Sigurnost operacijskih sustava i aplikacija

Dizajn arhitekture s naglaskom na sigurnost

Sebastian Medjaković, 11.4.2025.

Pregled predavanja

- Pitanja za ispite
- Motivacija
- Sigurnosni zahtjevi sustava
- Ranjivosti u arhitekturi vs. greške u kodu
- Vrste ranjivosti u arhitekturi
- Najčešće ranjivosti u arhitekturi
- Metode analize arhitekture
- Zaključak
- Literatura

Pitanja za ispite

- Navedite i objasnite tri sigurnosna zahtjeva koja bi trebao implementirati svaki sustav
- Zašto je bitno da se razmišlja o sigurnosti u samom početku SDLC-a
- Koja je razlika između bug-a i ranjivosti u arhitekturi
- Objasnite prednosti i mane korištenja eksternih komponenti
- Objasnite Tactic-Oriented Architectural Analysis (ToAA)

Motivacija

- Preveliki fokus na funkcionalnost
- Cijena popravka ranjivosti raste s napretkom sustava
- Zakonski i regulatorni zahtjevi
  - Opća uredba o zaštiti podataka (GDPR) – kazne za nepoštivanje mogu iznositi više milijuna eura

Sigurnosni zahtjevi sustava

- Autentifikacija
- Autorizacija
- Povjerljivost
- Integritet podataka
- Odgovornost
- Dostupnost
- Neopozivost

Sigurnosni zahtjevi sustava

- Autentifikacija (authentication)
  - Proces provjere identiteta korisnika
  - Samo ovlašteni korisnici bi trebali moći pristupiti resursima
  - Lozinka, biometrija, MFA
- Autorizacija (authorization)
  - Nakon autentifikacije
  - Određuje što korisnik smije raditi unutar sustava
  - Sustav uloga RBAC (Role-Based Access Control)

Sigurnosni zahtjevi sustava

- Povjerljivost (confidentiality)
  - Podaci moraju biti dostupni samo ovlaštenim korisnicima
  - Šifriranje podataka, prijenos sigurnim kanalima
- Integritet podataka (data integrity)
  - Jamči da podaci ostanu točni, potpuni i nepromijenjeni tijekom prijenosa i pohrane

Sigurnosni zahtjevi sustava

- Odgovornost (accountability)
  - Sustav mora omogućiti da se prate aktivnosti korisnika
  - Logiranje
- Dostupnost (availability)
  - Podaci i sustavi moraju biti dostupni, sustav neprekidno radi
  - Redundancija, zaštita od DDoS napada
- Neporecivost (non-repudiation)
  - Korisnici ne mogu negirati da su obavili neku radnju
  - Digitalni potpis

Ranjivosti u arhitekturi vs. greške u kodu

- Greška u kodu (bug)
  - Nešto ne radi kako je zamišljeno
  - Zanemarivanje nekog uvjeta ili korištenje krivog algoritma (ne mora biti vezan za sigurnost)
  - Popravak se postiže izmjenom koda bez izmjene arhitekture

Ranjivosti u arhitekturi vs. greške u kodu

- Ranjivost u arhitekturi
  - Loša odluka u dizajnu sustava
  - Autentifikacija na klijentskoj strani
  - Popravak zahtijeva redizajn dijela sustava
  - S vremenom popravak postaje sve zahtjevniji i povećava se šansa da napadač pronađe i iskoristi ranjivost

Vrste ranjivosti u arhitekturi

- Ranjivost zbog propusta
  - Sigurnosna funkcionalnost nedostaje jer nije planirana u dizajnu
  - Nema provjere unosa korisnika
- Ranjivost zbog pogrešnog izvršenja
  - Funkcionalnost je planirana i napravljena, ali loše ili pogrešno
  - Korištenje slabih algoritama za šifriranje kako bi se poboljšale performanse

Vrste ranjivosti u arhitekturi

- Ranjivost u realizaciji
  - Pogreška u implementaciji
  - Često na niskoj razini koda
  - Jako slično kao greška u kodu (bug) samo što za razliku od bug-a uvijek narušava sigurnost implementacije

Najčešće ranjivosti u arhitekturi

- Izvršavanje sigurnosnih funkcionalnosti u nesigurnoj okolini
- Nepravilno korištenje autentifikacije
- Nedostatak autorizacije nakon autentifikacije
- Nerazdvajanje podataka i naredbi, izvršavanje naredbi iz nepouzdanih izvora
- Nedostatak validacije podataka
- Nepravilno korištenje kriptografije
- Krivo upravljanje povjerljivim podacima
- Ne uzimanje u obzir različitih korisnika
- Slijepo vjerovanje i nepravilno korištenje vanjskih komponenti
- Razvoj bez fleksibilnosti na buduće promjene

