VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

VÍCEDIMENZIONÁLNÍ PŘÍSTUP K WWW APLIKACÍM

MUTLI-DIMENSIONAL ACCESS CONTROL IN WEB APPLICATIONS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. PAVOL GREŠŠA
AUTHOR

VEDOUCÍ PRÁCE — Doc. Dr. Ing. KOLÁŘ DUŠAN
SUPERVISOR

BRNO 2011

Abstrakt

Diplomová práce se zabývá analýzou, návrhem a implementací autentizačního a autorizačního subsystému do prostředí webové distribuované aplikace. Unifikuje známe bezpečnostní modely k vytvoření univerzálního bezpečnostního modelu, který slouží k vytvoření autorizačního aparátu schopného zabezpečit aplikace libovolným bezpečnostním modelem. Tuto integraci modelu navíc aplikuje do prostředí systému Takeplace.

Abstract

This master's thesis deals with the analysis, design and implementation of authentication and authorization subsystem into the environment of distributed web application. It unifies the well-known security models into the one universal security model that can be used for the development of authorization device enabling the user to secure the applications with various security models. Furthermore, it applies this integration of models into the Takeplace system.

Klíčová slova

autorizace, autentizace, auditování, správa identit, kontrola přístupu, vícedimenzionální bezpečnostní model, distribuované webové systémy

Keywords

authorization, authentication, auditing, identity management, access control, multi–dimensional security model, distributed web systems

Citace

Pavol Grešša: Vícedimenzionální přístup k WWW aplikacím, diplomová práce, Brno, FIT VUT v Brně, 2011

Vícedimenzionální přístup k WWW aplikacím

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Doc. Dr. Ing. Dušan Koláře

. . . . . .
Pavol Grešša
24. mája 2011

Poděkování

Na tomto mieste by som chcel poďakovať predovšetkým vedúcemu práce docentovi Dušanovi Kolářovi za jeho odborné vedenie. Vďaka patrí aj mojej rodine, ktorá ma počas celého štúdia podporovala a motivovala.

© Pavol Grešša, 2011.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

1 Úvod **4**
1.1 Cieľ práce 4
1.2 Obsah kapitol 5

2 Hrozby v prostredí webových aplikácií **6**
2.1 Open Web Application Security Project (OWASP) 6
2.1.1 A1 – Injection 8
2.1.2 A2 – Cross-Site Scripting (XSS) 8
2.1.3 A3 – Broken Authentication and Session Management 8
2.1.4 A4 – Insecure Direct Object References 9
2.1.5 A5 – Cross-Site Request Forgery (CSRF) 9
2.1.6 A6 – Security Misconfiguration 9
2.1.7 A7 – Insecure Cryptographic Storage 9
2.1.8 A8 – Failure to Restrict URL Access 9
2.1.9 A9 – Insufficient Transport Layer Protection 10
2.1.10 A10 – Unvalidated Redirects and Forwards 10

3 Správa identít (IdM, Identity Management) **11**
3.1 Single Sign–On(SSO) 12
3.2 Federovaná správa identít 13
3.3 Global Identity Management Service 13

4 Autentifikácia **14**
4.1 Správa sedení (Session Tracking) 14

5 Kontrola prístupu a autorizácia **16**
5.1 Bezpečnostný model 16
5.2 Univerzálny bezpečnostný model 17
5.2.1 Vymedzenie dimenzie subjektov (kto) 19
5.2.2 Vymedzenie dimenzie operácií (čo) 19
5.2.3 Vymedzenie dimenzie modalít 19
5.2.4 Vymedzenie dimenzie objektov (s čím) 19
5.2.5 Vymedzenie dimenzie času (kedy) 20

6 Služba Takeplace **21**

7 Existujúce distribuované AAA systémy **22**
7.1 DACS – Distributed Access Control System 22

8 Požiadavky na AAA systém **23**
8.1 Funkčné požiadavky . . . 23
8.1.1 Jadro systému (Core) . . . 23
8.1.2 Autentifikácia . . . 23
8.1.3 Autorizácia . . . 24
8.1.4 Auditing . . . 24
8.2 Nefunkčné požiadavky . . . 24
8.3 Prípady použitia systému . . . 25
8.4 Technológie . . . 26
8.4.1 Metodika výberu technológií . . . 26
8.4.2 Vybrané implementačné technológie . . . 26

9 Návrh **30**
9.1 jD3A . . . 30
9.1.1 Audit . . . 32
9.1.2 Správa sedení . . . 33
9.1.3 Správa subjektov . . . 35
9.1.4 Autentifikácia . . . 36
9.1.5 Autorizácia . . . 39
9.1.6 Konfigurácia . . . 43
9.1.7 Webová podpora . . . 43
9.2 Kontrola prístupu v systéme Takeplace . . . 44
9.2.1 Integrácia jD3A do prostredia služby . . . 45
9.2.2 Integrácia kontroly prístupu . . . 46

10 Implementácia **48**
10.1 Audit . . . 48
10.2 Autentifikácia . . . 49
10.3 Autorizácia . . . 49
10.3.1 Spracovanie anotácií . . . 49
10.3.2 Zabezpečenie zdrojov dostupných cez URL . . . 51
10.3.3 Komunikačná služba . . . 52
10.4 Perzistentná služba HBase . . . 54
10.4.1 DAO sedení . . . 54
10.4.2 DAO subjektov . . . 54
10.5 Kešovací systém EHCache . . . 54
10.6 Maven moduly . . . 55

11 Testovanie **57**
11.1 Programovanie riadené testami (Test Driven Development) . . . 57
11.2 Výkonostné testy (Performance testing) . . . 57
11.2.1 Popis testovacieho prostredia . . . 58
11.2.2 Testovanie autentifikácie . . . 58
11.2.3 Testovanie autorizácie . . . 59

12 Rozvoj **60**

13 Záver **61**

A Diagram tried jD3A **66**

B Komponentový diagram jD3A **67**

C Detailná špecifikácia prípadov použití AAA **68**

D OWASP–Session Management analýza **75**

E OWASP–Authentication analýza **77**

F Konfiguračné nastavenia rámca **80**

G Graf autentifikačného testu **82**

H Graf autorizačného testu **83**

I Obsah CD **84**

Kapitola 1

Úvod

Vývoj internetových služieb, ktoré sú označované ako SaaS[1](ďalej len služby) zaznamenáva v poslednom čase nevídaný rozmach. Môže za to aj nedávna finančná kríza, kedy medzi prvé segmenty, ktorých rozpočty sa rapídne zkresali, boli práve IT oddelenia a vývoj softvéru obecne. Náklady spojené s vývojom informačných systémov na mieru, školením užívateľov či údržbou sú vysoké a dlhodobo neudržateľné.

Globalizácia vtrhla do sveta internetu a priamym dôsledkom prudkého nárastu užívateľov je obrovská záťaž na elementárne funkčnosti služieb a ich poskytovateľov. Technológie a prístupy k vývoju, ktoré bolo možné bez problémov použiť na regionálnej, národnej či kontinentálnej úrovni, sú globálne nepoužiteľné. Medzi hlavné dôvody môžno považovať nepripravenosť technológií poskytovať služby státisícom užívateľov alebo dynamicky reagovať na záťažové špičky.

Následkom toho vznikli nové škálovateľné riešenia a štandardná architektúra služieb sa rozšírila o výkonné kešovacie servery či NoSQL databázy. Služba samotná je realizovaná ako množina niekoľkých subsystémov s jednoznačnou zodpovednosťou a funkčnosťou. Z pohľadu koncového užívateľa sa nič nemení, no systém vďaka tomu nadobudol na róbustnosti. Výpadok jedného subsystému neznamená kritickú nefunkčnosť celého systému.

Služby nie sú ďalej prevádzkované na serveroch poskytovateľa, ale na veľkých cloudových riešeniach označovaných ako IAAS[2]. Dôvodom sú opäť náklady, kde možnosť elasticity, škálovateľnosti, aktualizovateľnosti a prístupnosti výrazne znižuje náklady na prevádzkovanie služby na vlastných serveroch poskytovateľa. To nevylučuje fakt, že sám poskytovateľ môže vytvoriť pre svoju službu vlastný cloud, ktorý je privátny (Facebook) alebo ho ďalej distribuuje ako produkt (Amazon Book Store a Amazon Web Services).

Medzi hlavných predstaviteľov takýchto distribuovaných služieb možno považovať populárnu komunikačnú platformu Facebook či gigant Google. Práve vyššie spomenuté technológie a prístupy často pochádzajú z dielní týchto erudovaných spoločností, ktoré môžeme bez obáv nazvať lídrami v internetových technológiach.

1.1 Cieľ práce

Ako bolo spomenuté, vysoká záťaž predstavuje riziko pre elementárne funkčnosti služby. Medzi ne patrí nepochybne autentifikácia a autorizácia užívateľov. S tým súvisí aj cieľ

[1]Software as a Service – model nasadenia softvéru, kedy dochádza k hostingu aplikácie prevádzkovateľom služby. Služba je dostupná zákazníkom cez internet.

[2]Infrastructure as a Service – infraštruktúra ako služba. Príkladom IAAS sú Amazon WS, Rackspace alebo Windows Azure.

tejto práce, ktorým je definovať požiadavky, analyzovať, navrhnúť a implementovať takýto subsystém v rámci služby Takeplace, a to pri dodržaní najvyššieho štandardu bezpečnosti pri najvyššom možnom počte prihlásených užívateľov.

1.2 Obsah kapitol

Prvá časť práce bude venovaná analýze bezpečnostných rizík a definovaniu základných bezpečnostných požiadaviek na subsystém. Následne bude popísaná architektúra služby Takeplace ako platformy na organizovanie odborných akcií a konferencií. Ďalšia časť práce sa venuje popisu univerzálneho bezpečnostného modelu, ktorý bude v subsystéme implementovaný. Analýzu celého subsystému uzatvorí kapitola zameraná na definovanie funkčných požiadaviek a technológií, ktoré budú použité pri implementácii. Po analýze nasleduje kompletný návrh subsystému rešpektujúci vzťahy v architektúre služby Takeplace. Je na mieste poznamenať, že sa jedná o najkritickejší subsystém celej platformy, a preto bude podrobený výkonnostným testom, ktorých popis a výsledky sú obsahom kapitoly testovania.

Kapitola 2

Hrozby v prostredí webových aplikácií

Cieľom tejto kapitoly je definovať najbežnejšie bezpečnostné hrozby v prostredí informačných systémov a internetových aplikácií. Bezpečnosť v kontexte informačných systémov je opakovaný proces manažmentu rizík počínajúc samotným manažmentom ľudských zdrojov, obchodných procesov, verifikovaného návrhu, vývoja a testovania systému, jeho správy a údržby. K celému procesu bezpečnosti informačných technológií vznikla nezisková organizácia Open Web Application Security Project.

2.1 Open Web Application Security Project (OWASP)

Open Web Application Security Project [29] je otvorená komunita ľudí a spoločností, ktorých zámerom je dokumentovanie, vývoj a distribúcia informácií vzťahujúcich sa k bezpečnosti informačných technológií. Organizácia je nezisková a nezávislá na komerčných záujmoch. Riadi sa etickým kódexom a všetky dokumenty, nástroje, či iné formy informácií, ktoré vzniknú pod hlavičkou OWASP, sú zdarma a voľne šíriteľné či upravovateľné. Filozofiou či cieľom tejto organizácie, ktorej podporovateľmi sú okrem iných aj spoločnosti ako Adobe, Microsoft, HP či Oracle, je neustály proces vzdelávania vývojárov, architektov, dizajnérov a celých organizácií o dôsledkoch nezabezpečených hrozieb. Organizácia má pobočky po celom svete.

Vytvorené nástroje a dokumenty sú riadené formou projektu. Každý projekt je zameraný na úzku a špecifickú problematiku bezpečnosti. V rámci globálneho členenia projektov sú ustanovené tri kategórie:

- *Protect* – Nástroje a dokumenty obsahujúce opatrenia a ochranu voči bezpečnostným hrozbám v návrhu systému či implementácii.
- *Detect* – Obsahuje nástroje a dokumenty, ktoré môžu byť použité k identifikácii či odhaleniu bezpečnostných hrozieb v návrhu systému či implementácii.
- *Life Cycle* – Súbor nástrojov a dokumentov zameraných na integráciu manažmentu bezpečnostných rizík do životného cyklu vývoja softvéru (Software Development Life Cycle).

V tomto čase (2011) je registrovaných 75 projektov. Medzi najznámejšie projekty v rámci OWASP patrí:

Application Security Verification Standard Project

Dokument spadajúci do kategórie *detect*, ktorého účelom je vytvorenie štandardu, ktorý svojím rozsahom a vysokou precíznosťou definuje postup vykonania bezpečnostnej verifikácie internetových aplikácií. Vychádza z existujúcich komerčných či iných voľne šíriteľných štandardov a zo skúseností členov organizácie. Štandard je nezávislý ako na architektúre aplikácie, tak aj na použitých technológiách. Výsledkom verifikácie je úroveň dôvery verifikovaného systému [28].

AntiSamy Project

AntiSamy je nástroj z kategórie *protect*. Konkrétne sa jedná o knižnicu, poskytujúcu API, ktoré je možné využiť na validáciu užívateľských vstupov v HTML formáte. Cieľom je eliminovať hrozbu vloženia škodlivého kódu v jazyku JavaScript. Kontrola vstupu sa vykoná na základe nastavenia konfiguračného súboru, v ktorom sú definované všetky povolené hodnoty [27].

Top Ten Project

Top Ten projekt patrí medzi najznámejšie dokumenty organizácie OWASP. Jedná sa o celosvetový konsenzus bezpečnostných expertov, popisujúcich 10 najkritickejších bezpečnostných hrozieb internetových aplikácií. Práve z tohto dôvodu bude tento dokument z kategórie *detect* podrobený bližšej analýze v ďalšej časti kapitoly. Aktuálna tretia verzia je vydaná pre rok 2010, prvá verzia bola vydaná pre rok 2004, druhá verzia pre rok 2007 [31].

Software Assurance Maturity Model

Software Assurance Maturity Model (SAMM) je otvorený rámec, ktorý má pomôcť organizácii formulovať a implementovať postupy k dosiahnutiu bezpečnosti vzhľadom k hrozbám plynúcim z charakteru vyvíjaného softvéru. Rámec je definovaný s ohľadom na flexibilitu použitia, z čoho plynie, že môže byť nasadený v malých, stredných či veľkých organizáciach, bez ohľadu na používaný postup vývoja. Rovnako môže byť tento model aplikovaný na individuálne projekty, ako aj plošne v celej organizácii [30].

WebScarab Project

WebScarab je desktopová aplikácia určená na analýzu HTTP alebo HTTPS komunikácie. Má vstavaných niekoľko operačných módov, ktoré je možné rozšíriť o množstvo funkčností pomocou pluginov. Vo väčšine módov figuruje WebScarab ako záchytný proxy server, v ktorom umožňuje používateľovi analyzovať a upravovať dáta poslané z prehliadača či cieľového servera [32].

V následujúcich kapitolách budú predstavené najväčšie bezpečnostné hrozby internetových aplikácií, ktoré definuje OWASP Top Ten projekt. Ochrana proti týmto hrozbám patrí medzi hlavné bezpečnostné požiadavky na výsledný autentifikačný a autorizačný subsystém diplomovej práce.

2.1.1 A1 – Injection

Injektáž je technika napadnutia väčšinou databázovej vrstvy aplikácie vsunutím SQL, OS, LDAP reťazca, ktorého interpret služby spracuje ako súčasť príkazu. Týmto spôsobom útočník môže vykonať pôvodne nezamýšľané príkazy či sprístupniť neautorizované dáta. Táto nežiaduca vlastnosť vzniká v prípade úzkeho a nekontrolovaného prepojenia aplikačnej vrstvy aplikácie s databázovou vrstvou.

K predchádzaniu injektáže je nutné oddeliť vstupy, ktoré môžu obsahovať nebezpečné reťazce, od cieľových príkazov. Vstup musí byť skontrolovaný a až potom predaný interpretu služby. Mnohé aplikačné rámce a knižnice majú vstavané tzv. *bezpečné API*, ktoré automaticky kontroluje vstup na kľúčové reťazce interpretu. Bezpečné API je zahrnuté v projekte OWASP ESAPI [38].

2.1.2 A2 – Cross-Site Scripting (XSS)

XSS útoky spočívajú vo vložení HTML $<SCRIPT>$ príkazu do kódu webovej stránky. Následkom toho môže útočník spúšťať skripty na hostiteľských počítačoch. Tým môže nastať únik privátnych dát, úprava vzhľadu webovej stránky, presmerovanie hostiteľského prehliadača na nebezpečné stránky a pod. Existujú tri typy XSS útokov:

- *Odrazené (Reflected)* – útoky vznikajú vložením injektovaného kódu mimo aplikačný server, napríklad v chybových hláškach, výsledkoch vyhľadávaní alebo akejkoľvek inej odpovedi servera na požiadavku, ktorá obsahuje nejaké vstupy.
- *Uložené (Stored)* – injektovaný kód je uložený na servere v databáze aplikácie ako súčasť obsahu, ktorý sa posiela užívateľom.
- *Lokálne (DOM based)* – jedná sa o neošetrené prenesenie premennej z URL adresy do javascriptového kódu.

K ochrane sa používa technika "escaping", ktorá zabezpečí, že celý reťazec je interpretovaný ako cieľový bez potrebnej interpretácie. K tomu sa používajú práve "escape" znaky a sekvencie, ktoré informujú daný interpret, že sa nejedná o príkazy. Ako ochrana je taktiež možná silná validácia vstupov.

2.1.3 A3 – Broken Authentication and Session Management

Autentifikácia a správa sedení užívateľov nie je často správne a bezpečne implementovaná a umožňuje útočníkom odkryť heslá, identifikácie sedení alebo použiť iné exploity[1] na získanie a zneužitie užívateľských identít. Najčastejšími chybami sú hlavne práca s otvorenými heslami užívateľov, odhlasovanie sa zo systému, *remember me* funkčnosť, bezpečnostné otázky či preposlanie zabudnutého hesla na email.

Existujú návrhové a implementačné rámce, ktoré majú vstavaný bezpečný proces autentifikácie a správy sedení, napr. Spring Security [43], OWASP Application Security Verification Standard [28], OWASP ESAPI Authenticator and User APIs [38].

[1] Exploit je špeciálny program alebo sekvencia príkazov, ktorý využíva programátorskú chybu vo svoj prospech.

2.1.4 A4 – Insecure Direct Object References

K nebezpečnému priamemu odkazu na objekt dochádza vtedy, keď vývojár odkryje odkaz na vnútorný objekt, ako je súbor, adresár alebo identifikátor databázového kľúča. Bez kontroly vstupu alebo iného spôsobu ochrany môžu útočníci s týmto objektom manipulovať, aby dosiahli neoprávneného prístupu k dátam.

Objekty by mali byť odkazované cez nepriame referencie a na priamy odkaz budú prevedené až na serverovej strane aplikácie. Nepriame odkazy môžu byť generované pre každého užívateľa v rámci jeho sedenia, prípadne každý objekt bude obsahovať svoj nepriamy náhodný identifikátor. Základná zložka ochrany je kontrola prístupu k objektu, teda overenie, či má prihlásený užívateľ autorizáciu s objektom pracovať.

2.1.5 A5 – Cross-Site Request Forgery (CSRF)

Útok metódou CSFR prinúti prehliadač prihláseného užívateľa zaslať falošnú HTTP žiadosť, vrátane cookie aktívneho sedenia obete a ďalších automaticky zahrnutých autentifikačných informácií na server webovej aplikácie. Útočník vďaka tomu môže generovať žiadosti, ktoré webová aplikácia bude považovať za legitímne, pochádzajúce od napadnutej osoby. Takéto žiadosti sa môžu nachádzať v odkazoch na obrázok a pri načítaní stránky dôjde k legitímnej požiadavke.

Každému formuláru je priradený skrytý nepredvídateľný unikátny token, ktorý sa pošle s požiadavkou na server. Aplikačná logika spáruje token s formulárom, čím overí legitímnosť požiadavky. Dôležité akcie sú posielané ako POST požiadavky. Kritické transakcie (prevod peňazí) by mali byť potvrdené opätovnou autentifikáciou užívateľa.

2.1.6 A6 – Security Misconfiguration

Popisuje problém bezpečnej konfigurácie operačného systému, aplikácií, rámcov, aplikačných serverov, web serverov, databázových serverov a celej platformy. Všetky tieto nastavenia by mali byť definované, implementované a udržiavané, keďže mnohé nie sú dodávané s bezpečnými predvolenými hodnotami. To zahŕňa aktualizovanie softvéru vrátane všetkých knižníc, ktoré aplikácia používa. Základnou požiadavkou je aktuálnosť knižníc a existencia automatického procesu kontroly konfiguračných súborov na ich hodnoty pomocou kontrolného súčtu. Vývojové, testovacie a produktové nastavenie prostredia musí byť rovnaké.

2.1.7 A7 – Insecure Cryptographic Storage

Webové aplikácie nedostatočne chránia citlivé dáta, ako sú čísla kreditných kariet, rodné čísla, identifikačné údaje. Útočníci môžu ukradnúť či modifikovať tieto slabo chránené údaje a uskutočniť tak krádež identity, podvrhnúť kreditné karty alebo uskutočniť iné zločiny.

Všetky citlivé dáta a dôležité údaje musia byť uložené zašifrované, prípadne zahešované. Zálohy dát musia byť rovnako šifrované. Certifikáty a správa kľúčov musí byť vedená na izolovaných úložiskách.

2.1.8 A8 – Failure to Restrict URL Access

Pred tým, než webová aplikácia sprístupní (zobrazí) odkaz na chránenú stránku, overuje práva prístupnosti. Aplikácia musí vykonať túto kontrolu vždy a pre každú požiadavku bez ohľadu na to, či už bola podrobená autorizácii. Útočník môže podvrhnúť URL, aby si skrytú stránku sprístupnil.

Pred zobrazením chránených stránok musí byť vždy vykonaná autorizácia. To prirodzene kladie vyššie nároky na výkon servera. Z tohto dôvodu je dobré, aby autorizačný systém pracoval na princípe rolí a každá rola mala presne určené autorizované stránky.

