**Authentication and encryption of text messages in client-server communication**

Grupa “lima”: Kacper Modelski, Mikołaj Koper, Jakub Orluk

**Automated Security Helper**

Automated security helper sprawdza czy kod źródłowy zawiera hasła, klucze api, klucze AWS w repozytorium oraz sprawdza kod pod kątem występujących podatności (Python, Jupyter Notebook, JavaScript, NodeJS).

MediConv-client:

ASH nie znalazł żadnych podatności, warto zaznaczyć, że skanowanie programem checkov zawarte w ASH nie powiodło się.

Cały raport znajduje się w archiwum .zip w folderze Client/Automated-security-helper.txt.

MediConv-backend:

Checkov znalazł następujące zastrzeżenia co do kodu:

- Brak regularnego sprawdzania czy dany kontener w Dockerze przestał odpowiadać (instrukcja HEALTHCHECK). Zagrożenie wynikające z tej podatności jest niskie (LOW).

- Kontener Dockera nie jest uruchomiony w trybie użytkownika non-root. Zagrożenie wynikające z tej podatności jest niskie (LOW).

Cały raport znajduje się w archiwum .zip w folderze Backend/Automated-security-helper.txt.

**Betterscan-ce**

Betterscan-ce wykonuje ponad 2300+ testów pod kątem wad i luk w zabezpieczeniach.

MediConv-client:

Betterscan-ce znalazł 49 ostrzeżeń. Dotyczą one głównie dołączania do kodów zmiennych/bibliotek, które nie są potem wykorzystywane. Inne zagrożenie, o którym należy wspomnieć to przechowywanie klucza kryptograficznego w kodzie źródłowym zamiast w oddzielnych plikach konfiguracyjnych.

Cały raport znajduje się w archiwum .zip w folderze Client/Betterscan-ce.html

MediConv-backend:

Betterscan-ce znalazł 13 ostrzeżeń. W MediConv-backend, podobnie jak w MediConv-client, znajduje się wiele nieużywanych bibliotek oraz klucz kryptograficzny jest przechowywany bezpośrednio w kodzie źródłowym. Program wykorzystuje również Spring RestController który może być podatny na ataki.

Cały raport znajduje się w archiwum .zip w folderze Backend/Betterscan-ce.html

**Gitleaks**

Gitleaks sprawdza repozytorium czy zawiera wrażliwe hasła lub inne wartości które nie powinny być udostępnione publicznie. Zarówno jak w MediConv-backend jak i MediConv-client nie znalazł żadnych problemów.

Raport dla MediConv-client znajduje się w archiwum .zip w folderze Client/Gitleaks.txt

Raport dla MediConv-backend znajduje się w archiwum .zip w folderze Backend/Gitleaks.txt

**Horusec**

Horusec jest najbardziej rozbudowanym programem do automatycznego sprawdzania podatności. Znalazł on najwięcej problemów z aplikacją.

MediConv-client:

Horusec znalazł aż 93 podatności z czego 4 z nich są na poziomie LOW, 4 na poziomie MEDIUM, 27 na poziomie HIGH oraz 58 na poziomie CRITICAL.

Podatności na poziomie CRITICAL:

- używana wersja Spring Framework jest podatna na zdalne uruchamianie kodu

- zdalne wstrzykiwanie kodu Apache Log4j

- pliki debugowania zawierają token dostępu do modyfikacji repozytorium

Te trzy podatności dotyczą kilkunastu linii kodu (stąd aż 93 komunikaty).

Podatności na poziomie HIGH:

- wykorzystywanie biblioteki javax.crypto

- wartości podawane przez użytkownika nie są przekazywane do bazy danych SQL w sposób bezpieczny, co pozwala na SQL injection

Podatności na poziomie MEDIUM:

- Uwierzytelnianie wykorzystuje kodowanie Base64, które jest łatwe do złamania

Podatności na poziomie LOW:

- Uwierzytelnianie wykorzystuje dekodowanie Base64

Cały raport znajduje się w archiwum .zip w folderze Client/Horusec.txt

MediConv-backend:

Horusec znalazł jedną podatność na poziomie HIGH (użytkownik w kontenerze Dockera jest rootem) oraz jedną podatność na poziomie MEDIUM (wykorzystywanie kodowania Base64).

Cały raport znajduje się w archiwum .zip w folderze Backend/Horusec.txt

**Mobile Security Framework**

W przypadku MobSF pomyślny skan udał się na MediConv-client. Ponieważ Mediconv-backend jest serwerem a nie aplikacją mobilną skanowanie nie było możliwe.

MobSF nie znalazł żadnych problemów z aplikacją MediConv-client.

Cały raport znajduje się w archiwum .zip w folderze Client/Mobsfscan.txt