Izvršavanje sigurnosnih funkcionalnosti u nesigurnoj okolini

- Sustavi koji se sastoje od više komponenti ovise o međusobnom povjerenju
- Opasno ako se jedna komponenta izvršava u nepouzdanom okruženju
- Podaci koji dolaze s klijenta trebaju se smatrati nepouzdanima dok se ne dokaže suprotno, validacija podataka
- Nesigurni klijenti: mobilne aplikacije, API pozivi, pametni uređaji
- Treba razmisliti o tome gdje će se kod izvoditi, gdje će odlaziti podaci i otkud će dolaziti podaci u naš sustav

Nepravilno korištenje autentifikacije

- Korisnik se može autentificirati s nečim što zna (šifra), nečim što je (biometrija) ili nečim što ima (mobilni uređaj) ili kombinacijom tih faktora
- Vjerodajnice moraju biti izrađene na način da se ne mogu provaliti
- Potrebno je odrediti vrijeme trajanja vjerodajnice nakon koje je potrebno ponovno provesti autentifikaciju
- Sustav treba imati samo jednu komponentu za autentifikaciju

Nedostatak autorizacije nakon autentifikacije

- Nema svaki autentificirani korisnik pravo pristupiti svim podacima
- Princip najmanjih privilegija
- Sustav treba omogućiti brzu promjenu prava pristupa

Nerazdvajanje podataka i naredbi, izvršavanje naredbi iz nepouzdanih izvora

- Miješanje podataka i naredbi može dovesti do ranjivosti na injekciju
- Na niskoj razini može doći do ranjivosti vezanih uz memoriju koje omogućavaju izvršavanje napadačevog koda
- Može se zaštititi korištenjem hardverskih značajki koje odvajaju podatke i naredbe
- Na visokoj razini treba izbjegavati korištenje funkcija koje interpretiraju stringove kao kod (npr. funkcija eval() u pythonu)

Nedostatak validacije podataka

- Svi ulazni podaci moraju biti validirani prije korištenja
- Koristiti centralizirane mehanizme za validaciju (posebni modul za parsiranje i validaciju dokumenata)
- Validacija treba biti ovisna o kontekstu
- Prije validacije podaci se trebaju pretvoriti u kanonski oblik (trebaju se maknuti svi escape znakovi)
- Validacija treba biti provedena na više razina

Nepravilno korištenje kriptografije

- Sustav mora koristiti poznate sigurne biblioteke
- Programeri koji koriste biblioteke moraju znati kako se točno trebaju koristiti
- Kriptografija sustava treba biti centralizirana, jedna sigurna komponenta koju svi koriste
- U većini slučajeva isplati se raditi/posavjetovati sa stručnjakom

Krivo upravljanje povjerljivim podacima

- Prvi korak u upravljanju povjerljvim podacima je klasifikacija podataka u različite razine
- Arhitekt mora biti upoznat s regulativama o različitim vrstama podataka
- Sustav mora biti sposoban izmijeniti politiku zaštite (promjena zakona ili izvora podataka)

Ne uzimanje u obzir različitih korisnika

- Korisnici različitih tehničkih sposobnosti će na različite načine koristiti sustav, neki od njih će biti i napadači
- Sustav treba biti jednostavan za korištenje
- Početna konfiguracija sustava treba biti sigurna, a sigurnosne postavke lako dostupne kako bi ih napredni korisnici mogli prilagoditi po svojim željama
- U slučaju velikog broja dozvola koje korisnik mora prihvatiti ili odabrati treba uzeti u obzir da će korisnik sve prihvatiti bez čitanja

Slijepo vjerovanje i nepravilno korištenje vanjskih komponenti

- Vanjske komponente omogućuju jeftiniji i brži razvoj sustava
- Korištenjem vanjskih komponenti povećava se broj mjesta gdje napadači mogu pokušati kompromitirati sustav
- Prije korištenja trebaju biti validirane i testirane
- Odgovorne osobe moraju pratiti promjene vanjskih komponenti
- Trebaju biti konfigurirane samo one funkcionalnosti koje su potrebne
- Sustav treba omogućiti laku zamjenu ili izbacivanje komponente