2.1.9 A9 – Insufficient Transport Layer Protection

Aplikáciam sa často nedarí overiť, šifrovať a chrániť dôvernosť a integritu citlivej sieťovej prevádzky. Využívajú sa slabé šifrovacie algoritmy a certifikáty, ktoré sú staré, neplatné alebo ich nepoužívajú správne. Často sa stáva, že stránky sú dostupné aj bez zabezpečenia, prípadne cookies nemajú nastavený príznak *secure*.

Všetky certifikáty aplikácie musia byť periodicky kontrolované za účelom overenia ich validity. Stránky vyžadujúce SSL prístup musia užívateľa automaticky presmerovať na ich šifrovaný obsah, ak sa ich snaží získať v nešifrovanej podobe. Certifikáty musia používať silné šifrovacie algoritmy [11].

2.1.10 A10 – Unvalidated Redirects and Forwards

Webové aplikácie často presmerúvajú (redirect) a preposielajú (forward) užívateľov na iné stránky a používajú nedôveryhodné udaje pre určenie cieľových stránok. Bez riadnej validácie môžu útočníci presmerovať obete na phishingové a malwarové stránky alebo použiť presmerovanie na neautorizované stránky.

Najlepšou ochranou je to, že aplikácia nebude používať žiadne presmerovania a preposielania. V prípade nevyhnutnosti by cieľová adresa mala byť zostavená až na serveri, odkiaľ sa pošle prehliadaču príkaz na presmerovanie.

Kapitola 3

Správa identít (IdM, Identity Management)

Jedná sa o súbor služieb, ktoré vzhľadom na definovanú bezpečnostnú politiku poskytujú automatické a bezpečné riešenie pre správne úkony. Tie sa vzťahujú k autentifikačnému a autorizačnému procesu, ktorých účelom je chrániť obmedzené a citlivé informačné zdroje (aktíva). Neoddeliteľnou súčasťou procesov je podrobné protokolovanie jednotlivých akcií a podpora auditných nástrojov.

Identita entity vyjadruje unikátnu a jednoznačnú totožnosť subjektu vymedzenú oproti ostatným entitám vzhľadom na daný kontext. Správa identít pozostáva z nástrojov pre definovanie identít entít, bezpečné a efektívne uchovávanie relevantných identitných informácií o entitách, sprístupňovanie týchto informácií pomocou štandardizovaných rozhraní a pre poskytovanie odolnej, distribuovanej a výkonnej infraštruktúry na podporu autentifikačných a autorizačných procesov [18].

Správa identít je jediná autorizovaná autorita, ktorá je zodpovedná za celý životný cyklus identity v systéme, čiže vytvorenie identity a jej počiatočná inicializácia, zmeny vykonávané systémom, delegovaným správcom alebo vlastníkom identity, deaktivácia a zrušenie identity.

Základným cieľom správcu identít je vytvorenie jednoznačnej identity pre každú entitu systému, poskytnutie nástrojov pre jednoduchú a racionálnu definíciu kontextu identity na základe bezpečného a nepopierateľného overenia totožnosti. Tieto ciele sú často symbolicky označované ako štyri–A (4A) [18]:

- *Autentifikácia (Authentication)* – proces overenia totožnosti entity
- *Autorizácia (Authorization)* – proces udelenia práv a rolí
- *Kontrola prístupu (Access Control)* – nástroje na riadenie prístupu
- *Účtovanie a audit (Accounting, Audit)* – účtovanie využívania zdrojov a audit akcií

Medzi základné komponenty systému správy identít patrí [53]:

1. *Autentifikačná služba (Authentication Service)* – autentifikácia na základe poskytnutej identifikácie a dôvernej informácie.
2. *Riadenie prístupu a autorizácia (Access control and Authorization Service)* – riadenie prístupu k zdrojom a aktívam systému a proces overenia prístupu

3. *Správa dokladov (Credentials Management)* – bezpečný proces pokrývajúci zmenu dôverných a tajných informácií na vyžiadanie entity.

4. *Správa účtov (Entity Provisioning)* – obsahuje správu entitných účtov a ich dostupnosť v prostredí systému.

5. *Administrácia identít (Identity Administration)* – globálna správa identít entít pre administrátorské účely

IdM môže obsahovať nadstavbové služby rozširujúce jej použiteľnosť [18], [53]:

1. Samoobslužnosť – absolútna delegácia administrátorských služieb na entitu, t.j. automatická aktivácia účtu, zmena dokladov, zmena práv a obmedzení.

2. Meta–adresár – adresárová služba, ktorá integruje viacero používaných adresárových služieb organizácie do jedného globálneho adresára, t.j. LDAP, X.500, a podobne.

3. Správa a synchronizácia dokladov medzi subsystémami – jednotná správa dokladov a ich dostupnosť v celom systéme, Single Sign–On.

V tomto čase existuje množstvo technológií a štandardov, ktoré sú zamerané na tvorbu IdM. Dôležité technológie a postupy budú popísané v nasledujúcich kapitolách.

1. Štandardizované nástroje: OASIS, WS-I, W3C

2. Adresárové služby: X.500, LDAP, DSML

3. Webové služby: SOAP, WSDL, UDDI

4. Bezpečnostné štandardy: SAML, WS-S

5. Riadenie prístupu: XACML

6. Sigle Sign–On: OpenId

3.1 Single Sign–On(SSO)

Jednotné prihlásenie (Single sign-on) predstavuje jednotnú správu identít pre celý distribuovaný systém. Proces autentifikuje subjekt k prístupu ku všetkým službám a eliminuje tým všetky budúce požiadavky na autentifikáciu, ktoré by boli v rámci aktuálnej relácie vyžiadané v prípade prístupu do iného subsystému.

Existujú protokoly navrhnuté k výmene autentifikačných a autorizačných informácií medzi správcami identít. Medzi najznámejšie patrí:

1. *Security Assertion Markup Language [36]* – otvorený štandard založený na výmene XML správ medzi poskytovateľom služby a poskytovateľom identity. Domény komunikujú typicky na základe SOAP protokolu alebo HTTP (od verzie 2), s využitím asymetrickej kryptografie už od svojej prvej verzie.

2. *OpenID [33]* – decentralizovaný globálny správca identít poskytujúci autentifikáciu užívateľov, ktorí sú už existujúci užívatelia niektorej z partnerských služieb (Facebook, Google, AOL, PayPal, Seznam.cz, a ďalší). OpenID je otvorený štandard, ktorý bol vyvinutý za účelom okamžitého nasadenia a aplikovania do prostredia Internetu.

3.2 Federovaná správa identít

Federovaná správa identít poskytuje prostriedky k autentifikácii a autorizácii interných užívateľov k externým systémom alebo externých užívateľov k interným systémom, bez výmeny dôverných informácií.

To umožňuje vytvoriť dôverný vzťah medzi jednotlivými systémami či celými organizáciami (jurisdikciami). Indentita je federovaná pomocou unikátneho tokenu – prihlasovacie meno – v kontexte niekoľkých správcov identít, ktorí sú prepojení medzi sebou.

Federovaná identita je často považovaná za nástroj k realizácii SSO v už existujúcom heterogénnom prostredí.

3.3 Global Identity Management Service

Globálna správa identít je najvyšší stupeň SSO správy užívateľských identít. Služba poskytuje autentifikáciu užívateľa cez verejné rozhranie. To umožňuje používať jednu digitálnu identitu v celom spektre systémov a služieb.

Organizácie a konzorciá vytvárajú štandardy, ktorých cieľom je definovať prostriedky, obmedzenia a hlavne bezpečnostné požiadavky v rámci globálnej správy identity. Konzorcium W3C identifikovalo tieto požiadavky na služby poskytujúce globálne systémy správy identít takto [51]:

1. *Prenositeľnosť a interoperabilita* – identita musí byť globálne unikátne identifikovateľná vo všetkých aplikáciach, službách či doménach. Formát predávaných správ musí byť prenositeľný a univerzálny s možnosťou rozšírení.

2. *Rozšíriteľnosť* – identita môže obsahovať neobmedzené množstvo atribútov. Musí existovať služba popisujúca jednotlivé atribúty.

3. *Možnosť nastavenia úrovne súkromia a prístupu* – vlastník identity môže nastaviť úroveň súkromia a poskytovania atribútov pre každú doménu osobitne.

4. *Zodpovednosť* – všetci účastníci (užívatelia a organizácie) musia súhlasiť s podmienkami používania automatických prihlasovacích agentov v súlade s právnou zodpovednosťou.

5. *Distribuovateľnosť registračných autorít* – globálny správca identity sa nemôže spoliehať na to, že je jediná registračná autorita. Svojím rozhraním a používanými protokolmi musí umožňiť zmenu registračnej autority a bezproblémový prenos identity.

6. *Distribuovateľnosť certifikačných autorít* – globálny správca identity sa nemôže spoliehať na jednu sieť dôvery a musí umožňiť používať rozšíriteľný protokol a prostredie k uplatneniu viac sietí dôvery. Vlastníci identity sa môžu sami rozhodnúť, akú mieru dôvery použijú pre konkrétnu transakciu.

7. *Nezávislosť vládnucej autority* – globálny správca identít musí byť schopný vystupovať ako medzinárodná nezávislá nezisková organizácia. Nezávislosť musí byť preukazateľná hlavne v kontexte priemyslu, obchodu a štátnej správy.

Kapitola 4

Autentifikácia

Súčasťou kontroly prístupu do informačných systémov je proces overenia totožnosti subjektu. Autentifikácia je vykonaná na základe *identifikácie* subjektu, ktorá jednoznačne vymedzuje entitu a jej profil vzhľadom na kontext systému. Po preukázaní totožnosti poskytnutím dôverných dokladov (credentials), subjekt vystupuje v systéme pod svojou *digitálnou identitou* – elektronický záznam atribútov určujúcich identitu entity (ID, užívateľské meno, rodné číslo, meno, adresa, a pod.).

Existujú tri kategórie (faktory) dôverných informácií, ktoré slúžia k preukázaniu totožnosti na základe toho, že žiadateľ o overenie:

1. *pozná nejaké tajomstvo* – heslo, PIN
2. *niekým je* – biometrické údaje, ako je odtlačok prstu, sietnica
3. *vlastní niečo unikátne* – pas, autentifikačný token

Vzhľadom na typ a počet aplikovaných faktorov použitých pri autentifikácii sa rozlišujú dva stupne bezpečnosti celého procesu, a to:

- *slabá (weak)* – overenie na základe poskytnutia dôverných informácií z jedného faktora *pozná nejaké tajomstvo*, napríklad heslo, prípadne kombinácia heslo–PIN.
- *silná (strong)* – tiež nazývaná ako *multi–faktorová*. Vyžaduje poskytnutie dôverných informácií z dvoch alebo viac faktorov (heslo a token, heslo a odtlačok prstu).

4.1 Správa sedení (Session Tracking)

Vo webových aplikáciach je súčasťou úspešnej autentifikácie vytvorenie relácie medzi serverom a klientom – sedenie (session). Na jeho základe sa identifikuje entita voči serveru vo všetkých prichádzajúcich požiadavkach. Táto potreba vyplýva z charakteru bezstavového protokolu HTTP (HyperText Transfer Protocol), ktorý funguje na princípe: požiadavka – odpoveď. Pri absencii sedenia by sa musel klient pri každej požiadavke opakovane autentifikovať.

Cieľom správy sedení je na základe úspešnej autentifikácie vytvoriť nové sedenie s nepredvídateľným unikátnym identifikátorom aktívnym po obmedzený čas. Pri vysoko rizikových aplikáciach by sa mal identifikátor sedenia automaticky obnovovať. Pri neautorizovanej požiadavke sa musí sedenie zrušiť.

U protokolu HTTP existuje niekoľko metód správy sedení:

- **Skryté polia (Hidden fields)** – do formulára sa vloží skryté pole. Tento spôsob je použiteľný len pre konverzácie obsahujúce nestatické stránky. V prípade výskytu statickej stránky dochádza k strate kontextu sedenia.
  `<INPUT TYPE='hidden' NAME='sessionId' VALUE='12348421484123874154 6'>`

- **Prepisovanie URL (URL Rewriting)** – obsahuje dva spôsoby úpravy URL, do ktoreho vloží identifikátor sedenia. V prípade preposlania URL alebo jej uloženia dochádza k odhaleniu identifikátora a k narušeniu bezpečnosti.

  1. K URL sa pripojí parameter:
     `http://server:port/servlet/name?sessionId=238741546`.
  2. Pred kontextovú časť URL sa vloží identifikátor sedenia:
     `http://server:port/238741546/servlet/name`.

- **Cookies** [23] – najpoužívanejší spôsob správy sedenia. Využíva vlastnosti Cookies HTTP protokolu. Pre sedenie je vytvorená Session Cookie s HttpOnly atribútom, ktorý zabráni napadnutiu pomocou XSS 2.1.2. Tento spôsob nie je možné aplikovať, ak má klientský prehliadač zakázané Cookies.

- **Session API servera** – moderné webové aplikačné servery poskytujú natívne rozhranie, ktoré implementuje kombináciu prepisovania URL a Cookies.

Kapitola 5

Kontrola prístupu a autorizácia

Kapitola popisuje správu a riadenie prístupu (access control) subjektov (procesov, užívateľov) k objektom (dokumenty, entity, akcie) v informačných systémoch. Tento proces udelenia povolenia sa nazýva autorizácia.

5.1 Bezpečnostný model

Bezpečnostný model je formálna definícia povinných bezpečnostných vlastností a zložiek, ktoré musí systém poskytovať. Pozostáva z detailnej špecifikácie, často vyjadrenej v matematickej notácii, z popisu schválených a nepovolených vzťahov medzi subjektmi a objektmi vzhľadom na ich bezpečnostné poverenie a klasifikáciu, z výčtu akcií podliehajúcich auditnej kontrole a zaznamenávaniu.

Bezpečnostný model je schéma, ktorej účelom je špecifikovať a uplatňovať bezpečnostnú politiku. Bezpečnostných modelov riadiacich prístup existuje veľké množstvo [4]:

1. *Mandatory access control (MAC)* – správa prístupu založená na systémovom obmedzení možností subjektu (iniciátora) k prístupu alebo obecnému vykonaniu operácie nad objektom či iným cieľom. Každý subjekt a objekt má bezpečnostné atribúty. Kedykoľvek sa subjekt pokúsi sprístúpniť objekt, aplikuje sa autorizačné pravidlo, kde na základe bezpečnostných atribútov systém rozhodne, či je možné vzhľadom k definovanej bezpečnostnej politike vyhovieť požiadavke a prístup povoliť. Bezpečnostná politika – pravidlá prístupu – je riadená centrálne bezpečnostným administrátorom, ktorý je zodpovedný za správu práv užívateľov.

2. *Discretionary access control (DAC)* – model, ktorý obmedzuje prístup k objektom na základe identity subjektu alebo skupiny, do ktorej subjekt patrí. Voľnosť (discretionary) modelu spočíva v distribuovaní vlastnených práv iným subjektom.

3. *Role based access control (RABC)* – v rámci organizácie sú definované role, reprezentujúce konkrétne pracovné zaradenie a s ním súvisiace zodpovednosti a autorizované akcie. Užívateľovi je priradená rola, ktorá ho opravňuje k prístupu ku všetkým objektom, ktoré sú pre danú rolu pridelené. Pomocou RBAC je možné stimulovať MAC aj DAC.

Bez ohľadu na formu realizácie bezpečnostného modelu (MAC, DAC, RABC, a ďalšie) každý typ pokrýva jednu či viac základných bezpečnostných funkcií:

1. *Dôvernosť* – prevencia proti neautorizovanému odhaleniu informácie

2. *Integrita* – prevencia proti neautorizovanej modifikácii informácie

3. *Dostupnosť* – prevencia proti neautorizovanému odmietnutiu poskytnutia informácie

Problém v heterogénnom prostredí spôsobuje odlišnosť bezpečnostných modelov medzi jednotlivými časťami systému. Napriek tomu, že každý model používa inú terminológiu a iný prístup k popisu problému a jeho riešenia, cieľ je rovnaký. Tým je zamedziť, aby sa systém dostal do stavu, kedy sú dáta zneužité alebo znehodnotené.

Ak je každý subsystém zodpovedný za autorizáciu subjektov, dochádza k degradácii celého systému z dôvodu netransparentnosti vnútorných procesov, fragmentácii zodpovedností a redundancii dát vzťahujúcich sa k digitálnym identitám subjektov. Z týchto dôvodov sa do distribuovaných prostredí nasadzuje jedna autonómna centrálna autorita, ktorá poskytuje autorizačné služby všetkým subsystémom a preberá zodpovednosť za autentifikáciu a autorizáciu entít. Subsystémy požiadavky na autorizáciu buď delegujú, alebo sú zbavené tejto povinnosti a autorizáciu vykoná priamo centrálna autorita.

Unifikovaním bezpečnostných modelov je možné vytvoriť univerzálny bezpečnostný model, ktorý slúži ako jadro autorizačnej časti AAA subsystému Takeplace.

5.2 Univerzálny bezpečnostný model

Účelom každého bezpečnostného modelu je ochrániť dáta aplikovaním definovanej bezpečnostnej politiky. Bezpečnostný model je založený na vyhodnotení vstupných dimenzií vzhľadom k bezpečnostnej politike, ktorú realizuje. Základným vstupom je objekt, subjekt a operácia, ktorú chce subjekt vykonať nad objektom. Bezpečnostný model je definovaný kartézskym súčinom

$$S \times O \times A, \tag{5.1}$$

kde

- S je množina všetkých subjektov
- O je množina všetkých objektov
- A je množina všetkých operácií nad objektmi z množiny O

Tento model je aplikovateľný na systémy založené na vyhodnotení jednoduchých vzťahov medzi subjektom a objektom. Vyhodnotením sa získa buď kladná odpoveď, teda subjekt je autorizovaný k vykonaniu operácie nad objektom, alebo takýto vzťah v modelu neexistuje.

$$\exists s \in S, \exists o \in O, \exists a \in A : soa \in P, \tag{5.2}$$

kde

- s je iniciátor z množiny všetkých subjektov S
- a je operácia z množiny všetkých operácií A, ktoré sú aplikovateľné na objekt
- o z množiny všetkých objektov O
- P je bezpečnostný model definovaný politikou v 5.1

Tento bezpečnostný model vyjadruje vzťah pre doménu prítomného času. Pridaním dimenzie času sa rozšíri aplikovateľnosť modelu.

Definovaný jednoduchý bezpečnostný model vyhodnotí existenciu resp. neexistenciu práva subjektu aplikovať operáciu na objekt. Existencia resp. neexistencia vyjadruje tzv. modalitu vzťahu subjektu k právu. Vyhodnotenie existuje resp. neexistuje vyjadruje vzťah, že subjekt môže resp. nemôže vykonať operáciu nad objektom.

Modalita vyjadruje vzťah subjektu k právu. Každý bezpečnostný model definuje vlastnú modalitu vzhľadom na aplikovanú bezpečnostnú politiku. Mandatórny bezpečnostný model (MAC) používa modalitu "má vlastnosť". Modalita voľného bepečnostného modelu (DAC) je "je identity" či "je v skupine". Model založený na rolách (RBAC) analogicky "je v role". Účelom Business Process Modelingu [19] je štruktúrovaný a detailný popis procesov v rámci organizácie a určenie zodpovedností. Výsledok BMP môže slúžiť ako základ bezpečnostnej politiky, v ktorej modalita vyjadruje vzťah subjektov k procesu, alebo jeho časti, čím identifikuje právo. Každý proces identifikuje modalitu osôb vzhľadom k operácii procesu. Príklad modalít u BMP:

1. *vedúci pracovník segmentu A "je povinný" kontrolovať výkazy práce* – vedúci pracovník má autorizáciu k zobrazeniu výkazov práce segmentu A

2. *pracovník "je zodpovedný" za dokumentáciu spravovaných produktov* – pracovník je autorizovaný k úpravám dokumentov, ktoré spravuje

3. a iné [19]

Univerzálny bezpečnostný model je definovaný kartézskym súčinom piatich dimenzií

$$S \times O \times A \times M \times T, \tag{5.3}$$

kde

- S je množina všetkých subjektov a určuje dimenziu KTO
- O je množina všetkých objektov a určuje dimenziu S ČÍM
- A je množina všetkých operácií nad objektmi a určuje dimenziu ČO
- M je množina všetkých modalít a určuje dimenziu MODALITA
- T je množina všetkých časových údajov a určuje dimenziu KEDY

Univerzálny bezpečnostný model pracuje s dotazmi v rámci celej päťdimenzie:

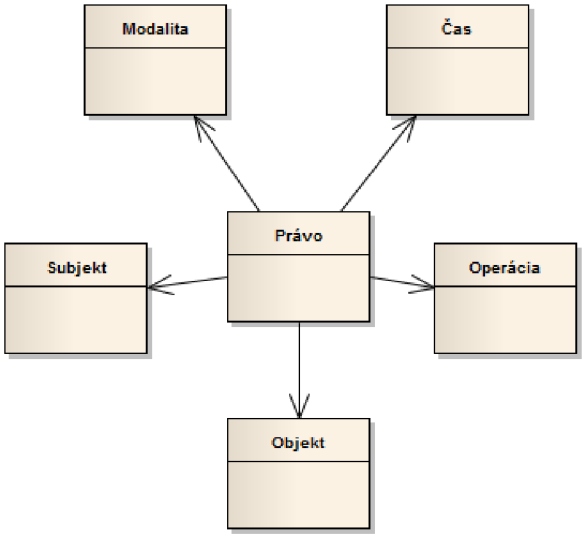
KTO ČO MODALITA S ČÍM KEDY

Príklad:

> "Môže (MODALITA) daný aktér (KTO) vykonať operáciu (ČO) na daný objekt (S ČÍM) v aktuálnom čase (KEDY)?"

Jadrom univerzálneho bezpečnostného modelu je zoznam pätíc (KTO, ČO, MODALITA, S ČÍM, KEDY). Zoznam pätíc definuje *dátovú (štruktúrnu) časť modelu* [42]. Diagram 5.1 konceptuálne znázorňuje vyjadrenie práva na základe jadra modelu.

K štruktúrnej časti modelu je pripojený proces na vyhodnocovanie dotazov. Tento proces sa nazýva *operačná časť modelu* [42]. Zložitosť operačnej časti je priamo úmerná mohutnosti množiny modalít.



