**Eötvös Loránd Tudományegyetem**

Informatikai Kar

*Programozási nyelvek és Fordítóprogramok Tanszék*

Nem biztonságos kód biztonságossá transzformálása a RefactorErl

segítségével

**Témavezetők:** **Szerző:**

Dr. Tóth Melinda, Dr. Bozó István Juhász Alexandra Nelli

Egyetemi docens, Tudományos munkatárs Programtervező informatikus MSc.

**Budapest, 2025**

Témabejelentő / Absract (azt újra kell írni)

A diplomamunkám és a kutatásom célja olyan automatikus/részben automatikus transzformációk definiálása a RefactorErlben, melyek a nem biztonságos vagy sérülékeny kódrészleteket biztonságossá alakítják az Erlang programozási nyelvben.

Az Erlang egy elosztott, skálázható, hibatűrő programok fejlesztését biztosító programozási nyelv. Funkcionális jellegű, magasszintű nyelv, melyben ugyan nem lehetetlen, de kevés lehetőség van arra, hogy biztonsági sérülékenységeket hagyjanak a programozók. Azonban a nyelvet gyakran használják kritikus infrastruktúrák (banki rendszerek, telekommunikáció, közlekedés irányító stb.) szoftvereinek megvalósításához, így különösen fontos az, hogy a forráskódokban ne maradjanak kiaknázható támadási pontok, melyek komoly biztonsági kockázatot jelenthetnek. Statikus elemzési módszerekkel az ilyen pontok felderíthetők.

A RefactorErl egy nyílt forráskódú statikus elemző eszköz, amely az Erlang fejlesztők mindennapi munkáját támogatja különféle kódelemzési és keresési funkciókkal. A kutatásom szempontjából azonban a legfontosabb, hogy képes azonosítani a biztonsági kockázatot jelentő kódrészleteket és lehetőséget biztosít forráskód transzformációk/refaktorálások definiálására is.

A RefactorErl által feltárt, Erlangban előforduló sérülékeny kódrészletek egy része olyan, mely a funkcionalitás megtartása mellett átalakítható lenne úgy, hogy a sérülékenység ne legyen kiaknázható. Például, egy beszúrásos támadás esetén egy megfelelő input ellenőrzéssel biztonságossá tehető az adott kód. Célom, hogy az ilyen biztonságossá tevő mintákat definiáljam és transzformációkat adjak a RefactorErlben, melyek a nem biztonságos kódrészletek biztonságossá alakítják. Ezek az automatikus transzformációk minimalizálják az emberi beavatkozást, hozzájárulva a hatékonyabb és biztonságosabb fejlesztési folyamatokhoz.

Tartalomjegyzék

[1. Bevezetés 4](#_Toc184843471)

[1.1. A kutatás jelentősége 4](#_Toc184843472)

[2. Irodalmi áttekintés 5](#_Toc184843473)

[3. Nem biztonságos kód felismerése 6](#_Toc184843474)

[3.1. Gyakori sérülékenységek 6](#_Toc184843475)

[3.2. Biztonsági minták 6](#_Toc184843476)

[3.3. Sérülékenységek azonosítása RefactorErl segítségével 6](#_Toc184843477)

[3.3.1. RefactorErl bemutatása 6](#_Toc184843478)

[4. Eredmények elemzése 7](#_Toc184843479)

[5. Összegzés és továbbfejlesztési lehetőségek 8](#_Toc184843480)

[Irodalomjegyzék 9](#_Toc184843481)

[Forráskód jegyzék 10](#_Toc184843482)

# 1. Bevezetés

## 1.1. A kutatás jelentősége

# 2. Irodalmi áttekintés

## 2.1. Statikus kódelemzés

## 2.3. RefactorErl és más statikus elemzők összehasonlítása

# 3. Nem biztonságos kód felismerése

## 3.1. Gyakori sérülékenységek

## 3.2. Biztonsági minták

## 3.3. Sérülékenységek azonosítása RefactorErl segítségével

# 4. Eredmények elemzése

# 5. Összegzés és továbbfejlesztési lehetőségek

# Irodalomjegyzék

A könyvek idézésének formája a következő:  
[<hivatkozási azonosítója>] <szerző neve>: <a könyv címe>, <a kiadó neve>, <a kiadás éve>, [<terjedelme>], <ISBN szám>. Például: [2] Jan Sommerville: Software Engineering, Addison Wesley, 1988, [742], ISBN-0201- 42765-6.  
Folyóiratcikk idézésének formája a következő:  
[<hivatkozási azonosítója>] <szerző neve>: <a cikk címe>, <a folyóirat címe>, <Vol. száma>, <évszám>, [terjedelme].  
Hálózaton elérhető forrásmunka esetén a „link” megadása után adjuk meg az **elérés dátumát**!

# Forráskód jegyzék