Razvoj bez fleksibilnosti na buduće promjene

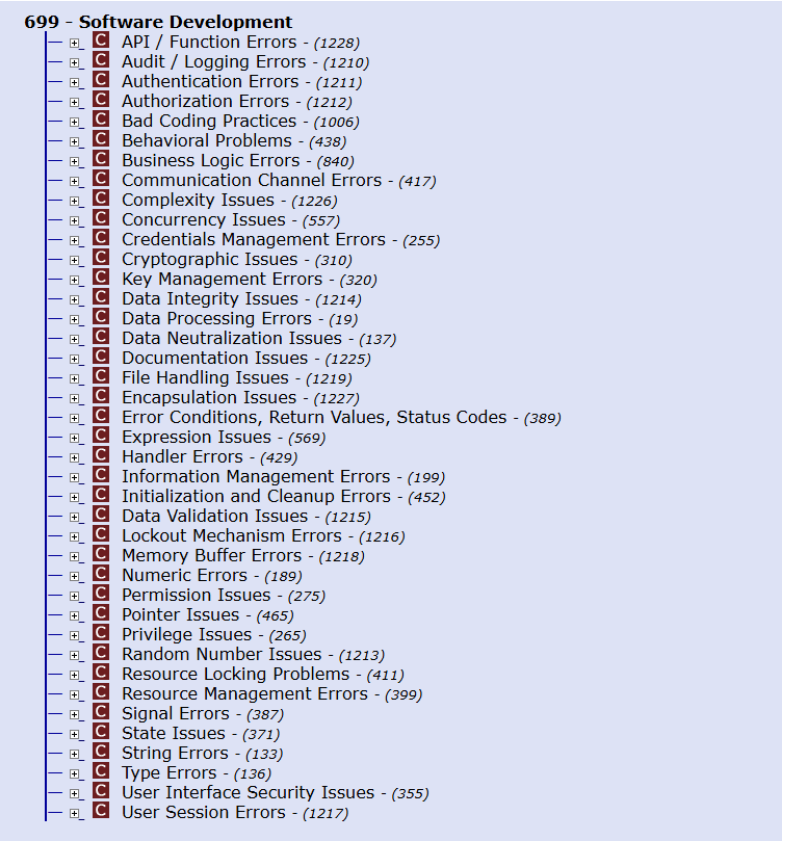
- Nadogradnje manjih dijelova su sigurnije i lakše za održavati nego velike promjene
- Sustav treba biti dizajniran s pretpostavkom da će nove ranjivosti biti pronađene, komponente zamijenjene, broj korisnika i funkcionalnosti rasti, uloge određenih korisnika biti promijenjene
- Tajne se moraju moći promijeniti (lozinke, ključevi)

Metode analize arhitekture

- Analiza arhitekture usmjerena na ranjivosti (VoAA)
- Analiza arhitekture usmjerena na obrasce (PoAA)
- Analiza arhitekture usmjerena na sigurnosne taktike (ToAA)
- Analiza arhitekture za sigurnost (AAFS)

Analiza arhitekture usmjerena na ranjivosti (VoAA)

- Pokušavanje pronalaska grešaka u dizajnu prateći postojeće ranjivosti
- Često se koristi lista Common Weakness Enumeration (CWE)
- Većina ranjivosti u CWE su greške u kodu, a ne u arhitekturi sustava
- Iscrpno pretraživanje neisplativo