Obr. 5.1: Konceptuálny diagram jadra univerzálneho bezpečnostného modelu

Spôsob vymedzenia prvkov z jednotlivých dimenzií je závislé na charaktere aplikácie. Väčšina aplikácií vystačí s použitím definície bezpečnostnej politiky v 5.3 s jednou modalitou "môže" resp. "nemôže". Modalita "nemôže" sa dá eliminovať pravidlom: "všetko čo nie je povolené je zakázané".

5.2.1 Vymedzenie dimenzie subjektov (kto)

Dimenzia subjektov je vymedzovaná užívateľom. U informačných systémov založených na rolách sú subjekty vymedzované tiež rolami. Subjekty sú vymedzované aj samostatnými procesmi, ktoré operujú automaticky v rámci systému.

5.2.2 Vymedzenie dimenzie operácií (čo)

Model je možné napevno zviazať so základnými operáciami manipulácie s dátami – READ, WRITE, DELETE. Dimenzia môže byť vymedzená aj operáciami vyššej úrovne, napríklad publikovanie článku, schválenie dokumentu [42].

5.2.3 Vymedzenie dimenzie modalít

Modalita je zviazaná s použitým bezpečnostným modelom. Často sa jedná o modalitu "môže" resp. "nemôže". K odhaleniu modalít sa použije Business Process Modeling [19]. Vymedzenie potom predstavuje jednoduché ukázanie na zodpovednosť v rámci procesu.

5.2.4 Vymedzenie dimenzie objektov (s čím)

Dimenzia objektov je vymedzovaná vzhľadom na charakter aplikácie a identifikácie aktív systému. Základná vymedzovacia zložka sú objekty systému podliehajúce nejakému obchodnému procesu. Aplikovaním abstrakcie nad základnými objektmi sa vymedzujú skupiny objektov s nejakou spoločnou vlastnosťou [42]:

1. *typ objektu* – všetky dokumenty

2. *účelovo zoskupené objekty* – rozpracované dokumenty
3. *viazané objekty k inému objektu* – všetky dokumenty k projektu A
4. *účelovo vytvorené kombinácie 1, 2, 3* – všetky rozpracované dokumenty k projektu A

5.2.5 Vymedzenie dimenzie času (kedy)

Základným vymedzovacím nástrojom je časový interval OD–DO. Je možné vymedzenie na základe periodicky sa opakujúcej formy času, napríklad každý 2. deň v mesiaci. Sofistikovanejšie vymedzenia vzhľadom k role dimenzie KTO, napríklad prístup člena tímu A k dokumentom projektu B po dobu trvania projektu.

Kapitola 6

Služba Takeplace

Systém Takeplace [1] je webová služba navrhnutá k efektívnemu organizovaniu akcií založených na stretávaní, zdieľaní, komunikácii a tvorbe odborných komunít. Takeplace funkčne pokrýva organizačné procesy v životnom cykle odborných akcií, akými sú semináre, školenia, kongresy, sympóziá, akademické konferencie a iné.

Takeplace umožňuje definovať role organizačných a programových zložiek (výborov) spoločností, hodnotiteľov, asistentov a pod. Je možné stanoviť míľniky – termíny pre registráciu účastníkov, zaplatenie poplatkov, poslanie a ohodnotenie príspevkov, atď. Takeplace zaisťuje automatické upozorňovanie účastníkov a organizátorov na termíny či rôzne činnosti vyžadujúce ich pozornosť – blížiaci sa koniec termínu, nedodanie výstupu od zainteresovaných osôb alebo prekročenie kvót, čím celú správu a organizáciu akcie robí jednoduchšou. K dispozícii je niekoľko variantov upozornení (email, SMS, syntetizovaná automatická hlasová notifikácia). Podané príspevky môžu podliehať hodnoteniu; z vybraných príspevkov je potom zostavený program akcie. Program je možné vytvoriť vo viacerých paralelných sekciách intuitívnou formou.

Historická kontinuita a previazanosť ročníkov a sérií odborných akcií dodržuje jednotné názvoslovie či formu. Všetky akcie poriadané v rámci systému sú archivované a dostupné užívateľom natrvalo. Systém se vyznačuje jednoduchým a intuitívnym grafickým užívateľským rozhraním.

Užívatelia môžu zdieľať skúsenosti z účasti, hodnotiť akcie, sledovať svojich kolegov alebo akcie, ktoré ich zaujímajú, a doporučovať ich ostatným. Dôsledkom je maximálna podpora vo vytváraní virtuálnej komunity pozostávajúcej z odbornej, akademickej, vedeckej či laickej verejnosti.

Kapitola 7

Existujúce distribuované AAA systémy

Potreba autentifikovať a autorizovať entity v distribuovanom prostredí nie je nová. Obsahom kapitoly je rešerš existujúcich AAA systémov, ich výhody a nevýhody vzhľadom na vyššie popísané potreby.

7.1 DACS – Distributed Access Control System

DACS je rozšírený nástroj na autentifikáciu a autorizáciu subjektov v distribuovanom prostredí. Autentifikačný aparát podporuje veľa štandardov a federované identity. Autorizačná časť zaostáva hlavne vo výrazových možnostiach.

Výhody:

- voľne šíriteľný – OpenSource
- autentifikácia podporuje najpoužívanejšie adresárové služby (X.509, LDAP, MS Active Directory, CAS, PAM, atď.)
- priama integrácia do Apache Web Server cez zásuvný modul
- podpora federovaných identít a jurisdikcií
- out-of-box – pripravený na okamžité nasadenie
- vstavaný aplikačný firewall

Nevýhody:

- autorizácia len zdrojov dostupných cez URL – problém pri RPC volaní nad jednou URL
- podporuje len role, skupiny a identity
- role a identity sú realizované ako rozšírené skupiny
- role a skupiny musia byť známe vo všetkých jurisdikciách, neexistuje synchronizačný nástroj
- zložitá a neprehľadná konfigurácia v XML
- neexistencia použiteľných auditných záznamov

Kapitola 8

Požiadavky na AAA systém

Cieľom kapitoly je definovať a detailne popísať funkčné, nefunkčné a technologické požiadavky na výsledný AAA subsystém. Požiadavky sú špecifikované pre každú komponentu správy identít osobitne.

8.1 Funkčné požiadavky

8.1.1 Jadro systému (Core)

Obsahuje nástroje na správu subjektov a sedení:

- základné CRUD[1] operácie pre správu subjektov
- základné CRUD operácie pre správu sedení
- transparentný prístup k subjektom a sedeniam bez závislosti na použitej perzistentnej vrstve
- procesy v celom systéme riadené na základe výnimiek

8.1.2 Autentifikácia

Obsahuje požiadavky na autentifikáciu a jej konfiguráciu:

- možnosť definovať prístupové body (entry point) a ich metódu autentifikácie či ďalších požadovaných vlastností a obmedzení. Príklad: prístupový bod `/oauth/login`, vyžadované SSL, len POST požiadavky.
- konfigurácia autentifikácie nezávislá na použitom spôsobe definovania. Príklad: .properties, XML, SpringFramework Bean.
- autentifikácia užívateľov vzhľadom k politike nastavenej v prístupovom bode. Príklad: OAuth, WWW-Authentication, username-password.
- možnosť definovať stratégiu správy sedení k použitému prístupovému bodu. Príklad: Cookies, url rewrite.
- základné bezpečnostné opatrenia definované v OWASP SAML 2.1

[1] Anglická skratka pre základné operácie create, read, update, delete

- bezpečná implementácia zmeny dôverných informácií – zmena možná len na veľmi krátku dobu.
- jednoduchý a priamočiary proces s neinvazívnymi možnosťami rozšírení

8.1.3 Autorizácia

Požiadavky na distribuovanú kontrolu prístupu k zdrojom subsystémov:

- jednoduchá a jednotná správa autorizačných pravidiel, ich definovanie a možnosti rozšírenia
- kontrola prístupu na základe univerzálneho bezpečnostného modelu
- nástroje ku kontrole prístupu k java metódam a zdrojom dostupných cez URL
- nástroje ku kontrole prístupu simulujúce bezpečnostné modely MAC, DAC a RBAC
- transparentná definícia komunikačného rozhrania
- základné komunikačné rozhranie pomocou WebServices
- jednoduchý proces s neinvazívnymi možnosťami rozšírenia
- integrácia do subsystému musí byť čo najmenej invazívna – využitie AOP

8.1.4 Auditing

Požiadavky na auditing a jej možnosti spracovania a konfigurácie:

- každá akcia v procese autorizácie, autentifikácie, správy sedení a správy subjektov generuje auditný záznam
- konfigurovateľné spracovanie auditných záznamov – spracovať, nespracovať
- jednotný formát auditného záznamu prispôsobeného k ďalšiemu spracovaniu
- možnosť definovať spôsob spracovania auditného záznamu

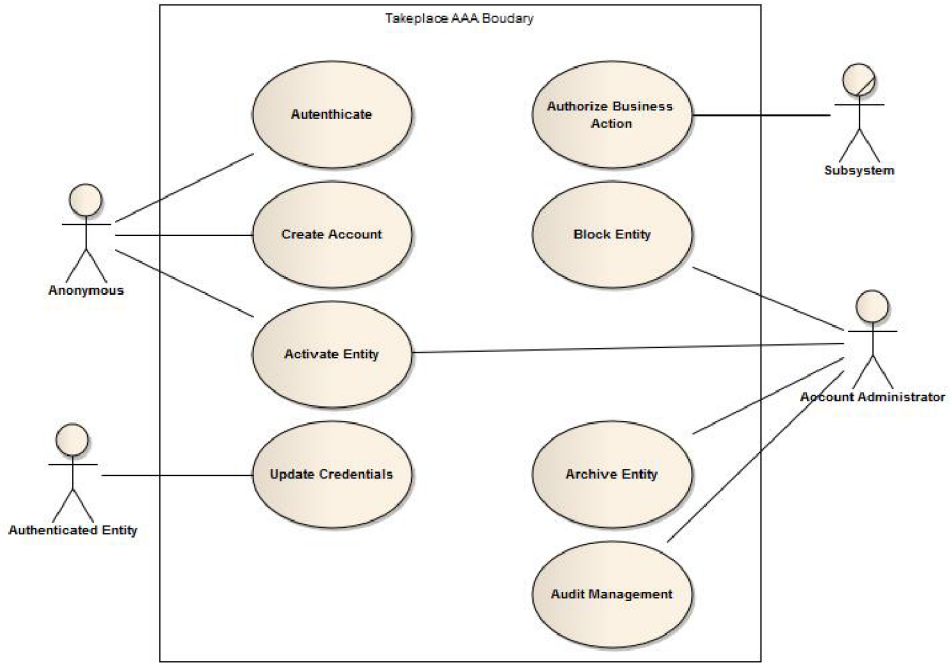
8.2 Nefunkčné požiadavky

- *výkonnosť* – základnou jednotkou merania výkonnosti je počet autentifikácií a autorizácií za sekundu. Ďalšou jednotkou výkonnosti je čas potrebný na vykonanie autentifikácie a autorizácie.
- *škálovateľnosť* – systém musí používať škálovateľné technológie, služby a sám musí mať vlastnosť škálovateľnosti.
- *bezpečnosť* – systém musí byť implementovaný s ohľadom na bezpečnostné hrozby rozoberané v 2.1.
- *kešovanie* – systém musí využívať kešovacie systémy v maximálnej možnej miere. Prístup ku kešovaciemu systému musí byť transparentný a voliteľný.

8.3 Prípady použitia systému

UseCase diagram na obrázku 8.1 definuje prípady použitia nad AAA subsystémom. V systéme vystupuje niekoľko aktérov:

- *Anonym (Anonymous)* – entita ktorá je neautentifikovaná voči systému
- *Autentifikovaná entita (Authenticated Entity)* – autentifikovaná entita, ktorá je identifikovateľná v správe sedení
- *Subsystém* – akýkoľvek subsystém heterogénneho prostredia
- *Správca (Administrator)* – autentifikovaná entita autorizovaná k správe účtov a auditných záznamov



Obr. 8.1: Jadrová komponenta rámca s vymedzenými subkomponentami

Detailnú špecifikáciu prípadov použití môže čitateľ nájsť v prílohe C. Krátka charakteristika prípadov použití:

- *Autentifikácia (Authentication)* – autentifikácia entity voči systému
- *Vytvorenie účtu (Create Account)* – počiatočné vytvorenie neaktívneho účtu v systéme
- *Aktivácia entity (Activate Entity)* – overenie existencie entity a aktivácia digitálnej identity
- *Zmena dokladov (Update Credentials)* – samoobslužná zmena dôverných dokladov v systéme
- *Autorizácia (Authorize Business Action)* – autorizácia požiadaviek zo subsystémov

- *Blokovanie entity (Block Entity)* – zakázanie prístupu entite do systému
- *Archivácia entity (Archive Entity)* – archivácia entity a jej deaktivácia
- *Správa auditných záznamov (Audit Management)* – zložený prípad použití obalujúci správu auditných záznamov pozostávajúci z filtrovania a čítania.

8.4 Technológie

Kapitola sa zameriava na technologické aspekty používané v distribuovaných systémoch. Zlý výber technológií má za následok postupnú degradáciu systému. Každý produkt, bez ohľadu na odvetvie, je tak dobrý a kvalitný ako sú kvalitné technológie, metodiky a postupy použité pri jeho výrobe.

8.4.1 Metodika výberu technológií

Metodika výberu technológií je systematický proces, ktorého cieľom je nájsť – prípadne navrhnúť a implementovať – technológie určené k riešeniu zadaného problému. Technológie sú vyberané na základe niekoľkých kritérií:

1. *Cena* – technológia musí byť cenovo dostupná a atraktívna z dlhodobého hľadiska
2. *Vyspelosť* – kvalita implementácie, jej dokumentácia a rešpektovanie štandardov
3. *Know–how dodávateľa* – skúsenosti a znalosti s využívaním technológie dodávateľom
4. *Škálovateľnosť* – možnosť jednoducho škálovať kapacitu a výkon
5. *Životnosť* – miera udržiavateľnosti a rozvoja technológie do budúcna

Použité technológie musia ďalej rešpektovať metodiku systému Takeplace. Metodika sa nezameriava na uvedené faktory, ale na použiteľnosť a preukázateľnú kvalitu vo forme referencií. Metodika služby Takeplace:

1. *Think Big = Think Scale* – Každá technológia musí obsahovať nástroje, ktoré umožnia škálovať záťaž za použitia čo najmenej prostriedkov. Rovnako musí byť možné u každej technológie jasne identifikovať maximálnu možnú záťaž pre aktuálnu konfiguráciu, a tým stanoviť čas, kedy je potreba škálovať.
2. *Always prefer Open Source* – Medzi hlavné benefity patrí úspora nákladov, nezávislosť na dodávateľovi, možnosť modifikácie, rapídny vývoj a pokrok, ktorý je charakteristický práve pre voľne šíriteľný softvér.
3. *Look at what big guys are doing* – Súčasťou metodiky je neustála snaha o získavanie informácií a skúmanie technológií súčasných lídrov internetových služieb ako Facebook, Google, Twitter, YouTube a podobne.

8.4.2 Vybrané implementačné technológie

Technológie použité v implementovanom AAA subsystéme podliehajú definovanej metodike 8.4.1 a požiadavkám v 8.1, 8.2.

Platforma systému

Systém je postavený na objektovo orientovanej platforme Java. Konkrétne Java Enterprise Edition [34], ktorá je určená na vývoj a prevádzku róbustných podnikových aplikácií a informačných systémov. Platforma sa skladá z komponent, kde každá predstavuje neoddeliteľnú súčasť platformy v procese spracovania zdrojového kódu.

- *Java Development Kit* – obsahuje nástroje a knižnice potrebné k vývoju aplikácií na platforme Java.
- *Java Compiler* – je súčasťou JDK. Plní štandardnú úlohu prekladača jazyka (syntaktická a sémantická analýza). Vstupom je zdrojový kód vo forme `.java` súborov. Výstupom prekladača je prenositeľný byte–kód pre každú triedu zdrojového súbora `.class`. Prekladač postráda optimalizáciu, ktorá je prenechaná na platforme behového prostredia (JRE).
- *Java Runtime Enviroment* – je súborom nástrojov a dátových štruktúr určených k spusteniu a behu aplikácie. JRE je závislé na konkrétnej platforme. Byte–kód je spracovaný virtuálnym strojom (Java Virtual Machine), ktorý vykonáva optimalizáciu kódu za behu (Just–In–Time Compiler).

Komunikačné rozhranie

Technológie komunikačného rozhrania rešpektujú použité protokoly v systéme Takeplace. Prenos informácií v distribuovaných systémoch je založený na využívaní bezpečných a optimalizovaných protokoloch. Mimo štandardnej komunikácie medzi užívateľom a službou, významne vystupuje komunikácia medzi subsystémami. Použité komunikačné protokoly pre jednotlivé schémy:

- *subsystém – subsystém* – medzisystémová komunikácia je realizovaná pomocou webových služieb protokolom SOAP [52], využívajúcim zasielanie správ vo formáte XML.
- *užívateľ – subsystém* – štandardné HTTP s rozšírením o nízkokapacitné protokoly využívajúce javascript klientského prehliadača (GWT RPC [15], JSON [7]).

Kešovacie systémy

Keš je výkonovo kritická služba v rámci navrhovaného subsystému. Účelom kešovacích systémov je optimalizovaná správa dát, často v podobe kľúč–hodnota v operačnej pamäti, a efektívny prístup k týmto dátam.

Keše majú vnútorne implementované kešovacie algoritmy, ktorých cieľom je pri naplnení dostupnej pamäte určiť, ktoré dáta budú z keše odstránené. Moderné kešovacie systémy obsahujú upravené algoritmy vzhľadom k optimalizáciam, ktoré sú charakteristické pre daný systém. Najpoužívanejším bázovým algoritmom je Least Recently Used(LRU) s rozšírením o timeout.

Ku kešovacím systémom k platforme Java vznikla špecifikácia JSR107–JCACHE API [26], ktorej účelom bolo definovať jednotné rozhranie k prístupu do kešovacích systémov. Na tomto štandarde sa prestalo pracovať v roku 2005 a v druhom kvartále roku 2011 boli práce obnovené. V tomto čase nie je štandard dokončený. Z tohto dôvodu nie je možné aplikovať tento štandard do výsledného subsystému.

Použitie kešovacieho systému je závislé od umiestnenia a vzťahu systému a keše. Existuje mnoho implementácií kešovacích systémov. V základe sa delia na:

- *lokálne v rámci jednej JVM* – využívajú výhody JVM a sú vstavané priamo do aplikácie. Kešovaný objekt nie je serializovaný[2] pri vložení resp. načítaní z keše. Najpoužívanejšie sú EHCache [46] a JBoss Cache [21].

- *distribuované* – samostatné serverové aplikácie dimenzované na vysokú záťaž a objem kešovaných dát. Dáta sú serializované a dostupné množstvom komunikačných protokolov. Medzi lídrov distribuovaných keší patrí Memcached [14]. Implementuje obrovskú tabuľku (Google Big Table), ktorá je distribuovaná na niekoľko serverov do ich operačnej pamäte. Rovnako JBoss Cache a EHCache môžu operovať ako distribuované keše.

Pri implementácii bude ako referenčná keš využitá EHCache z dôvodu jej podporovateľnosti a nízkym nárokom na zdroje. Návrh aplikácie musí umožňovať zmenu distribúcie keše.

Perzistentné úložište

Subjekty, sedenia a auditné záznamy musia byť bezpečne perzistentne uložené. Výber báze riadenia dát v distribuovaných systémoch sa odvíja od charakteru aplikácie a predpokladanom počte entít, ktoré budú vstupovať a vykonávať akcie v systéme.

U podnikových aplikácií, kde predpokladaný počet entít nepresahuje rádovo desať tisíce, sa používajú relačné báze riadenia dát. U sociálnych sietí či globálnych služieb sa nasadzujú tzv. NoSQL databáze. Rozdiel je výrazný hlavne z pohľadu tvorby logického schématu, prístupu a tvorby dotazov. V nasledujúcej časti budú podrobne popísané výhody a nevýhody jednotlivých riešení.

Relačná databáza Relačná báza riadenia dát je založená na relačnom modele definovanom Edgarom Coddom v roku 1970 [5]. Táto báza riadenia dát je najrozšírenejším riešením v prípade perzistovania informácií. V súčasnosti sa relačné databáze rozširujú o podporu perzistencie objektov a vzťahov medzi nimi, grafov s podporou hľadacích algoritmov, priestorových dát, multimédií či štrukturovaných textov ako XML.

Existuje množstvo implementácií, kde vedúcim predstaviteľom je firma Oracle poskytujúca rovnomernú komerčnú databázu [35]. Lídrom nekomerčných databáz je open source riešenie PostgreSQL.

Relačné databázy poskytujú róbustnú a mohutnú platformu obsahujúcu množstvo integritných obmedzení, podporu vstavaných triggerov a procedúr.

NoSQL databáza Tento typ riadenia báze dát začal byť populárny v roku 2009 v súvislosti s masívnym nárastom sociálnych sietí a ich vysokými nárokmi na poskytované služby. Charakteristickými rysmi sú: dynamická schéma uloženia dát (tabuľky, stĺpce); nepodporovanie spájania tabuliek (join); typicky horizontálne škálované; jednoducho replikovateľné určené k správe biliónov riadkov a stĺpcov. Dáta sú ukladané štrukturované v distribuovanej multidimenzionálnej mape a dostupné cez jednoduché API. Typickou vlastnosťou je postrádanie podpory ACID[3], ktorá tvorí základ relačných databáz. Množstvo riešení (HBase, Cassandra) má zavedenú podporu ACID ako nadstavbu základnej funkčnej vrstvy [3].

[2] Serializácia je prevod objektu do sekvencie bytov a naopak

[3] Vlastnosti relačných databáz zaisťujúce atomicitu, konzistenciu, izoláciu a trvácnosť (Atomicity, Consistency, Isolation, Durability)

Implementácie NoSQL databáz sú kategorizované podľa štruktúry uloženia dát: dokumentovo zamerané (CouchDB), grafové báze (Neo4j), kľúč–hodnota (HBase,Cassandra). Ďalej bude venovaná pozornosť len kategórii kľúč–hodnota.

Spoločnou vlastnosťou NoSQL a relačných databáz je základné logické oddelenie tabuľkami. Každá operácia nad jedným riadkom je atomická bez ohľadu na počet operácií čítania alebo zapisovania. Primárny kľúč riadku môže byť neobmedzený reťazec, avšak typicky 16 až 36 bytov dlhý. Hodnota riadku v tabuľke je komplexne štrukturovaná. Stĺpce sú zoskupené do množín nazývaných *column–family*. Tie môžu obsahovať ďalšie stĺpce. Zanorenie je obmedzené v závislosti na konkrétnom riešení: HBase 2 úrovne, Cassandra 3 úrovne [3].

