. **Automatyzacja zabezpieczeń w PowerShell:** PowerShell pozwala na automatyzację procesu obliczania i sprawdzania sum kontrolnych za pomocą wbudowanych poleceń cmdlet takich jak Get-FileHash i Out-File. Dzięki temu można łatwo wdrażać i zarządzać zabezpieczeniami oprogramowania.
3. **Przechowywanie sum kontrolnych:** Ważne jest przechowywanie wygenerowanych sum kontrolnych w bezpiecznym miejscu, takim jak plik tekstowy. Pozwala to na późniejszą weryfikację integralności pliku poprzez porównanie zapisanej sumy kontrolnej z wartością obliczoną w przyszłości.
4. **Weryfikacja sum kontrolnych:** Regularne sprawdzanie sum kontrolnych pozwala na szybkie wykrycie ewentualnych zmian w plikach. Warto zautomatyzować proces weryfikacji i powiadomień w przypadku niezgodności sum kontrolnych, co umożliwia szybką reakcję na ewentualne zagrożenia.
5. **Uwzględnienie białych znaków:** W skrypcie należy pamiętać o uwzględnieniu ewentualnych białych znaków w sumach kontrolnych wczytanych z pliku tekstowego. Metoda .Trim() jest używana w celu usunięcia zbędnych białych znaków, co pomaga w dokładnym porównaniu sum kontrolnych.