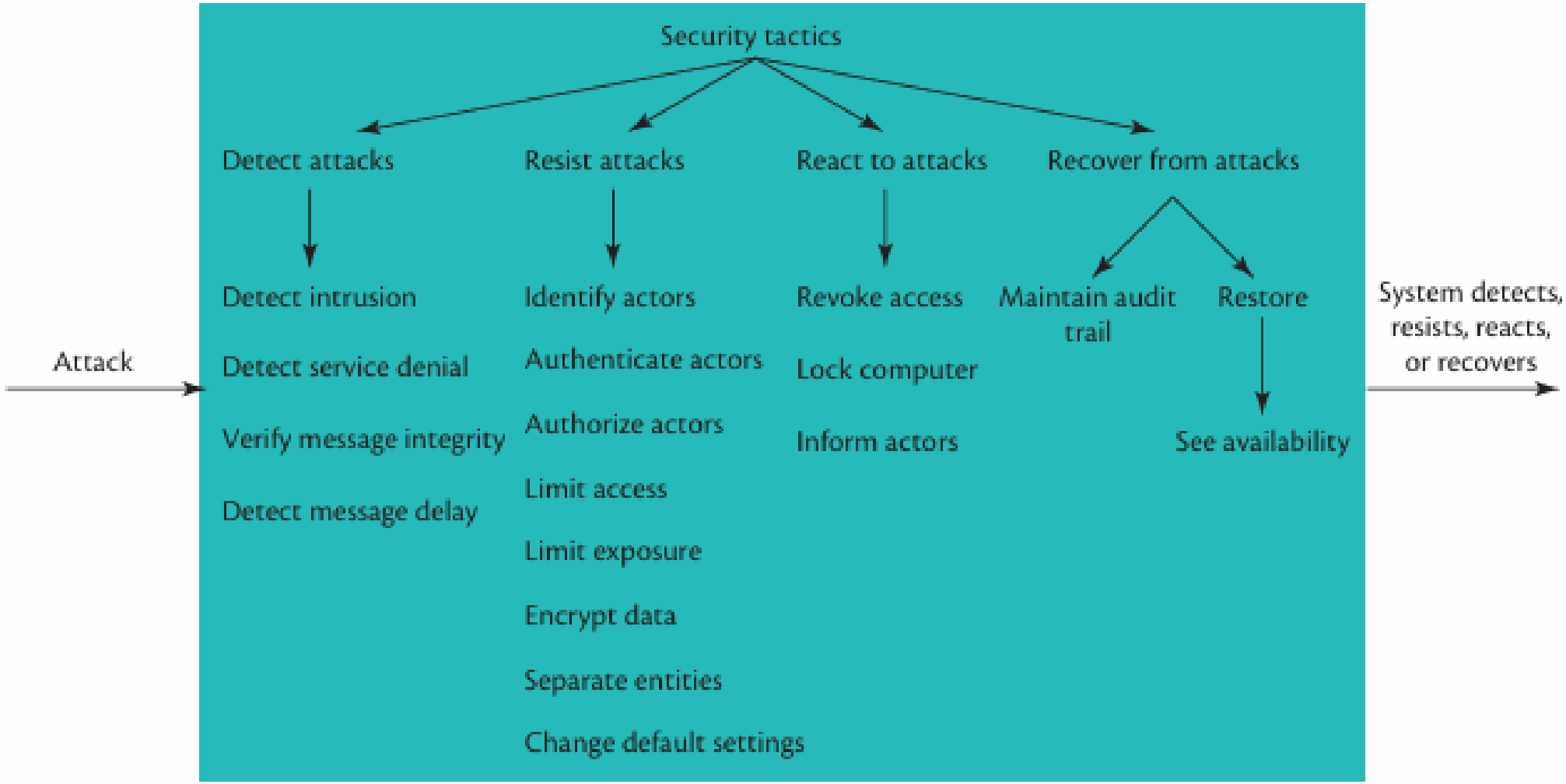
Slika 1. Broj CWE ranjivosti po grupama

Analiza arhitekture usmjerena na obrasce (PoAA)

- Dizajnerski obrasci su rješenja za česte probleme u dizajnu sustava
- Temeljeni su na iskustvu i dokazani u praksi
- Analitičar na temelju dizajnerskih obrazaca analizira arhitekturu
- Puno brži proces nego analiza usmjerena na ranjivosti jer ima puno manje dizajnerskih obrazaca

Analiza arhitekture usmjerena na sigurnosne taktike (ToAA)

- Taktike su temeljni dijelovi arhitekturalnih uzoraka koji predstavljaju realizaciju dizajnerskih namjera (detekcija napada, otpor napada)
- Analitičar na temelju taktika postavlja pitanja arhitektu koji je jako dobro upoznat sa sustavom
- Cilj je u kratkom vremenu otkriti koje dijelove sustava treba popraviti
- Jako apstraktan proces koji možda nije povezan s kodom



Slika 2. Sigurnosne taktike

Analiza arhitekture za sigurnost (AAFS)

- Kombinacija ostalih metoda
- Analizom arhitekture usmjerenom na sigurnosne taktike odredi se koji dijelovi sustava su problematični tj. koje obrasce treba provjeriti
- Analizom arhitekture usmjerenom na obrasce detaljnije se odredi koji obrasci nisu primijenjeni ili nisu dobro napravljeni
- Analitičar na temelju problematičnih obrazaca analizira programski kod i zapravo radi analizu arhitekture usmjerenu na ranjivosti
- Korištenjem sve tri metode efikasno je analiziran programski kod

Zaključak

- Bitno je u samom startu dizajna arhitekture razmišljati o sigurnosti
- Dizajn sigurnog sustava je kompliciran te je potrebno raditi s ekspertima
- Sustav treba biti dizajniran tako da bude fleksibilan na promjene

Literatura

- Arce, Iván, et al. “Avoiding the top 10 software security design flaws.” IEEE Computer Society Center for Secure Design (CSD), Tech. Rep (2014).
- Santos, Joanna CS, Katy Tarrit and Mehdi Mirakhorli. “A catalog of security architecture weaknesses.” 2017 IEEE International Conference on Software Architecture Workshops (ICSAW). IEEE, 2017.
- Ryoo, Jungwoo, Rick Kazman and Priya Anand. “Architectural analysis for security.” IEEE Security & Privacy 13.6 (2015): 52-59
- Common Weaknesses Enumeration (CWE) https://cwe.mitre.org/index.html

Hvala!