Subsystém musí byť nezávislý na použitom riešení. K referenčnej implementácii sa použije NoSQL databáza HBase. Riešenie HBase nedisponuje výrazovými možnosťami relačných databáz (SQL jazyk). Špecifikom AAA subsystému určeného do distribuovaného prostredia (správy subjektov a sedení) nie je potreba realizovať relačný model. Operácie využívajúce zložité mechanizmy relačných databáz sú pre distribuovaný systém, pracujúci s veľkým množstvom štruktúr (subjektov a sedení), nezaujímavé a nepotrebné. Dôraz je v prípade distribuovaného systému, pracujúceho s miliónmi entít, kladený hlavne na rýchlosť, výkon a škálovateľnosť. Operácie sú jednoduché a presne definované.

Správa projektu

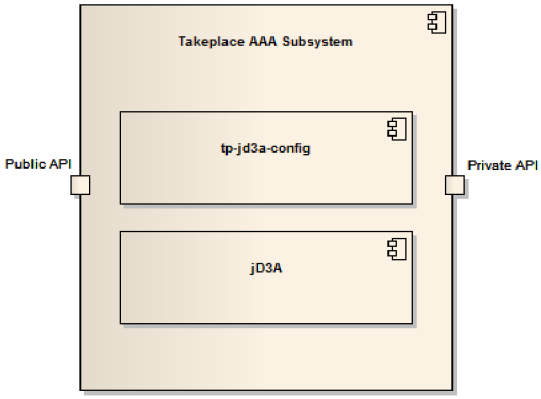
Správa projektu je realizovaná nástrojom Maven [48]. Maven používa deklaratívnu definíciu projektu. Poskytuje nástroje k automatickej správe knižníc, pokrýva celý životný cyklus projektu od jeho inicializácie, vystavania (build), otestovania a nasadenia (deploy) do cieľového prostredia. Životný cyklus je možné ovplyvniť pomocou širokej palety rozširujúcich modulov (pluginov). Významná výhoda Maven–u je nezávislosť projektu na použitom vývojovom prostredí (IDE, Integrated development environment) alebo platforme operačného systému. Súčasťou sú nástroje zamerané na kvalitu kódu a analýzu pokrytia kódu testami.

Kapitola 9

Návrh

Táto kapitola popisuje podrobný návrh celého subsystému na kontrolu prístupu použitím univerzálneho bezpečnostného modelu, implementujúci požiadavky definované v 8 a bezpečnostné opatrenia a doporučenia v 2.1.

Vzhľadom na rešerš existujúcich riešení v 7 a jej negatívny výsledok je vytvorený vlastný rámec, ktorý rešpektuje všetky definované požiadavky a na ktorom bude vystavaný cieľový subsystém. Tento rámec, pomenovaný jD3A – Java Distributed AAA, bude distribuovaný ako Open Source riešenie do heterogénnych webových prostredí.



Obr. 9.1: Integrácia rámca jD3A ako cieľový AAA subsystém Takeplace

Konceptuálne je subsystém riešený spôsobom naznačeným v 9.1, teda poskytnutím konfigurácie k rámcu. V nasledujúcich kapitolách bude nástroj podrobne predstavený, rovnako ako jeho integrácia do subsystému Takeplace.

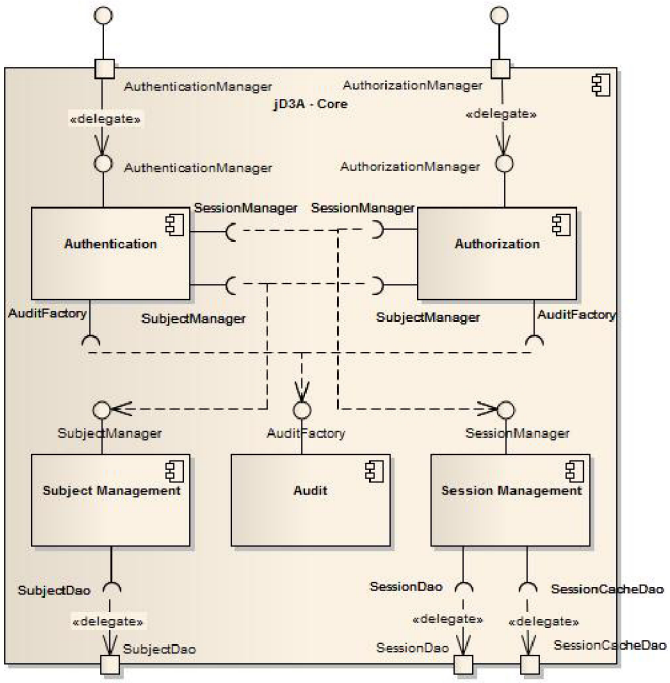
9.1 jD3A

jD3A je názov implementačného rámca obsahujúceho nástroje a realizujúceho procesy potrebné ku kontrole prístupu v distribuovanom prostredí webových aplikácií. Rámec je implementovaný nad univerzálnym bezpečnostným modelom rešpektujúc požiadavky na systém definované v 8.

Návrh rámca je inšpirovaný v tomto čase najpoužívanejšími autentifikačnými a autorizačnými rámcami na platforme Java ako Spring Security [43] a Apache Shiro [49]. Postupy

použité pre autentifikáciu a autorizáciu sú v oboch rámcoch podobné až identické. Apache Shiro nepotrebuje k integrácii žiadne ďalšie závislosti, naopak Spring Security je určený do aplikácií, ktoré su vybudované nad rámcom Spring Framework. Nevýhodou týchto rámcov je absencia podpory auditných záznamov, nepodporovanie viac autentifikačných metód v jednom čase a neprispôsobenie procesov k distribuovanému prostrediu a rôznym bezpečnostným modelom.

Rámec jD3A sa skladá z niekoľkých bázových komponent, ktoré sú nezávislé na použitých databázových, kešovacích či iných systémoch a technológiach. K bázovým komponentám sú vytvorené podporné moduly implementujúce rozšírenia k vyžadovaným službám alebo štandardom. V prílohe B je možné nájsť celý komponentový model rámca. Príloha A obsahuje detailný diagram tried a architektúry jD3A. Achitektúra rámca a návrhu každej komponenty rešpektuje tieto tri základné zásady:



Obr. 9.2: Jadrová komponenta rámca s vymedzenými subkomponentami

- *Auditovateľný* – všetky aktivity, ktoré ovplyvňujú stav subjektu alebo sedenia, musia podliehať auditu.
- *Vystopovateľný* – v každej vrstve rámca je možné určiť zdroj aktuálnej aktivity. Procesy sú jednoduché, priamočiare a spracovávané hierarchicky. Každá aktivita meniaca stav má presne vymedzenú softvérovú komponentu či vrstvu, ktorá je za zmenu zodpovedná.
- *Vysoká integrita* – celý rámec je navrhovaný a implementovaný konceptom založeným na programovaní oproti rozhraniu (Interface–As–A–Type, Interface–Based programming) [44]. Aplikovaním tohto konceptu sa zvyšuje modularita rámca. Jeho použitím nad bázovými objektmi sa zabráni neautorizovanému prepísaniu alebo narušeniu integrity dát.

V nasledujúcich kapitolách budú predstavené návrhy jednotlivých logických komponent, ich vlastnosti a procesy, ktoré pokrývajú vzhľadom k definovaným požiadavkám. Bázu rámca tvoria komponenty realizujúce základné procesy tak, ako prezentuje diagram 9.2.

9.1.1 Audit

Účelom modulu je jednoduchým a konfigurovateľným spôsobom vytvárať auditné záznamy v definovanom formáte. Návrh auditného modulu je inšpirovaný nástrojom Basic Security Module (BSM) [45] pôvodne predstaveným pre operačný systém Solaris, v tomto čase integrovaný v operačných systémoch Mac OS X a FreeBSD. Návrh modulu umožňuje spracovanie auditných záznamov vo formáte BSM.

Základnou jednotkou auditu je auditovateľná udalosť. Udalosti a obsah auditného záznamu, ktorý je udalosťou generovaný, sú presne špecifikované. Tabuľka 9.1 obsahuje prehľad udalostí generovaných jednotlivými komponentami jadra rámca.

Modul	Udalosť	Popis udalosti
Autentification	login	pokus o prihlásenie
	logout	odhlásenie
	create	vytvorenie účtu subjektu
	update	zmena dokladov subjektu
Authorization	request	požiadavka o autorizáciu
Subject Management	create	vytvorenie nového subjektu
	update	úprava vlastností subjektu
	load	načítanie subjektu
Session Management	create	vytvorenie nového sedenia
	update	úprava vlastností sedenia
	load	načítanie sedenia
Configuration	setup	nastavenie konfigurácie do globálneho kontajnera

Tabuľka 9.1: Udalosti generované jadrom rámca jD3A

Auditný záznam je komplexná dátová štruktúra skladajúca sa z týchto častí:

- *Čas* – čas vygenerovania auditného záznamu
- *UUID* – unikátny identifikátor auditného záznamu podliehajúceho RFC–4122 [25].
- *Kategória* – identifikuje oblasť, v ktorej bola udalosť vygenerovaná. Na základe kategórie je možné filtrovať auditné záznamy. Každý modul rámca definuje vlastnú unikátnu kategóriu, ktorá je lexikograficky vyskladaná na základe java balíku modulu a typu udalosti. Príklad: `org.jd3a.core.authorization.login`.
- *Výsledok* – súčasťou auditného záznamu je výsledok akcie, ktorá je definovaná udalosťou, teda úspech (success) alebo neúspech (failure).
- *Hodnoty* – obsah záznamu tvorí množina zložených hodnôt vo forme: *kľúč–hodnota*. Hodnoty sú poplatné charakteru danej udalosti. Príklad: pri neúspešnej autentifikácii je obsahom IP adresa klienta, užívateľské meno, použitá metóda autentifikácie, integritné obmedzenia subjektu.

Konfigurácia auditného modulu obsahuje nástroj pre vymedzenie udalostí, ktoré sa majú spracovať. Modul je možné konfigurovať priamo v zdrojovom kóde alebo konfiguračným súborom. Formát konfiguračného súboru v tabuľke 9.2 vychádza z formátu BSM.

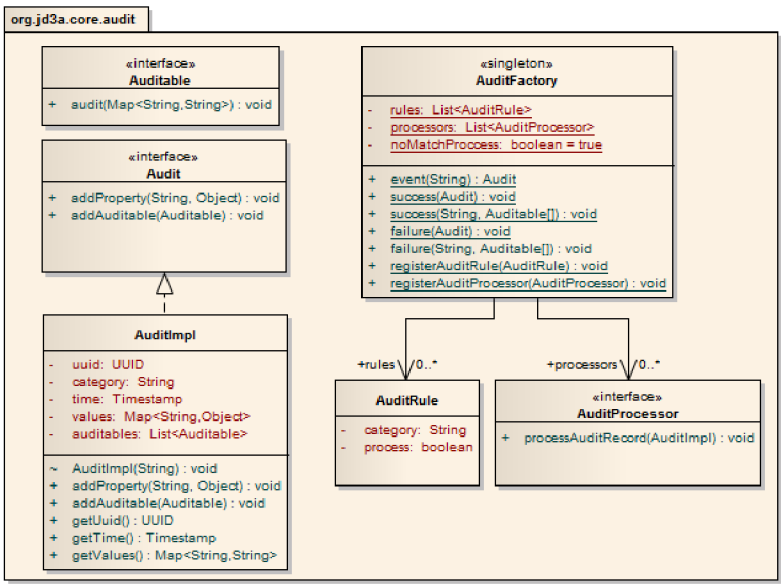
Vzor	Popis
`+<category>`	kategória `category` bude spracovaná
`-<category>`	kategória `category` nebude spracovaná
`+all`	všetky kategórie budú spracované
`-all`	všetky kategórie nebudú spracované

Tabuľka 9.2: Možnosti konfiguračného súboru auditného modulu

Diagram 9.3 zobrazuje triednu štruktúru auditného modulu. Centrálnym prvkom komponenty je statická trieda `AuditFactory`, ktorá je zodpovedná za generovanie nových auditných záznamov `AuditFactory.event()` a ich spracovanie `AuditFactory.success()`, `AuditFactory.failure()`.

Forma spracovania auditného záznamu sa definuje implementáciou rozhrania `AuditProcessor` a jej zavedením do auditného modulu `AuditFactory.registerAuditProcessor()`. To umožňuje zaregistrovať neobmedzené množstvo spôsobov, ako bude auditný záznam spracovaný.

Účelom rozhrania `Auditable` je poskytnúť jednoduchý nástroj k vytvoreniu obsahu auditného záznamu pre dôležité objekty, ktoré budú predmetom auditu (subjekty, sedenia). Vyhodnotenie sa vykonáva oneskorene až pri konečnom spracovaní.



Obr. 9.3: Diagram tried pre modul Audit rámca jD3A

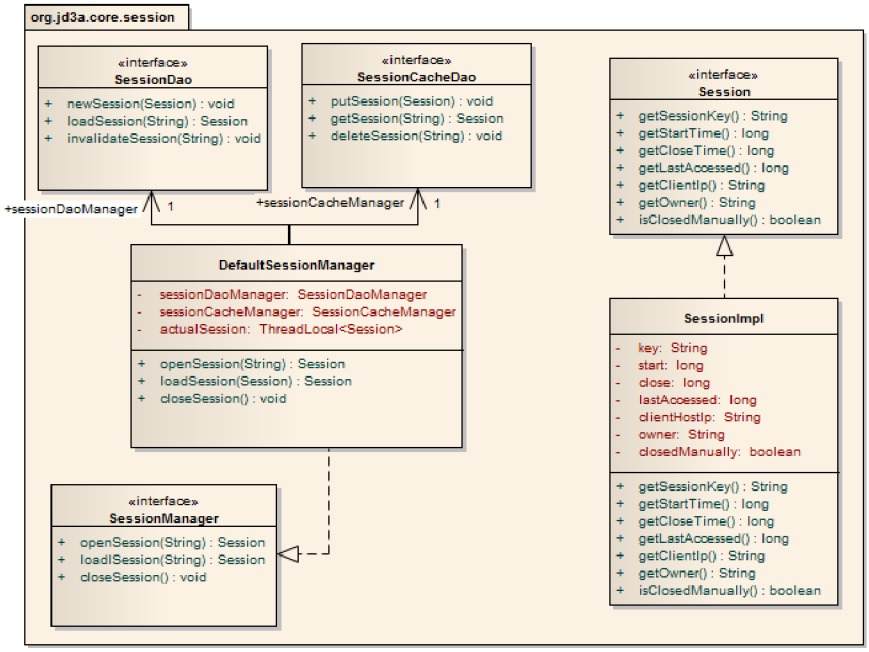
9.1.2 Správa sedení

Modul správy sedení poskytuje jednoduché rozhranie `SessionManager` k manipulácii so sedeniami. Cieľom je poskytnutie jedinej autority, ktorá môže meniť vnútorný stav sedení.

Sedenie (Session) je bázovou jednotkou, s ktorou modul pracuje, s jednoduchou štruktúrou:

- *Čas vytvorenia* – čas vytvorenia sedenia.
- *Čas posledného sprístupnenia* – čas, kedy bolo sedenie naposledy načítané.
- *Čas ukončenia platnosti* – čas, kedy končí platnosť sedenia.
- *UUID* – unikátny identifikátor sedenia, rešpektuje RFC–4122 [25].
- *IP adresa klienta* – IP adresa priradená pri vytvorení sedenia.
- *Príznak manuálneho ukončenia* – príznak či sedenie bolo ukončené manuálne alebo vypršalo na timeout.
- *Vlastník sedenia* – identifikátor vlastníka sedenia a jeho identifikačný token

Diagram 9.4 predstavuje triednu štruktúru modulu. K činnosti modulu je potreba dodať implementáciu DAO[1] rozhrania k službám realizujúcim kešovanie `SessionCacheDao` a perzistenciu `SessionDao` sedení. Sedenie je perzistované z dôvodu historickej dohľadateľnosti. DAO implementácie sa nastavujú do východzieho správcu sedení `DefaultSessionManager`.



Obr. 9.4: Diagram tried pre modul Session Manager rámca jD3A

Návrh správy sedení rešpektuje bezpečnostné doporučenia organizácie OWASP. Nasledujúci výčet predstavuje výber najdôležitejších doporučení, podrobnú analýzu môže čitateľ nájsť v prílohe D:

- *Identifikátor sedenia musí byť unikátny pre každý subjekt, náhodný a nepredvídateľný* – táto požiadavka je splnená použitím algoritmu na tvorbu UUID.

[1] Data–Access–Object návrhový vzor, ktorého účelom je poskytnúť abstraktné rozhranie k perzistentnej službe bez odhalenia špecifických vlastností alebo nastavení

- *Identifikátor sedenia musí používať čo najväčší možný rozsah* – UUID je kombinácia písmen latinskej abecedy a arabských číslic dlhá 32 znakov. Veľkosť kľúča sa odvíja od možností technológie určenej k správe sedení v klientskom prostredí. Identifikátor sedenia sa musí pri každej požiadavke preniesť spolu s obsahom, čo v prípade dlhých identifikátorov môže výrazne ovplyvniť prácu s aplikáciou na mobilných platformách.

- *Dĺžka aktívneho sedenia sa musí dať nastaviť* – táto požiadavka je splnená vzhľadom na globálnu konfiguráciu 9.1.6.

9.1.3 Správa subjektov

Modul správy subjektov, podobne ako modul správy sedení, poskytuje jednoduché rozhranie a predstavuje jedinú autoritu určenú k manipulácii so subjektmi.

Súčasťou dátovej štruktúry subjektu sú často dáta vzťahujúce sa k digitálnej identite (meno, adresa, telefónne číslo, rodné číslo a podobne). Rámec jD3A striktne oddeľuje dáta určené k autentifikácii, autorizácii a správe digitálnej identity. Tým sa vymedzia jednotlivé dátové jednotky v perzistentnom prostredí (databázi), čím sa zvýši úroveň bezpečnosti behového prostredia aplikovaním bezpečnostnej politiky v danej službe (nastavením vlastníkov tabuliek a databáz). Dôvod oddelenia vychádza tiež z charakteru distribuovaného prostredia a vlastností existujúcich subsystémov.

Jeden subjekt môže obsahovať neobmedzený počet dôverných dokladov (credentials), ktoré sú použité rozdielnymi autentifikačnými metódami. Perzistencii dôverných dokladov predchádza niekoľko bezpečnostných opatrení:

- Doklady sa neukladajú v otvorenom formáte, ale ako výsledok hešovacej funkcie[2]. Útočník nie je schopný ani po preniknutí do perzistentného úložišťa zistiť dôverné informácie.

- Vstupom do hešovacej funkcie je dôverný doklad konkatenovaný so soľou (salt). Tento proces sa nazýva solenie (salting), kde každý subjekt má vytvorenú náhodnú postupnosť znakov definovanej dĺžky (v prípade jD3A je to 8 bytov), ktorá je uložená v otvorenom formáte s dátovou štruktúrou subjektu. Proces je aplikovaný z dôvodu ochrany v prípade použitia dúhových tabuliek[3] alebo výskytu rovnakých hesiel v perzistovanom úložišti.

- Proces hešovania je opakovaný n-krát nad výsledkom predchádzajúcej iterácie. Zvýšením počtu opakovaní sa priamo–úmerne zvyšuje odolnosť voči útoku hrubou silou (hádaním hesla). Iterácie nie sú pre subjekt postrehnuteľné a pre útočníka predstavujú takmer neprekonateľnú prekážku – v prípade nedisponovania informáciou o presnom počte aplikovaných iterácií.

- Počet iterácií a typ hešovacej funkcie sú súčasťou globálneho nastavenia a je ich možné meniť pre každú autentifikačnú metódu osobitne.

Základná jednotka je subjekt, ktorá obsahuje len dáta potrebné k autentifikácii a identifikácii entity. Významnú časť dátovej štruktúry tvoria informácie o úspešných a neúspešných prihláseniach, ktoré sú použité pri ochranách voči útokom:

[2] Matematická funkcia, pre ktorú reverzný algoritmus neexistuje alebo je výpočetne nerealizovateľný. Pre lubovoľne dlhý vstup vytvorí vždy rovnako dlhý výstup.

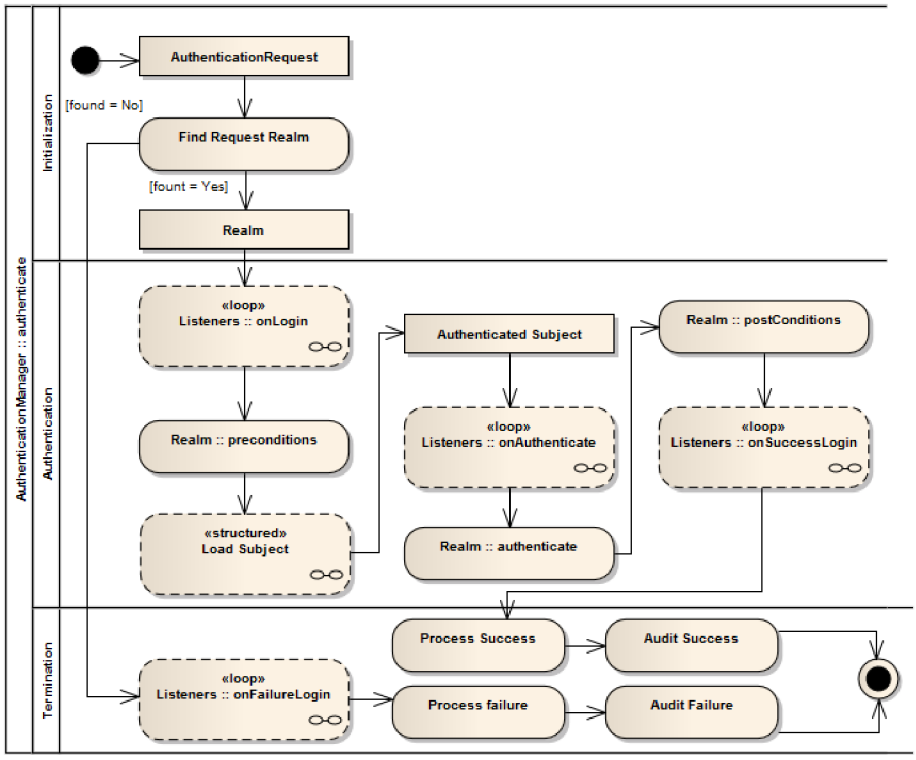
[3] Slovníková databáza obsahuje dvojice slovo–heš. Následne je možné reverzne nájsť otvorenú reprezentáciu dôvernej informácie.

- *Kľúč identity* – unikátny reťazec, ktorý predstavuje identifikáciu subjektu (užívateľské meno, email a podobne).
- *UUID identity* – unikátny identifikátor sedenia, rešpektuje RFC–4122 [25]. Identifikátor je primárnym kľúčom v subsystémoch odkazujúcich na subjekt.
- *Príznak expirácie* – príznak expirovaných dôverných informácií. Použitie pri vynútenej zmene dôverných informácií.
- *Príznak aktivácie* – definuje, či je účet subjektu aktívny resp. neaktívny, teda je možné resp. nie je možné sa prihlásiť do systému.
- *Mäkký zámok* – je možné ho odomknúť samoobslužne. Uzamknutie môže spôsobiť porušenie bezpečnostnej politky v distribuovanom systéme.
- *Tvrdý zámok* – môže ho odomknúť len administrátor. Uzamknutie môže spôsobiť snaha o útok hrubou silou alebo nebezpečné porušenie bezpečnostnej politiky v distribuovanom systéme.
- *Naposledy úspešná IP adresa autentifikácie* – IP adresa naposledy úspešnej autentifikácie.
- *Dátum naposledy úspešnej autentifikácie* – dátum, kedy sa subjekt naposledy úspešne autentifikoval.
- *Naposledy neúspešná IP adresa autentifikácie* – IP adresa naposledy neúspešnej autentifikácie.
- *Dátum naposledy neúspešnej autentifikácie* – dátum naposledy uskutočnenej neúspešnej autentifikácie.
- *Počet neúspešných autentifikácií v rade* – číslo, ktoré sa inkrementuje resp. nuluje po každej neúspešnej resp. úspešnej autentifikácii. V prípade prekročenia nakonfigurovaného prahu (treshold) sa účet automaticky uzamkne tvrdým zámkom.
- *Soľ subjektu* – pole bytov, ktorým sa solí každý dôverný doklad pri perzistencii resp. autentifikácii

Modul obsahuje východzieho správcu subjektov `DefaultSubjectManager`, ktorý implementuje rozhranie `SubjectManager`. Rovnako ako správca sedení, potrebuje správca subjekt dodať DAO `SubjectDao` implementáciu k perzistentnej službe.

9.1.4 Autentifikácia

Cieľom modulu je vykonať autentifikáciu entity vzhľadom na poskytnutý identifikačný token a dôverné doklady. V rámci modulu sú definované tzv. prístupové body (Entry Point), ktorých účelom je spracovanie autentifikácie na základe osobitnej politiky definovanej v globálnej konfigurácii. Každý prístupový bod je mapovaný na presne definovanú URL, v rámci ktorej môžu byť uplatnené špecifické obmedzenia a autentifikačná metóda (Realm). Návrh modulu a jeho vnútorného procesu autentifikácie je konceptuálne špecifikovaný na diagrame 9.5. Architekúra procesu je prispôsobená maximálnemu ovplyvneniu toku autentifikácie so zachovaním definovaných krokov. Proces je rozdelený do troch fáz:



Obr. 9.5: Diagram aktivity k procesu autentifikácie. Z dôvodu zvýšenia prehľadnosti nie je v diagrame zanesená aktivita, ktorá zachytáva všetky výnimky generované v časti Authentication.

1. *Inicializácia* – Z rozhrania `AuthenticationRequest`, ktoré predstavuje požiadavku k autentifikácii, sa zistí, na aký prístupový bod sa entita pripojila. V prípade neexistencie mapovania vzhľadom k nájdenému prístupovému bodu proces autentifikácie končí neúspechom a prechádza sa k fáze ukončenia.

2. *Autentifikácia* – Návrh jadra procesu umožňuje ovplyvniť tok na základe požiadaviek autentifikačných metód (`Realm`) alebo implementovaných a zaregistrovaných naslúchačov (`AuthenticationListener`). Ak autentifikačná metóda alebo naslúchač vyhodia výnimku, proces autentifikácie končí neúspechom a prechádza sa k fáze ukončenia. Podrobný popis aktivít jadra procesu autentifikácie:

   (a) `AuthenticationListener::onLogin` – notifikácia zaregistrovaných naslúchačov o novom požiadavku.

   (b) `Realm::preconditions` – kontrola autentifikačnej metódy a vyžadovaných vlastností či nastavení požiadavku.

   (c) Načítanie subjektu na základe identifikátoru subjektu extrahovaného z požiadavky autentifikačnou metódou `Realm::identityCredential`.

   (d) `AuthenticationListener::onAuthenticate` – notifikácia zaregistrovaných naslúchačov s vloženým subjektom.

   (e) `Realm::authenticate` – špecifický proces autentifikácie realizovaný vybranou metódou.

(f) `Realm::postconditions` – kontrola autentifikačnej metódy a vyžadovaných vlastností či nastavení po úspešnom vykonaní autentifikácie.

(g) `AuthenticationListener::onSuccessAuthenticate` – notifikácia zaregistrovaných naslúchačov o úspešnej autentifikácii.

3. *Ukončenie* – Účelom fáze je na základe výsledku autentifikácie vytvoriť auditný záznam. Ak sa ukončilo neúspechom, je potrebné auditovať dôvod neúspechu a informovať naslúchačov o neúspešnom pokuse o autentifikáciu `AuthenticationListener::onFailureAuthenticate`. V závislosti na výsledku procesu sa vykoná konečné spracovanie subjektu a prípadne vytvorenie nového sedenia. Účelom autentifikačného modulu nie je spracovanie a predanie sedenia entite. O to sa postará podporný modul, ktorý implementuje konkrétne stratégie správy sedení vzhľadom k použitému komunikačnému protokolu.

Množstvo informačných systémov v tomto čase využíva v rámci procesu autentifikácie doplnkové funkcie, ktorých použitie rámec jD3A ako aj služba Takeplace rezolútne odmieta, sú to:

1. Základné účty (default accounts) – do kódu aplikácie vložené špeciálne účty určené administrátorom či iným entitám, ktorých autentifikácia nepodlieha rovnakému procesu. Táto vlastnosť vnáša do systému netransparentné prvky a ohrozuje systém hlavne z hľadiska personálnej bezpečnosti. Nežiaducou vlastnosťou sú účty, ktoré by mohli slúžiť ako cieľ útoku. Typicky sa jedná o používané administrátorské identifikačné tokeny ako: administrator, root, sudo a podobne.

2. Funkčnosť zapamätania prihlásenia (Remember–Me) – zapamätanie prihláseného subjektu a jeho automatické príhlásenie pri nasledujúcom pripojení k službe. Zaručenie bezpečnosti účtu pred zneužitím je pri tejto funkčnosti veľmi zložité až neuskutočniteľné. Existujú rôzne metódy implementácie, ktoré odstraňujú nebezpečné faktory zneužitia účtu, ako je kontrola na krajinu hostiteľa, jeho IP adresu s kombináciou asymetrickej kryptografie. Často je potrebné ukladať dôverné informácie do klientského prostredia, čo je v rozpore s požiadavkami organizácie OWASP. Moderné webové prehliadače majú vstavanú funkčnosť správy hesiel a ich automatické doplňovanie do formulárových polí. Táto vlastnosť predstavuje náhradu funkčnosti remember–me.

3. CAPTCHA (Completely automated Turing Tests To Tell Humans and Computers Apart) – často využívaná metóda ochrany proti automatickým robotom prepisom slova z obrázku. Zavedením tejto funkčnosti vzniká diskriminácia ľudí s fyzickým (očným) postihnutím. Existuje riešenie, kedy namiesto obrázka je možné prezentovať slovo audio technikou, čo obmedzuje ľudí, ktorí sú sluchovo postihnutí.

4. Kontrolné otázky a odpovede – z počiatku veľmi využívaná metóda kontroly autenticity a identity užívateľa. Z dôvodu obáv zo zabudnutia užívatelia používajú odpovede, ktoré sú napadnuteľné slovníkovým útokom[4].

[4] Útok hrubou silou postupným skúšaním slov z oborovo vybraných slovníkov.

9.1.5 Autorizácia

Modul vyhodnocuje autorizačné požiadavky, ktoré sú generované subsystémami (v tomto kontexte je považovaný za subsystém aj systém, ktorý implementuje rámec jD3A). Úlohou procesu je nájsť na základe použitia dimenzií univerzálneho bezpečnostného modelu, ktorý tvorí autorizačnú požiadavku `AuthorizationRequest`, relevantné autorizačné pravidlo `AuthorizationRule` a vložením požiadavky ho vyhodnotiť. Výsledok `AuthorizationResponse` je poslaný späť do subsystému.

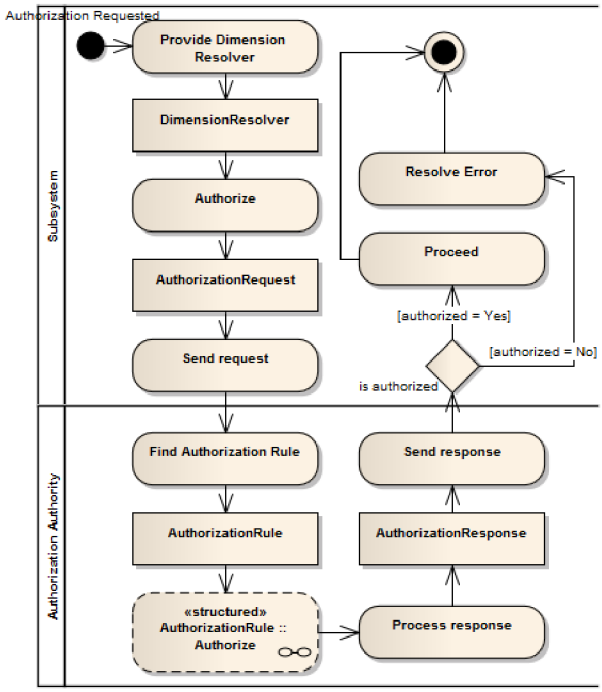
Autorizačné pravidlo

Každé autorizačné pravidlo je unikátne identifikované vzhľadom k ostatným pravidlám. V rámci distribuovaného prostredia môžu existovať logicky rovnaké pravidlá, avšak pracujúce s odlišnými zdrojmi autorizačných informácií – často je aj sám subsystém distribuovaný nad viac entitami (organizáciami či inými logicky alebo fyzicky oddelenými prostrediami). Logická časť (implementácia) autorizačného pravidla musí byť do rámca zaregistrovaná implementátorom. Za správne vyhodnotenie autorizačného pravidla je zodpovedný vývojár, ktorý dané pravidlo implementoval. To umožňuje vykonať bezpečnostné testovanie logiky pravidla a maximálnu kontrolu nad vyhodnotením, čo predstavuje základnú požiadavku pre vysoko kritické systémy.

Autorizačný proces

Proces vzniká na strane subsystému, ktorého účelom je vytvoriť autorizačnú požiadavku, a pokračuje do autorizačnej autority. Jej účelom je vzhľadom na definovanú bezpečnostnú politiku, relizovanú autorizačným pravidlom, vyhodnotiť požiadavku, vytvoriť odpoveď a odoslať odpoveď späť do subsystému. Tento proces je konceptuálne znázornený na diagrame aktivity 9.6:

1. *Vytvorenie požiadavky* – počiatočná fáza na strane subsystému, ktorá generuje autorizačnú požiadavku:
   (a) Podporný autorizačný nástroj vytvorí `DimensionResolver`, ktorého účelom, v náväznosti na aktuálny kontext subsystému a operácie podliehajúcej autorizácii, je vymedziť jednotlivé dimenzie a autorizačné pravidlo. Tento objekt predstavuje elementárnu operačnú jednotku, ktorá je vstupným parametrom autorizačnej služby texttt Authorization::authorize.
   (b) Autorizačná služba vyhodnotí jednotlivé dimenzie a vytvorí `AuthorizationRequest`, ktorý je vložený do zvolenej implementácie komunikačnej služby `AuthorizationCommService`.
   (c) Komunikačná služba musí byť podporovaná na strane subsystému a centrálnej autorizačnej autority. Použitá technológia na realizáciu komunikačnej služby musí byť synchrónna, v opačnom prípade je potreba v rámci implementácie zasiahnuť do jadra rámca.
2. *Vyhodnotenie požiadavky autorizačnou autoritou* – účelom fáze je vyhodnotiť požiadavku na strane autorizačnej autority a vzhľadom na výsledok spracovať dodatočné operácie:



Obr. 9.6: Diagram aktivity znázorňujúci proces autorizácie a spracovania autorizačného požiadavku a odpovede.

(a) Komunikačná služba autorizačnej autority prijme požiadavku a predá ju správcovi autorizácií `AuthorizationManager`.

(b) Vzľadom k autorizačnému pravidlu, ktorého identifikátor je obsahom požiadavky, správca nájde v registri odpovedajúce pravidlo a vloží jednotlivé dimenzie k vyhodnoteniu.

(c) Z výsledku autorizácie je vytvorená autorizačná odpoveď `AuthorizationResponse`, ktorá je vrátená komunikačnou službou do subsystému.

(d) Súčasťou autorizácie je vytvorenie auditného záznamu a v prípade neúspechu uzamknutie účtu subjektu a zrušenie sedenia.

3. *Spracovanie odpovede* – autorizačná odpoveď je spracovaná vzhľadom na kontext odpovede, ktorý definuje niekoľko stavov:

   - *OK* – subjekt je vzhľadom na vyhodnotené dimenzie autorizovaný. Súčasťou úspešnej autorizácie je vytvorenie bezpečnostného kontextu subjektu.
   - *NOT_AUTHORIZED* – subjekt nie je autorizovaný.
   - *RE_AUTHENTICATE* – identifikátor sedenia, ktorý predstavoval v autorizačnej požiadavke dimenziu subjektu, nie je aktívny a je potreba vykonať opätovnú autentifikáciu.
   - *RULE_NOT_FOUND* – definované autorizačné pravidlo nebolo v rámci autorizačného správcu nájdené.
   - *ERROR* – chyba pri autorizácii. Žiadne detaily o chybe nie sú do subsystému prenesené z dôvodu zachovania bezpečnosti a integrity systému.

Bezpečnostný kontext subjektu

Kontext `SecurityContext` definuje vlastnosti práve aktuálnej entity žiadajúcej o vykonanie služby, ktorú subsystém poskytuje. Rámec sprístupňuje bezpečnostný kontext cez jednoduché rozhranie `SecurityContextHolder::context()`. Bezpečnostný kontext je automaticky vytvorený pre každú novú požiadavku a udržovaný počas spracovávania požiadavky. Typ kontextu je nastavený na základe jednotlivých procesov podpory autorizácie alebo špecifickým nastavením užívateľa rámca. Typy základných bezpečnostných kontextov a ich popis:

1. `NullContext` – východzí kontext, ktorý neobsahuje žiadne atribúty identifikujúce entitu
2. `SessionSecurityContext` – kontext je inicializovaný v prípade, že podporné autorizačné procesy extrahovali kľúč sedenia z požiadavky. Existencia tohto kontextu je vyžadovaná v prípade autorizačných požiadaviek, kde dimenzia subjektu je identifikovaná kľúčom sedenia.
3. `SubjectSecurityContext` – kontext je automaticky nastavený v prípade úspešnej autorizácie subjektu reprezentovaného kľúčom sedenia resp. rozširuje `SessionSecurityContext` o UUID a identifikačný token subjektu.
4. `LocalSecurityContext` – nastavuje sa pre automatické rutiny v rámci vnútorných procesov subsystému.
5. `SubsystemSecurityContext` – rozhranie identifikuje entitu ako iný subsystém v rámci distribuovaného prostredia.

Bezpečnostný kontext je využívaný k identifikácii subjektu a uchovaniu kontextovo závislých informácií vzťahujúcich sa k entite pri procese autorizácie aj autentifikácie. Typy kontextov sa môžu lubovoľne rozširovať.

Komunikačné rozhranie

Medzi autorizačnou autoritou a subsystémami prebieha komunikácia, ktorej predmetom je autorizačná požiadavka a výsledok autorizácie. Rozhranie musí byť nezávislé na platforme a technológii subsystému.

Autorizačné nástroje

Súčasťou modulu autorizácie sú podporné autorizačné nástroje, ktorých účelom je jednoduchým a v rámci možností neinvazívnym spôsobom zabezpečiť subsystém a jeho zdroje. V nasledujúcich kapitolách budú predstavené a popísané navrhované nástroje pre platformu Java. Celkový počet nástrojov bez ohľadu na platformu sa vďaka transparentnej podpore komunikačnej služby a dimenzií môže v rámci ďalších verzií rozširovať.

Java Anotácie Java anotácie predstavujú syntaktickú formu metadát, ktoré sú priamo vložené do zdrojových súborov. Anotácia rozširuje vlastnosti, ktoré môžu byť spojené s balíkmi, triedami, metódami, atribútmi a podobne. Môže obsahovať premenné, ktoré sú súčasťou nastavenia anotácie pri jej použití. Atribúty anotácie môžu byť len primitívne dátové typy, pole, trieda alebo anotácia. Pre každú anotáciu sa definujú dva význačné atribúty (opäť anotáciou), ktoré definujú možnosti použitia anotácie:

1. `@Retention` – určuje úroveň, do akej bude anotácia v rámci procesu spracovania zdrojového súboru zachovaná a prístupná. Rozlišujú sa tri typy politiky zachovania:
   - `SOURCE` – anotácia je len súčasťou zdrojového kódu a je spracovávaná prekladačom. Príklad: detekcia a generovanie upozornení v čase prekladu (`@Deprecated`).
   - `CLASS` – anotácia je zachovaná v rámci Java JVM a pri kompilácii je možné dodatočné spracovanie nástrojov tretích strán. Príklad: automatické generovanie kódu alebo XML súborov (JAXB2). Toto nastavenie je východzie.
   - `RUNTIME` – anotácia je súčasťou Java JVM a prístup k nej je zabezpečený cez java reflexiu.
2. `@Target` – definuje množinu programových elementov, ku ktorým môže byť anotácia priradená: balík, trieda, metóda, atribút, lokálna premenná, parameter metódy, konštruktor, anotácia.

Rámec obsahuje anotácie, ktoré simulujú bezpečnostné modely definované v požiadavkách na systém, ako prezentuje tabuľka 9.3. Anotácie sú definované pre programový element metódy a triedy s retenciou `RUNTIME`.

Bezp. model	Anotácia	Popis
DAC	`@RequiresGroups`	Subjekt musí byť súčasťou jednej alebo všetkých definovaných skupín
DAC	`@RequiresSubject`	Subjekt musí byť identifikovaný definovaným identifikátorom.
DAC	`@RequiresAuthenticated`	Predstavuje základnú anotáciu, ktorá požaduje aby subjekt bol úspešne autentifikovaný a mal aktívne sedenie.
MAC	`@RequiresSecurityAttributes`	Subjekt musí mať priradený jeden alebo všetky požadované bezpečnostné atribúty.
RBAC	`@RequiresRoles`	Subjekt musí mať priradenú jednu alebo všetky požadované role.

Tabuľka 9.3: Podpora anotáciami pre voľný bezpečnostný model (DAC), mandatórny bezpečnostný model (MAC) a model založený na rolách (RBAC).

Úlohou anotácie je vzhľadom na bezpečnostný model, ktorý simuluje, vymedziť jednotlivé dimenzie bez dodatočnej konfigurácie priamo v zdrojovom kóde danej metódy. Každá anotácia obsahuje okrem vlastností špecifických pre daný bezpečnostný model, ktoré vymedzujú dimenziu modality, aj ďalšie atribúty, ktorých účelom je definovať ostatné dimenzie 9.4.

Priame volanie autorizácie V prípade komplikovaného procesu vymedzenia dimenzií sa dá použiť priame volanie autorizácie vložením objektu `DimensionResolver` do autorizačnej služby `AuthorizationService::authorize`.

Parameter anotácie	Popis
`restriction`	Parameter obsahuje anotácie, ktoré umožňujú vložiť viac ako jeden bezpečnostný atribút (role, skupiny, atribúty). Obmedzenie vyhodnotenia autorizačnej požiadavky sa vzťahuje k logickým operátorom AND (`Restriction.ALL`) a OR (`Restriction.AT_LEAS_ONE`).
`rule`	Definuje triedu autorizačného pravidla, ktoré sa má aplikovať na vymedzené dimenzie.
*`argIndex`	Index parametru volanej metódy, ktorý má slúžiť k vymedzeniu dimenzie objektu.
*`objects`	Pole anotácií `@Obj`, ktorá definuje parametry volanej metódy, ktoré budú použité k vyhodnoteniu dimenzie objektov.
* – Vymedzenie je podporované natívne pre primitívne dátové typy. V prípade vymedzenie objektov musí objekt implementovať rozhranie `Extractable`.	

Tabuľka 9.4: Spoločné vlastnosti anotácií.

9.1.6 Konfigurácia

Rámec obsahuje množstvo nastavení, ktoré sú záväzné pre celý distribuovaný systém. Konfigurácia sa skladá z dvoch častí: konfigurácia rámca a konfigurácia subsystému.

Konfigurácia rámca je globálne prístupná všetkým modulom a komponentám. Obsahuje východzie a špecifické nastavenia prípojných bodov. Zavedenie konfigurácie je vykonané implementáciou rozhrania `ConfigurationProvider`, ktorý nastaví spracované nastavenie do `ConfigurationContainer`. To umožňuje transparentne implementovať rôzne spôsoby spracovania konfiguračných dát z .properties súborov, XML, Spring kontextu a podobne. Podrobný zoznam nastavení rámca nájde čitateľ v prílohe F.

9.1.7 Webová podpora

Komponenta realizuje konečnú interpretáciu globálnych nastavení a rozširuje ich o špecifické vlastnosti charakteristické pre webové prostredie definované protokolom HTTP. Úlohou komponenty je poskytnúť autentifikačné a autorizačné nástroje k integrácii rámca do tohto prostredia.

Autentifikácia

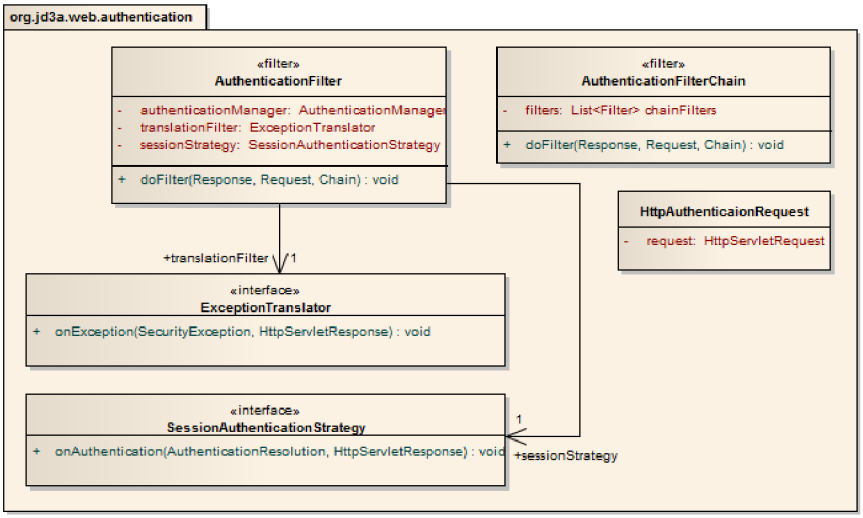
K autentifikácii entít vo webovom prostredí existuje množstvo metód, ktoré často implementujú rozdielnym spôsobom rovnaký postup poskytnutia identifikačného tokenu subjektu (užívateľského mena) a dôverného dokladu (hesla) serverovej aplikácií. V tomto prípade sa jedná o jednofaktorovú tj. slabú autentifikáciu. Prenášané dáta protokolom http sa posielajú v otvorenom formáte. K zaručeniu maximálnej bezpečnosti je treba, aby komunikácia bola šifrovaná pomocou SSL (https). Rámec poskytuje implementáciu najpoužívanejších autentifikačných metód, ktorými sú:

1. *Štandard RFC 2617 : HTTP Authentication: Basic and Digest Access Authentication* – ktorý je podporovaný vo všetkých majoritných webových prehliadačoch. Obsahuje

dve možnosti autentifikácie, a to základnú (Basic), u ktorej sa vyžaduje užívateľské meno a heslo, a pomocou certifikátu (Digest), ktorá vyžaduje vlastnenie certifikátu.

2. *Username–Password* – poskytnutie užívateľského mena a hesla vyplnením formulárových polí a odoslanie formulára metódou POST alebo GET.

Každý prístupový bod funguje na princípe filtrovania `javax.servlet.Filter` požiadaviek nad definovanou kontextovou cestou URL. Vnútorný proces spracovania autentifikačnej požiadavky, filtrovania obsahu `javax.servlet.Filter::doFilter`, je realizovaný použitím virtuálnej reťaze `javax.servlet.FilterChain` aplikačných filtrov. Tie umožňujú rozširovať obmedzenia kladené na autentifikačný proces vo webovom prostredí. Požiadavka je postupne spracovaná každým aplikačným filtrom v reťazi, kde každý filter má právo veta – ukončiť proces autentifikácie ešte pred vstupom do jadra rámca.



Obr. 9.7: Koncepčný diagram tried webovej podpory autentifikácie.

Posledný filter v reťazi je hlavný autentifikačný filter `AuthenticationFilter`. Jeho účelom je vytvoriť autentifikačnú požiadavku `HttpAuthenticationRequest` a vložiť ju do autentifikačného jadra rámca. Návrh umožňuje v rámci prístupového bodu zaregistrovať prekladač výnimiek `ExceptionTranslator`, ktorý spracováva výnimky generované autentifikačným jadrom, a vzhľadom na typ výnimky reagovať odlišným spôsobom.

Po úspešnej autentifikácii sa na základe zvolenej stratégie `SessionAuthenticationStrategy` predá sedenie do klientského priestoru. Zvolená stratégia správy sedení musí byť rovnaká v autorizačnej podpore, ktorá identifikuje subjekty podľa kľúča sedenia.

Autorizácia

Modul webovej podpory rozširuje autorizačné nástroje o možnosť zabezpečenia aktív prístupných cez definovanú URL. Dimenzie sú vymedzené na základe kontextu cesty a parametrov požiadavky.

9.2 Kontrola prístupu v systéme Takeplace

Kapitola popisuje spôsob integrácie rámca jD3A do prostredia služby Takeplace.

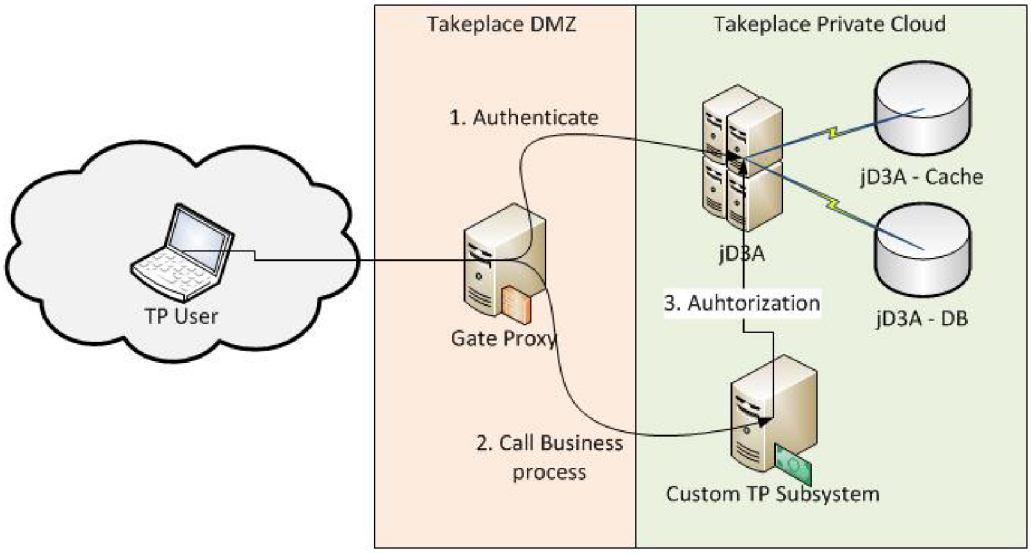
9.2.1 Integrácia jD3A do prostredia služby

Celý AAA subsystém bude pracovať nad rámcom jD3A. Tento rámec môže byť zaimplementovaný do existujúceho subsystému alebo prevádzkovaný ako samostatný subsystém.

Obrázok 9.8 zobrazuje konečné zasadenie jD3A ako autonómneho subsystému do prostredia služby Takeplace. Subsystém sa nachádza v privátnej zóne, ktorá je dostupná cez proxy server umiestnený v demilitarizovanej zóne[5]. Za účelom zvýšenia bezpečnosti systému proxy server preposiela len autentifikačné požiadavky na presne definované autentifikačné prístupové body (entry pointy).

Proxy server je realizovaný službou Apache Web Server, a ten je následne odosielateľom preposlanej (forwarded) požiadavky. Z tohto dôvodu je potreba upraviť spôsob vymedzenia IP adresy z autentifikačnej požiadavky, a to bude poplatné vlastnostiam vloženým web serverom. Pôvodná IP adresa je uložená v parametri `X-Forwarded-For` ako je definované v [47].

Autorizačné rozhranie subsystému nie je z externého prostredia dostupné. Autorizačné služby môžu využívať len subsystémy umiestnené vo vnútornom prostredí realizovanom službou Amazon Virtual Private Cloud (VPC) [2]. Z dôvodu vlastností plynúcich z Amazon VPC je možné ďalej rozširovať dostupnosť služby Takeplace aj do ďalších svetových uzlov bez zmeny vnútornej sieťovej architektúry prostredia.



Obr. 9.8: Zasadenie AAA subsystému do aktuálneho prostredia systému Takeplace

Postup na obrázku 9.8 popisuje priebeh autentifikácie a následnej autorizácie volania zabezpečenej služby subsystému. Klient sa musí najskôr autentifikovať voči jednému prístupovému bodu AAA subsystému. Následne klient vykoná volanie vystavenej služby voliteľného subsystému. Za predpokladu, že je táto služba zabezpečená podporným nástrojom jD3A, sú z požiadavky vymedzené dáta na vykonanie autorizácie – identifikátor aktuálneho sedenia a jednotlivé dimenzie. Tie sú potom poslané autorizačnej autorite, v ktorej prebehne proces autorizácie.

[5]DMZ alebo demilitarizovaná zóna je fyzická alebo logická časť sieťovej architektúry organizácie, ktorá vystavuje služby softvérového charakteru do internetu

9.2.2 Integrácia kontroly prístupu

Pre systém Takeplace je nastavený jeden autentifikačný prístupový bod použitím metódy Username–Password. Nasledujúce podkapitoly detailne popisujú návrh procesu integrácie autorizácie do jednotlivých subsystémov Takeplace použitím rámca.

Subsystém správy akcií a digitálnych identít

Subsystém je vystavaný nad voľným bezpečnostným modelom (DAC). Autorizácia je vyhodnotená na základe identity subjektu. Všetky akcie subsystému, okrem vytvorenia účtu a jeho aktivácie, podliehajú autorizácii. Autorizačné pravidlo zaregistrované do rámca vyhodnocuje identitu prihláseného užívateľa. Všetky rozhrania služieb sú zabezpečené anotáciou `@RequiresAuthenticated`.

Konferenčný subsystém

Kombinuje skupiny voľného bezpečnostného modelu a model založený na rolách. Skupina pozostáva z množín členov a rolí skupiny. Priestor každej akcie je izolovaný od ostatných. Výbory sú abstrakciou niekoľkých skupín: predsedovia, podpredsedovia, členovia, kde počet členov môže byť aplikačne odmedzený.

Dimenzia objektu je vymedzená unikátnym identifikátorom konferencie. Operácie konferenčného subsystému sú autorizované vzhľadom na požadované role. Autorizačný subsystém komunikuje s konferenčným subsystémom, kde predmetom je požiadavka na existenciu role v akejkoľvek skupine, ktorej je subjekt členom, teda:

$$\exists R \in R_A : S \in G_M \wedge R \in \bigcup_{i=0}^{n} G_{R_i}, \tag{9.1}$$

kde R je žiadaná rola, R_A množina všetkých rolí v systéme, S je subjekt, voči ktorému je pravidlo aplikované, G_M je množina všetkých členov skupiny G, G_R je množina všetkých rolí skupiny G.

Subsystém komunitných služieb

Komunitné služby sú autorizované identitným pravidlom – overuje sa identita užívateľa, na základe ktorej sú vyhodnotené požiadavky. Dimenzia objektu je vymedzená komponentou, ku ktorej subjekt pristupuje: nástenka užívateľa, nástenka konferencie, nasledovníci, následovaný. Dimenzia operácií je vymedzená jednou reprezentáciou CRUD akcie.

Notifikačný subsystém

Spracovanie notifikácie podlieha typovej autorizácii notifikačných metód: email, sms, hlasové notifikácie. Integráciou do rámca vzniknú tri autorizačné pravidlá, pre každý typ notifikácie osobitné pravidlo.

Autorizačné pravidlo emailovej notifikácie vyhodnocuje existenciu pridelenej systémovej role v konferenčnom priestore. Autorizácia sms a hlasovej notifikácie podlieha dvom krokom. Konferencia musí mať povolený typ notifikácie a subjekt musí mať pridelenú rolu v konferenčnom systéme umožňujúcu posielať daný typ notifikácie.

Dimenzia objektu je vymedzená unikátnym identifikátorom konferencie, dimenzia operácie je vymedzená jedinou operáciou v celom subsystéme reprezentujúcou poslanie notifikácie.

Kapitola 10

Implementácia

Táto kapitola rozoberá špecifické implementačné detaily rámca jD3A a doplňkových služieb, ktoré sú použité pri integrácii rámca do prostredia služby Takeplace. Rozširujúce moduly sú súčasťou verejnej distribúcie rámca.

10.1 Audit

Auditný modul je používaný aj mimo presne vymedzené udalosti, ktoré boli identifikované v tabuľke 9.1. Ukážka vytvorenia auditného záznamu a jeho spracovania:

```
// generates new audit record with given category
Audit audit = AuditFactory.event(CUSTOM_CATEGORY);
// adds new audit record property
audit.addProperty("foo","foo");
// bar implements Auditable interface
audit.addAuditable(bar);
..
// action ends with success
AuditFactory.success(audit);
// or failure
AuditFactory.failure(audit);
...
// usage when only auditable objects occurs
AuditFactory.success(CUSTOM_CATEGORY,foo,bar);
// same for failure audit
AuditFactory.failure(CUSTOM_CATEGORY,foo,bar);
```

Návrh auditného modulu umožňuje registrovať neobmedzené množstvo spracovávačov auditných záznamov. Súčasťou jadra rámca je automaticky registrovaný spracovávač `SL4J-AuditProcessor`, ktorý auditné záznamy loguje pomocou jednoduchého logovacieho rozhrania. Ukážka výstupu:

```
Audit[org.jd3a.core.session.load] [error:SessionNotFoundException] ...
```

Konfigurácia auditného modulu môže byť konfigurovaná priamo alebo načítaním zo súboru. Nástroj sa automaticky po štarte rámca pokúša nájsť konfiguračný súbor *jd3a-audit*. Vyhodnotenie pravidiel je v ronakom slede, v akom boli registrované – prvé pravidlo je na konci sledu. Ukážka zavedenia auditného pravidla:

```
// audit all events from package org.jd3a.core.session
AuditFactory.registerAuditRule("+org.jd3a.core.session");

// configure audit module from custom file
AuditFactory.configureFromFile("audit.settings");
```

10.2 Autentifikácia

Subjekty sú identifikované podľa kľúča sedenia, ktoré bolo predané do klientského priestoru po úspešnej autentifikácii. Kapitola 4.1 identifikovala metódy správy sedení pre webové aplikácie. Podporný modul pre webové prostredie implementuje tri stratégie`SessionAuthenticationStrategy` správy sedení:

1. `HttpSessionAuthenticationStrategy` – využíva Session API definované špecifikáciou *JSR-000154 : Servlet 2.5 Specification* [6]. Tento spôsob správy sedení nie je určený do distribuovaného prostredia z dôvodu prebratia správy sedení vnútornými procesmi aplikačného servera, na ktorom beží rámec jD3A. V prípade použitia rámca pre nedistribuovanú aplikáciu je táto metóda optimálna.

2. `UrlRewriteSessionAuthenticationStrategy` – prepisuje URL do formátu `scheme://domain:port/servlet/context?query;<sessionid=value>`. Použitie tejto správy sedení vyžaduje, aby ostatné webové technológie neodstránili identifikátor sedenia z URL, v opačnom prípade dôjde ku strate kontextu sedenia.

3. `CookieSessionAuthenticationStrategy` – kľúč sedenia je vložený do klientského priestoru ako cookie s menom `jd3a-session` a hodnotou kľúča sedenia.

10.3 Autorizácia

Podkapitola popisuje implementačné detaily vytvorenia a spracovania autorizačnej požiadavky. Predstavuje konkrétne technológie realizujúce spracovanie autorizačných anotácií a komunikačnú službu.

10.3.1 Spracovanie anotácií

Anotácie vkladajú do zdrojového súboru metadáta, ktoré musia byť automaticky spracované, k čomu sa používa aspektovo orientované programovanie [9]. Toto programovacie paradigma bude rozobrané v ďalšej časti. Rámec jD3A je nezávislý na použitej implementácii AOP. Nasledujúca ukážka demonštruje spôsob definície anotácií v rámci:

```
@Target({ElementType.TYPE, ElementType.METHOD})
@Retention(RetentionPolicy.RUNTIME)
@Inherited
@Documented
public @interface RequiresGroups {
  ...
}
```

Anotácia `@Target` a `@Retention` sú špecifikované v 9.1.5. Anotáciou `@Inherited` sa špecifikuje vlastnosť dedenia anotácie na potomkov triedy. Anotácia `@Documented` zaistí, že anotácia bude súčasťou dokumentácie danej triedy či metódy.

Aspektovo orientované programovanie (AOP)

Paradigma AOP je konceptuálne postavené na spracovaní funkčnosti logických aplikačných blokov programu, ktoré sú pre danú aplikáciu opakovane použité. Jedná sa o tzv. prierezové záujmy či pretínajúce záležitosti (Cross-cutting concerns) [40]. Účelom AOP je vymedziť prierezové záujmy a ich spracovanie sústrediť do jedného bodu, čím sa zvýši modularita danej aplikácie.

AOP je možné nahradiť v určitých prípadoch použitím návrhových vzorov Template Method, Proxy, Interceptor, Observer [41], no ich použitie musí byť už súčasťou návrhu aplikácie.

AOP je najčastejšie postavené nad modelom prípojných bodov (Join–Point Model), ktorý definuje štyri komponenty [40]:

1. *Bod spojenia (Join–Point)* – presne definované miesto v programe, ku ktorému má byť priradená nová funkčnosť. Týmto bodom sa definuje miesto, ku ktorému môže byť priradený prierez.

2. *Bod rezu (Pointcut)* – z množiny všetkých bodov spojenia vymedzuje podmnožinu, ktorá definuje, kde bude aplikované konkrétne rozširujúce chovanie programu.

3. *Pokyn, rada (Advice)* – jedná sa o konkrétnu implementáciu prídavnej funkčnosti, ktorou má byť rozšírená aplikačná logika. Rozširujúce chovanie môže byť aplikované nad bodom spojenia v týchto variantoch:

   (a) *Pred (Before–Advice)* – pred aplikačnú logiku, ktorá je definovaná bodom spojenia.

   (b) *Po (After–Advice)* – rozširujúca funkčnosť bude vložená za aplikačnú logiku, bez ohľadu na výsledok priebehu pôvodného chovania programu (výnimka, normálny priebeh).

   (c) *Obalenie (Around–Advice)* – najsilnejší typ, ktorý umožňuje plne kontrolovať spustenie resp. nespustenie pôvodnej funkčnosti s dodanými alebo upravenými parametrami.

4. *Aspekt (Aspect)* – konkrétna štruktúra spájajúca body spojenia, bod rezu a rozširujúcu aplikačnú logiku vo forme pokynu.

Konečnou fázou zavedenia aspektov do aplikácie je proces nazývaný preplietanie (weaving). Jeho účelom je do stávajúcej aplikačnej logiky vložiť definované aspekty a zabezpečiť ich úspešné zapracovanie. Vzhľadom na výkonnosť aplikácie a zavedenie aspektov existujú štyri typy preplietania na platforme java:

1. *Preplietanie počas prekladu (Compile-time weaving)* – rozširujúca funkčnosť aspektu je priamo vložená v čase prekladu zdrojových súborov.

2. *Preplietanie po preklade (Byte-code weaving)* – proces rozširuje byte kód preložených tried o rozširujúcu funkčnosť aspektu.

3. *Preplietanie pri nasadení (Deploy-time (load-time) weaving)* – rozšírenie funkčnosti o aplikačnú logiku aspektu sa deje pri načítaní triedy pomocou upraveného nástroja Java Class Loader, ktorý automaticky rozpoznáva body spojenia.

4. *Preplietanie za behu (Run-time weaving)* – tento spôsob vytvára proxy objekty pre požadované triedy identifikované bodmi spojenia. To musí zabezpečiť vnútorný proces aplikácie. Jedná sa o najpomalšie zavedenie a spracovanie aspektov.

AspectJ

Spracovanie autorizačných anotácií vo východzej verzii rámca je realizované pomocou najznámejšieho AOP rámca na platforme java AspectJ [8]. Podporuje zavedenie aspektov (preplietanie) počas a po preklade a pri nasadení. Preplietanie za behu nie je podporované, ale je možné ho realizovať užitím návrhových vzorov.

Aspekt spracovávajúci anotácie je implementovaný v `AspectJAuthorizationAnnotationAspect`, ktorého body rezu (pointcuts) sú definované ako:

```
execution(@org.jd3a.authorization.support.dac.annotation.RequiresGroups * *(..))
execution(@org.jd3a.authorization.support.dac.annotation.RequiresSubject * *(..))
execution(@org.jd3a.authorization.support.mac.annotation.RequiresSecurityAttribu-
tes * *(..))
execution(@org.jd3a.authorization.support.rbac.annotation.RequiresRoles * *(..))
execution(@org.jd3a.authorization.support.annotation.RequiresAuthenticated * *
(..))
```

Implementácia spracovania pokynu (advice) vytvorí objekt `AspectJInvokedMethod` reprezentujúci práve volanú metódu, ktorá je definovaná rozhraním `InvokedMethod` a ktoré je vložené do nezávislej služby `AuthorizationAnnotationMethodInterceptor`. Tá vzhľadom na zaregistrované nástroje spracovania anotácií `AuthorizationAnnotationHandler` vyhľadá prítomné anotácie v práve zavolanej metóde či triede a získa objekt `DimensionResolver`, ktorý je spracovaný štandardným spôsobom.

10.3.2 Zabezpečenie zdrojov dostupných cez URL

Súčasť webovej podpory umožňuje zabezpečiť zdroje dostupné cez URL. K tomu je vytvorený filter `SecuredResourceFilter`, ktorý vzhľadom na konfiguráciu vytvára autorizačné požiadavky, ktorých dimenzie sú automaticky vymedzované podobne ako u anotácií. Dimenzia objektov je definovaná parametrami požiadavky a operácia zase použitou metódou (PUT, GET, POST).

Ak je objekt súčasťou kontextovej cesty[1] je možné identifikovať objekt pomocou značky `#`. Kontextová cesta je zadaná presne alebo pomocou java regulárnych výrazov. Príklad:

```
<!-- Specific URI -->
/private/administration.jsp
<!-- Identifies objects by # -->
/category/#cat/product/#product
<!-- Regular expression -->
/download/(.)*
```

[1] Používa sa pri tzv. pekných URL, ktorých parametry sú obsahom kontextovej cesty napr. `/category/-220304/product/30`.

10.3.3 Komunikačná služba

Medzi autorizačnou autoritou a subsystémami prebieha komunikácia, ktorej predmetom sú autorizačné požiadavky. Implementácia komunikačnej služby musí byť nezávislá na použitej platforme. Východzia implementácia komunikačného rozhrania využíva webové služby, ktoré sú priamo vstavané do rámca.

Webové služby (Web Services)

Webové služby umožňujú vzdialené volanie procedúr (RPC - Remote Procedure Call) pomocou prenosu správ vo formáte XML protokolom HTTP. K definícii typov dát slúži štandard XML Schema [39]. K identifikácii objektov (dát, názvov operácií a podobne) využíva štandard XML Namespaces. Webové služby tvoria tri časti [24]:

1. Vzdialené volanie procedúr SOAP [52] – je protokol k výmene správ založených na XML. Tvorí základnú vrstvu komunikácie a poskytuje prostredie na tvorbu zložitých dotazov. Pracuje na princípe klient–server, kde server okamžite odpovedá na požiadavky klienta. Požiadavka (envelope) pozostáva z hlavičky (envelope–header) a tela správy (envelope–body).

2. Jazyk pre popis poskytovaných služieb WSDL [10] – popisuje dostupné operácie webovej služby, jej datové typy, parametre, návratové hodnoty a kde a ako je služba dostupná (HTTP/HTTPS/SMTP, port, stroj, URL). WSDL popisuje syntax dostupných operácií vo formáte XML.

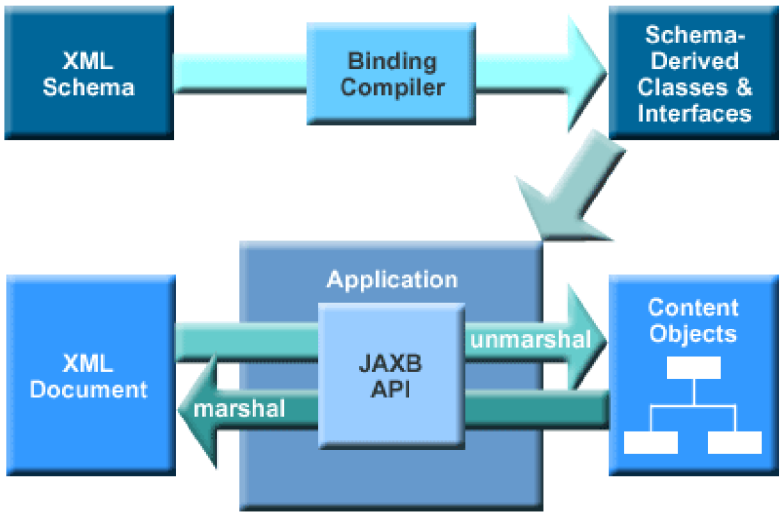
3. Nástroj k vyhľadaniu služieb UDDI [50] – poskytuje mechanizmus k registrovaniu, kategorizovaniu a vyhľadávaniu webových služieb. Funguje na princípe rozsiahleho adresára, obsahujúceho informácie o subjektoch (organizáciach) a ich poskytovaných službách. Register je realizovaný ako verejná webová služba.

Výhodou webových služieb je nezávislosť na platforme vďaka definícii komunikačného formátu pomocou XML Schema a dostupných nástrojov na spracovanie správ. Prenos dát vo formáte XML tvorí zároveň nevýhodu tejto technológie. Je tomu z dôvodu veľkého objemu riadiacich dát (XML tagy) a nutnosti koncových bodov spracovávať opakovane textové správy do vnútorných objektov a naopak.

JAXB–WS

Východzia implementácia autorizačnej webovej služby je vystavaná na Java štandarde *JSR 224: API for XML-Based Web Services (JAX-WS) 2.0* [22], ktorý ku spracovaniu XML využíva štandard *JSR 222: Architecture for XML Binding (JAXB) 2.0* [13].

Charakteristickou vlastnosťou použitia štandardu JAX–WS ako aj ďalších implementácií webových služieb v rámci, je prístup ich tvorby metódou *Contract–First*. Teda najskôr sa vytvorí XML Schema definujúca poskytované služby a objekty, ktorá sa vloží do konfigurácie webovej služby WSDL. Nástroj JAXB pri kompilácii z určenej XML Schemy vygeneruje triedy obohatené o JAXB anotácie, ktoré sú použité pri behu aplikácie k serializovaniu resp. deserializovaniu objektov do resp. zo XML. Tento proces znázorňuje obrázok 10.1. Automatizovanými nástrojmi JAX–WS sa vygeneruje na základe definovanej WSDL konfigurácie webová služba.



Obr. 10.1: Proces spracovania XML štandardom JAXB a jeho použitie v aplikácii. Obrázok je prevzaný z [37]

Autorizačná webová služba

Celý komunikačný aparát autorizácie rámca je generovaný z niekoľkých XML schém nástrojom JAXB. Výhoda tohto prístupu je vo vždy aktuálnej a platformovo nezávislej interpretácii komponent vzťahujúcich sa ku komunikačnej časti rámca.

Pri každom preklade zdrojovej knižnice budú zo schémy vygenerované všetky komponenty. Užívatelia rámca môžu vytvorením nových schém rozšíriť možnosti prenášaných objektov resp. dimenzií, ktoré tvoria autorizačnú požiadavku. Súčasťou serverovej časti webových služieb je vystavenie WSDL súboru, generovaného z definovaných XML schém. Rámec poskytuje tieto schémy:

1. `core.xsd` – definuje komunikačné rozhranie služby `AuthorizationRequest` a `AuthorizationResponse`. Súčasťou sú bázové objekty vymedzujúce základné dimenzie `Object, Subject, Action, Time, Modality`.

2. `derived.xsd` – obsahuje vymedzené prvky dimenzie použité pri podpore autorizácie simulovaných bezpečnostných modelov MAC, DAC a RBAC.

Implementácia serverovej časti `AuthorizationCommServiceImpl` webovej služby obsahuje anotácie JAX–WS:

```
@WebService(...)
public class AuthorizationCommServiceImpl{

  @WebMethod(...)
  public AuthorizationResponse authorize(AuthorizationRequest req){
    ...
  }
}
```

10.4 Perzistentná služba HBase

V kapitole špecifikácie technologických požiadaviek 8.4.2 bola ako služba k perzistencii vybraná NoSQL databáza HBase. Toto riešenie poskytuje stabilnú platformu s vlastnosťami relačných databáz – ACID. HBase obsahuje jednoduché API pozostávajúce z troch príkazov, kde konštruktor každého príkazu vyžaduje identifikačný kľúč riadku. Všetky hodnoty a identifikátory, ktoré sa vkladajú do príkazov, musia byť automaticky prevedené do sekvencie bytov:

1. *Put* – vytvorenie alebo upravenie riadku.
2. *Get* – načítanie štruktúry alebo presne vymedzeného stĺpca.
3. *Delete* – odstránenie celého riadku tabuľky alebo riadku v štruktúre

Komponenta implementujúca podporu HBase realizuje DAO sedení a subjektov. V rámci komponenty je vytvorená abstraktná DAO služba `HBaseDaoSupport`, ktorá automatizuje proces práce s tabuľkami – jednoduchým zavolaním metódy `HBaseDaoSupport::getHTable-(tableName)` sa vytvorí referencia na tabuľku a uloží do zoznamu spravovaných tabuliek aktuálneho vlákna. V rámci správy tabuliek je možné nastaviť automatické alebo ručné vykonanie nakešovaných `Put` príkazov.

Dáta v HBase sú uložené až v druhej úrovni štrukturovanej tabuľky. Prvú úroveň tvoria logické štruktúry `column-family`, ktoré musia byť vytvorené osobitne a sú identifikované menom. Záznamy v `column-family` sú tvorené dynamicky.

10.4.1 DAO sedení

Sedenia sú uložené v tabuľke `session` a obsahujú jednu `column-family info`. Identifikačný kľúč riadku je kľúč sedenia.

10.4.2 DAO subjektov

Subjekty sú uložené v tabuľke `subject` a obsahujú tri `column-families base, credential` a `session`. Identifikačný kľúč riadku je identifikačný token subjektu. Sedenia sú ukladané vo forme: čas vytvorenia – kľuč sedenia, podobne dôverné doklady sú uložené vo forme: identifikátor dokladu – zahašovaný doklad.

10.5 Kešovací systém EHCache

K východziemu riešeniu kešovania sedení je použitý populárny systém EHCache, ktorý je postavený na platforme java. Medzi jeho prednosti patrí široká variabilita v použití ako súčasť aplikácie v rámci jednej JVM, samostatný server alebo distribuované riešenie. Nástroj umožňuje konfiguráciu pomocou XML alebo priamo v zdrojovom kóde.

Služba je vnútorne štrukturovaná do samostatných inštancií keší `net.sf.ehcache.Cache`, ktoré sú identifikované menom. Každá inštancia môže obsahovať špecifické nastavenia. Ku kešovaniu je možné použiť operačnú pamäť aj súborové úložište – pevný disk. Kapacita keše sa obmedzuje počtom nakešovaných objektov. Pre keš nad operačnou pamäťou sa kapacita môže vyjadriť percentuálne vzhľadom k poskytnutým zdrojom.

Modul poskytuje implementáciu kešovacieho rozhrania `EhSessionCacheDao`. Súčasťou modulu je východzia konfigurácia služby `default-ehcache.xml` v prípade, ak nie je poskytnutá iná konfigurácia. Rámec vyžaduje existenciu keše identifikovanú menom `sessions`. Kľúč prvku je identifikátor sedenia a hodnota je objekt sedenia.

Špecifickou vlastnosťou integrácie služby je použitá politika k odstraňovaniu objektov pri zaplnení keše. Vybraný algoritmus musí byť prispôsobený charakteru dát, ktoré bude keš spravovať. Sedenia nie sú pevne časovo viazané, ale relatívne k času, kedy boli naposledy sprístupnené. EHCache poskytuje tri základné algoritmy, ktoré je možné rozšíriť o vlastnú implementáciu:

1. *V poslednej dobe najmenej použitý (Less Recently Used)* – východzí algoritmus, ktorý náhodne vyberie vzorku niekoľkých prvkov a vyradí najmenší z nich. Tento algoritmus nie je vhodný ku kešovaniu z dôvodu pravdepodobnosti odstránenia nových sedení.

2. *Najmenej používaný (Less Frequently Used)* – vyradí prvok, ktorý bol najmenej použitý. Algoritmus nie je vhodný z dôvodu odstránenia najnovších sedení.

3. *Najstarší použitý (First In First Out)* – štandardná implementácia fronty. Tento algoritmus môže dávať dobré výsledky za predpokladu veľkej kapacity keše.

Prvý algoritmus *(Less Recently Used)* je náhradou pôvodne (do verzie 1.6, aktálna 2.4) použitého algoritmu *Najdlhšie nepoužitý (Least Recently Used)*. Tento algoritmus je optimálnym riešením, keď už neaktívne sedenia postupne stárnu. Použitie je možné vynútiť nastavením systémovej premennej JVM `net.sf.ehcache.use.classic.lru=true`, čo sa automaticky vykoná v konštruktore `EhSessionCacheDao`.

10.6 Maven moduly

Z dôvodu nasadenia rámca do niekoľkých prostredí, akými sú autorizačná autorita a subsystémy, je rámec rozdelený do niekoľkých Maven modulov, kde každý realizuje špecifickú časť riešenia. Obrázok 10.2 zobrazuje jednotlivé moduly a závislosti medzi nimi. Rámec je hierarchicky vystavaný od obecne používaných funkčností po špecifické vlastnosti:

1. `jd3a-audit` – modul obsahujúci podporu auditu.

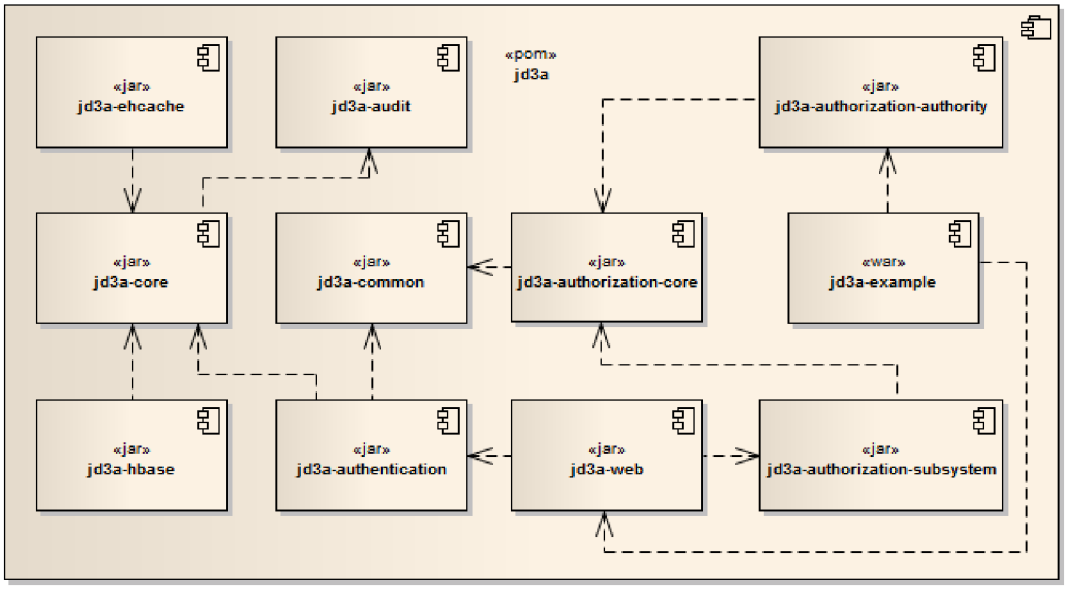
2. `jd3a-core` – bázový modul obsahujúci správu a DAO rozhrania k subjektom a sedeniam.

3. `jd3a-hbase` – implementácia DAO k správe subjektov a sedení nad NoSQL HBase.

4. `jd3a-ehcache` – implementácia kešovania sedení nad systémom EHCache.

5. `jd3a-common` – modul obsahuje spoločné komponenty pre autentifikačný a autorizačný proces. Implementuje bezpečnostný kontext `SecurityContext` a kontext služieb `BeanContext`.

6. `jd3a-authentication` – implementácia jadra a procesov autentifikačného modulu. Súčasťou sú naslúchači realizujúci obmedzenia autentifikačného procesu vyplývajúce z globálneho nastavenia.



Obr. 10.2: Maven moduly rámca jD3A. Z dôvodu prehľadnosti sú vynechané virtuálne moduly určené pre autorizačnú autoritu a subsystém obecne.

7. `jd3a-authorization-core` – autorizácia je rozdelená do troch modulov. Jadrový modul obsahuje definíciu univerzálneho bezpečnostného modulu, východzie autorizačné pravidlá a prvky dimenzií. Súčasťou je implementácia komunikačnej služby realizovanej pomocou webových služieb.

8. `jd3a-authorization-authority` – implementuje autorizačnú autoritu realizujúcu proces autorizácie nad zaregistrovanými autorizačnými pravidlami.

9. `jd3a-authorization-subsystem` – obsahuje podporné autorizačné nástroje nad platformou java.

10. `jd3a-web` – podporný modul pre webové prostredie.

11. `jd3a-example` – spustiteľná webová aplikácia demoštrujúca použitie rámca jD3A.

Kapitola 11

Testovanie

Súčasťou implementácie rámca jD3A je testovanie fundamentálnej funkčnosti, bezpečnostných opatrení, verifikácia a validácia všetkých procesov. Rámec musí byť vzhľadom na technológie a služby, na ktorých závisí, maximálne optimalizovaný z dôvodu vysokej záťaže v produkčnom prostredí. Táto kapitola detailne popisuje činnosti, ktoré zaisťujú tieto požiadavky.

Možnosti spôsobu testovania aplikácií je možné rozdeliť do dvoch kategórií: automatizované a manuálne testovanie. Manuálne testovanie predstavuje najstarší spôsob overovania funkčnosti, no pre moderné aplikácie je nevyhovujúci. Pri vývoji rámca jD3A ako aj služby Takeplace sú použité technológie k automatizovanému testovaniu. Aplikácia je testovaná z niekoľkých hľadísk [20]:

- *Jednotkové testy (Unit testing)* – testujú sa tzv. jednotky (unity), ktoré predstavujú samostatné objekty a jednoduché procesy. Pri testovaní sa využívajú nástroje simulujúce chovanie závislých vrstiev jednotky.
- *Integračné testy (Integration testing)* – testuje sa funkčnosť modulov a systému ako celku. Súčasťou sú testy využívajúce závislé služby (databáze, keše).
- *Výkonostné testy (Performance testing)* – ich účelom je odhaliť hranice stability systému a možné úzke miesta (bottle–neck) pri ktorých strávia procesy najviac času.

11.1 Programovanie riadené testami (Test Driven Development)

Vývoj rámca je riadený testami (TDD) [20], čo umožňuje prístup programovania oproti rozhraniu. Najprv sú implementované jednotkové a integračné testy a následne sa implementuje logická funkčnosť rozhraní.

Účelom testov je overiť funkčnosť vzhľadom na definované vstupy. K správe a realizácii jednotkových a integračných testov je použitá knižnica TestNG. K simulácii (mocking) závislých jednotiek sa využíva knižnica Mockito.

11.2 Výkonostné testy (Performance testing)

Výkonostné testy sú realizované v prostredí domácej siete. Cieľom testovania je overenie funkčnosti celého systému a získanie hodnôt definovaných metrík (počet prihlásení za sekundu, počet autorizácií za sekundu) vzhľadom na možnosti testovacieho prostredia.

Výsledky testovania nie sú smerodatné, a to z dôvodu nepriblíženia sa testovacieho prostredia k produkčnému, ktoré je realiované na AWS VPC. Príprava takéhoto prostredia je finančne a časovo náročná, no k získaniu reálnych štatistík potrebná.

K výkonostným testom bola použitá aplikácia jMeter, ktorá poskytuje nástroje k automatizovanému testovaniu webového prostredia a špecifikácii testovacích scenárov. Nastavenie scenára je súčasťou dokumentácie v prílohe I.

11.2.1 Popis testovacieho prostredia

Tabuľka 11.1 špecifikuje vlastnosti zariadenia, ktoré simuluje distribuovaný systém. Aplikačný kontajner obsahuje aplikáciu realizujúcu autorizačnú autoritu a autentifikačný vstupný bod. Tabuľka 11.2 obsahuje vlastnosti klientského prostredia, ktoré generuje požiadavky. Zariadenia sú pripojené cez 100 Mbit/s linku domácej siete.

Kategória	Subkategória	Popis
Hardvér	RAM	4GB 1066MHz DDR3 SDRAM - 2x2GB
	CPU	Intel Core 2 Duo P8600 (2.4GHz)
	Architektúra	64Bit
Softvér	OS	Mac OS X Snow Leaopard
	Java	Java JRE 1.6.0_22
	Aplikačný kontajner	Apache Tomcat 6.22
	Perzistentné uložište	HBase 0.90.2
	Kešovanie	Vstavaná EHCache 2.4

Tabuľka 11.1: Tabuľka hardvérovej a softvérovej vybavenosti serverovej časti simulujúceho distribuovaný systém.

Kategória	Subkategória	Popis
Hardvér	RAM	2GB DDR2 667MHz (2x1GB dual.ch)
	CPU	Intel Core 2 Duo T7300 (2.0GHz)
	Čipová sada	Intel i965PM Express
	Architektúra	32Bit
Softvér	OS	Windows 7 Professional
	Java	Java JRE 1.6.0_24
	jMeter	jakarta–jmeter–2.3.4

Tabuľka 11.2: Tabuľka hardvérovej a softvérovej vybavenosti stanice generujúcej klientské požiadavky.

11.2.2 Testovanie autentifikácie

K testovaniu autentifikačného procesu sa použila metóda *username–password* a vygenerovalo sa 100 tisíc užívateľov do databáze HBase. Postupným zvyšovaním záťaže sa vykonávala autentifikácia. Počet vygenerovaných požiadaviek je závislý od klientskej stanice, čo skresľuje výsledky. Optimálnym riešením je distribuované použitie niekoľkých klientských staníc paralelne.

Tabuľka obsahuje 11.4 výsledky testovania, ktoré sú graficky prezentované v prílohe G.

Vlastnosť	Hodnota
Počet požiadaviek	10 000
Vzorkovanie	100
Medián	6034 ms
Priemerný čas	6292 ms
Smerodatná odchýlka	3577 ms

Tabuľka 11.3: Tabuľka výsledkov testovania autentifikačnej metódy *username–password*.

Štatistické hodnoty výsledkov sú príliš vysoké. Priebeh testovania zobrazeného na grafe G dokazuje, že v počiatočnej fáze boli hodnoty optimálne, no po určitom čase nastal skok a hodnoty začali rapídne stúpať, čo sa po čase ustálilo do rovnomerného stúpania. Tento fakt môže byť spôsobený postupne zvyšovanou záťažou, ktorú zariadenie simulujúce server výrazne výkonovo ovplyvnilo. Dôležitým faktorom ovplyvňujúcim výsledok je konfigurácia databáze HBase, ktorá k svojmu chodu vyžaduje veľké množstvo zdrojov. V produkčnom prostredí beží HBase na niekoľkých vysoko kapacitných serveroch presne nakonfigurovaných pre daný účel.

11.2.3 Testovanie autorizácie

Subsystém má registrované jednoduché autorizačné pravidlo. Vlastnosti autorizačného aparátu sú závislé na logike autorizačnych pravidiel. Výsledky testovania autorizácie slúžia k identifikácii úzkych miest pri spracovávaní XML správ.

Vlastnosť	Hodnota
Počet požiadaviek	10 000
Vzorkovanie	100
Medián	7 ms
Priemerný čas	8 ms
Smerodatná odchýlka	19 ms

Tabuľka 11.4: Tabuľka výsledkov testovania jednoduchého autorizačného pravidla.

Výsledky výkonostného testu jednoduchého autorizačného pravidla, pozostávajúceho z porovnania kľúča sedenia, zodpovedajú predpokladaným hodnotám, a to z dôvodu nezávislosti procesu na ďalších službách. Výkonostné charakteristiky autorizačného aparátu sú priamo závislé na zložitosti vnútornej logiky pravidla a službách alebo subsystémoch, s ktorými komunikuje. Ako prezentuje graf v prílohe H, spracovanie XML správ štandardom JAXB je rýchle a ani v prípade vysokej záťaže nespôsobuje zahltenie systému.

Kapitola 12

Rozvoj

Rozvoj rámca sa bude prioritne odvíjať od požiadaviek kladených zo služby Takeplace. V tomto čase je rámec jD3A zaregistrovaný ako projekt na portále SourceForge [16], kde je dostupný k stiahnutiu. V rámci procesu prieniku do Java komunity bude rámec zaregistrovaný ako Maven projekt a zavedený do verejných repozitárov.

V nadchádzajúcom období bude rámec rozvíjaný hlavne v týchto oblastiach (zoradené podľa priority):

1. Podpora kešovacích technológií – implementácia modulu `jd3a-memcached` umožňujúceho kešovanie nad službou Memcached [14], ktorá sa má stať hlavnou kešovacou technológiou v prostredí Takeplace.

2. Podpora SQL databází – modul `jd3a-sql` implementujúci DAO správy subjektov a sedení nad relačnými databázami. Úlohou modulu je zvýšiť záujem a dostupnosť integrácie rámca Java komunite z dôvodu rozšírenosti relačných databáz, ktoré sú majoritou pri perzistencii dát.

3. Rozvoj distribuovaných auditných nástrojov – modul `jd3a-scribe` implementujúci napojenie na distribuovaný logovací server Scribe [12] spoločnosti Facebook.

4. Bezpečnosť – rámec v tomto čase poskytuje základné bezpečnostné opatrenia plynúce z doporučení OWASP. V rámci zvýšenia bezpečnosti budú realizované:

   (a) Autentifikačná metóda OAuth [17] a jej variácia OAuthMobile.

   (b) Aplikačný autentifikačný filter realizujúci ochranu proti distribuovanému útoku hrubou silou.

   (c) Autorizačná požiadavka môže vyžadovať autentifikáciu definovanou metódou v prípade prístupu do kritických subsystémov.

5. Rozvoj podpory na iných platformách – autorizačný proces využíva webové služby, ktoré umožňujú platformovo nezávislú komunikáciu. Prioritne bude podpora zameraná na platformy, ktoré budú integrované do služby Takeplace.

Kapitola 13

Záver

Cieľom práce bolo vytvorenie systému na kontrolu prístupu v systéme Takeplace s použitím viacdimenzionálneho bezpečnostného modelu. Kontrola prístupu je úzko zviazaná s procesom autentifikácie subjektov a autorizácie ich akcií v rámci systému. Tieto procesy nemôžu existovať samostatne z dôvodu zdielanej správy identít. Veľký dôraz je kladený hlavne na bezpečnosť týchto procesov.

Bola vykonaná analýza bezpečnostných hrozieb a ich opatrení, definovaná autentifikácia, autorizácia a spôsoby kontroly prístupu s použitím moderných bezpečnostných modelov. Na ich základe sa unifikoval a definoval univerzálny bezpečnostný model. Nasledovala špecifikácia požiadaviek na cieľový systém kontroly prístupu v službe Takeplace. Vzhľadom na chýbajúce riešenie kontroly prístupu v distribuovaných prostrediach bol navrhnutý a implementovaný rámec jD3A, ktorý pokrýva všetky stanovené ciele a požiadavky. Zadaný subsystém je následne vystavaný nad týmto rámcom.

Pri návrhu rámca bolo vynaložené veľké úsile k zaisteniu nezávislosti jednotlivých komponent a použitých technológií a služieb. Vďaka týmto vlastnostiam a univerzálnemu bezpečnostnému modelu je problém kontroly prístupu v distribuovanom prostredí výrazne minimalizovaný. Rámec je poskytovaný ako Open Source, čo môže propagovať službu Takeplace v obore vývoja webových informačných systémov.

Rámec má široký technologický záber využívajúci aspektovo orientované programovanie, čoraz viac rozšírené NoSQL databáze a distribuované kešovacie systémy. Modulárna architektúra umožňuje nezávislosť v použitých technológiach. Komunikácia s využitím webových služieb dovoľuje autorizovať entity bez ohľadu na použitú vývojovú platformu subsystému. Všetky tieto vlastnosti rámca naznačujú ideálne predpoklady k jeho ďalšiemu rozvoju v budúcnosti v rámci služby Takeplace alebo komunity, ktorá sa okolo rámca vytvorí.

Integráciou rámca do služby Takeplace sa zvýši jej bezpečnosť. Využitím vlastností univerzálneho modelu a nezávislej komunikačnej služby sa odstráni pretrvávajúci problém autorizácie špecifických požiadaviek nad množstvom platforiem. Rámec umožní ďalší rozvoj v oblasti bezpečnosti služby Takeplace ako aj distribuovaných systémov obecne.

Významným prínosom pri spracovaní tejto práce bola možnosť štúdia známych technológií do väčšej hĺbky a vloženie znalostí získaných pri štúdiu a tvorbe systému Takeplace do návrhu a realizácie rámca.

Literatúra

[1] Acemcee, s.r.o.: Takeplace. [online], 2008, [cit. 2010-12-25].
URL http://www.takeplace.eu

[2] Amazon Web Services: Virtual Private Cloud. [online], [cit. 2011-05-20].
URL http://aws.amazon.com/vpc/

[3] Avinash, L.; Prashant, M.: Cassandra – A Decentralized Structured Storage System. *SIGOPS Oper. Syst. Rev.*, ročník 44, apr 2010: s. 35–40, ISSN 0163-5980.
URL http://doi.acm.org/10.1145/1773912.1773922

[4] Castano, S.; Fugini, M. G.; Martella, G.; aj.: *Database Security*. Acm Press Books, 1996, ISBN 978-0201593754.

[5] Codd, E. F.: A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, ročník 13, č. 6, 1970: s. 377–387.

[6] Coward, D.; Yoshida, Y.: JSR-000154 JavaTM Servlet 2.4 Specification. Technická zpráva, Java Community Process, 2011, [cit. 2011-05-20].
URL http://jcp.org/aboutJava/communityprocess/mrel/jsr154/index.html

[7] Crockford; Douglas: RFC 4627: The application/json Media Type for JavaScript Object Notation (JSON). Technická zpráva, IETF, [cit. 2010-12-28].
URL http://tools.ietf.org/html/rfc4627

[8] Eclipse Foundation: AspectJ. [online], 2011, [cit. 2011-05-15].
URL http://www.eclipse.org/aspectj/

[9] Elrad, T.; Filman, R. E.; Bader, A.: Aspect-oriented programming: Introduction. *Commun. ACM*, ročník 44, oct 2001: s. 29–32, ISSN 0001-0782.
URL http://doi.acm.org/10.1145/383845.383853

[10] Erik, C.; Francisco, C.; Greg, M.; aj.: Web Service Definition Language (WSDL). Technická zpráva, W3C, Marec 2001.
URL http://www.w3.org/TR/wsdl

[11] Evans, D. L.; Bond, P. J.; Bement, A. L.: Security Requirements for Cryptographic Modules (**FIPS 140-2**). Technická zpráva, National Institute of Standards and Technology, 2001, [cit. 2010-12-28].
URL http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

[12] Facebook: Facebook Scribe. [online], [cit. 2011-05-05].
URL https://github.com/facebook/scribe#readme

[13] Fialli, J.; Vajjhala, S.: JSR 222: JavaTM Architecture for XML Binding (JAXB) 2.0. Technická zpráva, Java Community Process, 2011, [cit. 2011-05-20].
URL http://jcp.org/aboutJava/communityprocess/final/jsr222/index.html

[14] Fitzpatrick, B.; komunita: Memcached. [online], 2011, [cit. 2010-12-28].
URL http://memcached.org/

[15] Google: Google Web Toolkit Remote Procedure Calls. [online], 2010, [cit. 2010-12-28].
URL http://code.google.com/intl/cs-CZ/webtoolkit/doc/2.0/DevGuideServerCommunication.html

[16] Grešša, P.: SourceForge – Java Distributed AAA. [online], [cit. 2011-05-20].
URL http://sourceforge.net/projects/jd3a/

[17] Hammer-Lahav, E.: The OAuth 1.0 Protocol Specification. Technická zpráva, IETF, Február, [cit. 2011-05-20].
URL http://tools.ietf.org/html/draft-hammer-oauth-10

[18] Hanáček, P.; Staudek, J.: Správa identity. V *Sborník konference DATAKON 2005*, Masaryk University, 2005, ISBN 80-210-3813-6, s. 123–146.

[19] Havey, M.: *Essential business process modeling*. O'Reilly Media, 2005, ISBN 978-0596008437.

[20] Janzen, D.; Saiedian, H.: Test-Driven Development: Concepts, Taxonomy, and Future Direction. *Computer*, ročník 38, 2005: s. 43–50, ISSN 0018-9162.

[21] JBoss: JBoss Cache. [online], 2010, [cit. 2010-12-28].
URL http://community.jboss.org/wiki/JBossCacheOfficialDocumentation

[22] Kohlert, D.: JSR 224: JavaTM API for XML-Based Web Services (JAX-WS) 2.0. Technická zpráva, Java Community Process, 2011, [cit. 2011-05-20].
URL http://jcp.org/aboutJava/communityprocess/final/jsr224/index.html

[23] Kristol, D.; Montulli, L.: RFC 2965: HTTP State Management Mechanism. Technická zpráva, IETF, Október 2000, [cit. 2010-12-28].
URL http://www.ietf.org/rfc/rfc2965.txt

[24] Kuba, M.: Tutoriál Web Services. V *Konference EurOpen.CZ*, Masaryk University, 2005.

[25] Leach, P.; Mealling, M.; Salz, R.: RFC 4122: A Universally Unique IDentifier URN Namespace. Technická zpráva, IETF, 2005, [cit. 2010-12-28].
URL http://tools.ietf.org/html/rfc4122

[26] Luck, G. R.; Luck, G. R.; Purdy, C.: JSR 107: JCACHE - Java Temporary Caching API. Technická zpráva, Java Community Process, 2001, [cit. 2011-04-17].
URL http://jcp.org/en/jsr/summary?id=107

[27] Open Web Application Security Project: AntiSamy Project. [online], [cit. 2010-12-28].
URL http://www.owasp.org/index.php/Category:OWASP_AntiSamy_Project

[28] Open Web Application Security Project: Application Security Verification Standard Project. [online], [cit. 2010-12-28].
URL http://www.owasp.org/index.php/Category:OWASP_Application_Security_Verification_Standard_Project

[29] Open Web Application Security Project: Open Web Application Security Project. [online], [cit. 2010-12-28].
URL http://www.owasp.org

[30] Open Web Application Security Project: Software Assurance Maturity Model. [online], [cit. 2010-12-28].
URL http://www.owasp.org/index.php/Category:Software_Assurance_Maturity_Model

[31] Open Web Application Security Project: Top Ten Project. [online], [cit. 2010-12-28].
URL http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

[32] Open Web Application Security Project: WebScarab Project. [online], [cit. 2010-12-28].
URL http://www.owasp.org/index.php/Category:OWASP_WebScarab_Project

[33] OpenID Foundation: OpenID. [online], 2006, [cit. 2010-12-25].
URL http://openid.net

[34] Oracle: Java Platform, Enterprise Edition (Java EE) Technical Documentation. [online], 2010, [cit. 2010-12-30].
URL http://download.oracle.com/javaee/

[35] Oracle: Oracle Database. [online], 2010, [cit. 2010-12-28].
URL http://www.oracle.com/us/products/database/index.html

[36] Organization for the Advancement of Structured Information Standards: Security Assertion Markup Language. [online], 2007, [cit. 2010-12-25].
URL http://saml.xml.org/

[37] Ort, E.; Mehta, B.: Java Architecture for XML Binding (JAXB). [online], [cit. 2011-05-20].
URL http://www.oracle.com/technetwork/articles/javase/index-140168.html

[38] OWASP: Enterprise Security API. online.
URL http://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API

[39] Paul, B.; Ashok, M. (editori): *XML Schema Part 2: Datatypes*. W3C Recommendation, W3C, druhé vydanie, Október 2004.
URL http://www.w3.org/TR/xmlschema-2/

[40] Púdelka, I.: *Aspektovo orientované programovanie a jeho podpora*. Diplomová práca, Masarykova Univerzita – Fakulta Informatiky, 2010.

[41] Pecinovský, R.: *Návrhové vzory - 33 vzorových postupů pro objektové programování*. COMPUTER PRESS, 2007, ISBN 978-80-251-1582-4.

[42] Procházka, F.: eTrium – Univerzální nástroj pro správu přístupových práv. V *Sborník konference DATAKON 2003*, Masarykova Univerzita v Brňe, Fakulta informatiky, Masaryk University, 2003, ISBN 80-210-3215-4, s. 265–275.

[43] SpringSource: Spring Security. [online], 2011, [cit. 2011-05-12].
URL http://static.springsource.org/spring-security/site/

[44] Steimann, F.; Mayer, P.; Hannover, U.; aj.: Patterns of interface-based programming. *Journal of Object Technology*, ročník 4, 2005: s. 75–94.

[45] Sun Microsystems Computer Corporation: *SunSHIELD Basic Security Module Guide*. Sun Microsystems Computer Corporation, 1994, [cit. 2011-05-20].

[46] Terracotta: Ehcache–The Most Widely Used Java Cache. [online], 2011, [cit. 2010-12-28].
URL http://ehcache.org/

[47] The Apache Software Foundation: Apache HTTP Server Version 2.2 – Proxy Module. [online], 2011, [cit. 2011-05-21].
URL http://httpd.apache.org/docs/2.2/mod/mod_proxy.html

[48] The Apache Software Foundation: Apache Maven Project. [online], 2011, [cit. 2010-12-30].
URL http://maven.apache.org/

[49] The Apache Software Foundation: Apache Shiro. [online], 2011, [cit. 2011-05-12].
URL http://shiro.apache.org/

[50] Tom, B.; Steve, C.; Luc, C.; aj.: UDDI Version 3.0.2. Technická zpráva, OASIS, Október 2004.
URL http://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm

[51] W3C: Requirements for a Global Identity Management Service. [online], 2001, [cit. 2010-12-28].
URL http://www.w3.org/2001/03/WSWS-popa/paper57

[52] W3C: SOAP Version 1.2 Part 0: Primer (Second Edition). [online], apr 2007, [cit. 2010-12-28].
URL http://www.w3.org/TR/soap12-part0/

[53] Witty, R. J.; Allan, A.; Enck, J.; aj.: Identity and Access Management Defined. [online], 2003, [cit. 2010-12-28].
URL
http://www85.homepage.villanova.edu/timothy.ay/DIT2160/IdMgt/118281.pdf

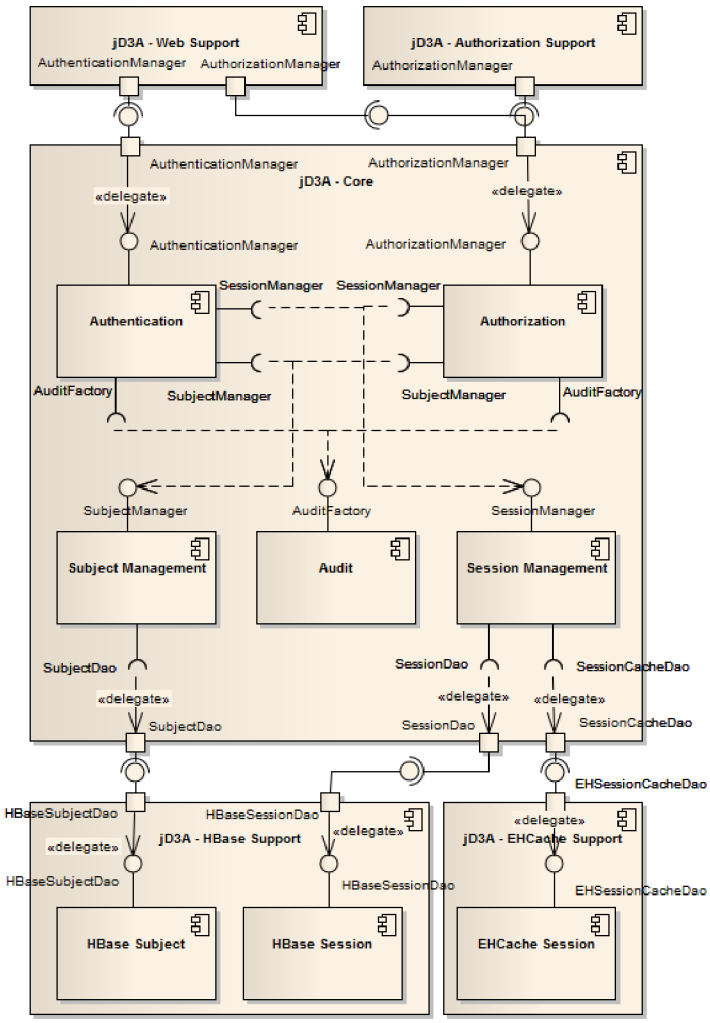
Dodatok A

Diagram tried jD3A

Z dôvodu veľkosti je diagram uložený na dodanom CD v `/doc/UML/jd3a-class.png`.

Dodatok B

Komponentový diagram jD3A



Dodatok C

Detailná špecifikácia prípadov použití AAA

Vlastnosť	Špecifikácia
ID	1:CreateAccount
Názov	Vytvorenie účtu
Stručný popis	Vytvorenie účtu neregistrovanej entite
Primárni aktéri	Neznáma entita
Alternatívni aktéri	AAA subsystém
Vstupné podmienky	žiadne
Scenár	1. Entita vloží užívateľské meno a heslo 2. Systém overí či je užívateľské meno voľné 3. AK je uživateľské meno už registrované výnimka 4. Systém vytvorí nový účet 5. Systém nastaví účtu príznak *neaktívny účet* 6. Systém vytvorí aktivačný kód 7. Systém perzistuje nový účet 8. Systém vytvorí auditový záznam 9. Systém pošle vlastníkovy účtu aktivačný kód
Výstupné podmienky	Vytvorený nový neaktívny užívateľský účet.
Výnimky	Ak účet už existuje.

Vlastnosť	Špecifikácia
ID	2:ActivateEntity
Názov	Aktivácia účtu
Stručný popis	Aktivácia neaktívneho účtu
Primárni aktéri	Neznáma entita s aktivačným kódom, Administrátor systému
Alternatívni aktéri	AAA subsystém
Vstupné podmienky	Existujúci účet entity
Scenár	1. Aktér vloží aktivačný kód 2. Systém overí aktivačný kód 3. AK aktivačný kód neexistuje výnimka 4. AK je aktivačný kód zviazaný s aktívym účtom výnimka 5. Systém vytvorí auditný záznam 6. Systém aktivuje účet
Výstupné podmienky	Aktívny užívateľský účet entity
Výnimky	• Ak aktivačný kód neexistuje • Ak je aktivačný zviazaný s aktívnym účtom

Vlastnosť	Špecifikácia
ID	3:Authenticate
Názov	Autentifikácia
Stručný popis	Autentifikácia entity
Primárni aktéri	Neznáma entita
Alternatívni aktéri	AAA subsystém
Vstupné podmienky	Existujúci účet entity

Vlastnosť	Špecifikácia
Scenár	1. Aktér vloží dôverné informácie a identifikačný token do autentifikačnej požiadavky 2. Systém na základe požiadavky zistí metódu autentifikácie 3. AK neexistuje metóda autentifikácie výnimka 4. Systém na základe identifikačného tokenu načíta subjekt 5. AK neexistuje subjekt výnimka 6. Systém overí autentifikačnou metódou dôverné doklady 7. AK doklady neodpovedajú perzistovaným dokladom výnimka 8. Systém vytvorí nové sedenie zpárované so subjektom 9. Systém vytvorí auditný záznam 10. Systém predá sedenie do klientského prostredia
Výstupné podmienky	Autentifikovaný užívateľ systému
Výnimky	• Ak neexistuje metóda autentifikácie • Ak neexistuje subjekt vzhľadom na poskytnutý identifikačný token • Ak doklady neodpovedajú perzistovaným dokladom

Vlastnosť	Špecifikácia
ID	4:UpdateCredentials
Názov	Úprava dôverných dokladov
Stručný popis	Úprava dôverných dokladov vykonaná samoobslužne alebo administrátorom
Primárni aktéri	Autentifikovaný vlastník účtu, Administrátor systému
Alternatívni aktéri	AAA subsystém
Vstupné podmienky	Existujúci účet entity

Vlastnosť	Špecifikácia
Scenár	1. Systém načíta subjekt entity 2. Aktér špecifikuje dôverný doklad ktorý chce upraviť 3. Aktér vyberie novú hodnotu dôverného dokladu 4. Systém overí špecifické vlastnosti dokladu 5. AK nový doklad nerešpektuje povinné vlastnosti dokladu výnimka 6. Systém nahradí starý dôverný doklad novým 7. Systém uloží zmenu do perzistentného uložišťa 8. Systém vytvorí auditný záznam
Výstupné podmienky	Zmenený dôverný doklad subjektu
Výnimky	• Ak nový dôverný doklad neobsahuje povinné parametre definované autentifikačnou metódou ktorá s dokladom pracuje.

Vlastnosť	Špecifikácia
ID	5:AutorizeBusinessAction
Názov	Autorizácia akcie
Stručný popis	Autorizácia akcie subjektu nad subsystémom
Primárni aktéri	Autentifikovaný subjekt, AAA subsystém, Subsystém
Alternatívni aktéri	žiadny
Vstupné podmienky	žiadne

Vlastnosť	Špecifikácia
Scenár	1. Subjekt požiada o vykonanie služby nad subsystémom 2. Subsystém vymedzí dimenzie a vytvorí autorizačnú požiadavku 3. Subsystém pošle požiadavku AAA subsystému 4. AAA subsystém vyhodnotí požiadavku 5. AAA subsystém vytvorí auditný záznam 6. AAA subsystém pošle autorizačnú odpoveď 7. Subsystém spracuje odpoveď 8. AK nie je odpoveď autorizovaný výnimka 9. Subsystém vykoná požadovanú službu
Výstupné podmienky	Vykonaná služba
Výnimky	• Ak subjekt nie je autorizovaný k vykonaniu služby

Vlastnosť	Špecifikácia
ID	6:BlockEntity
Názov	Blokovanie entity
Stručný popis	Zakázanie prístupu entity do systému
Primárni aktéri	Administrátor
Alternatívni aktéri	AAA subsystém
Vstupné podmienky	žiadne
Scenár	1. Administrátor identifikuje subjekt 2. Systém načíta subjekt 3. Systém nastaví príznak blokovania 4. Systém uloží zmenu subjektu 5. Systém vytvorí auditný záznam

Vlastnosť	Špecifikácia
Výstupné podmienky	Identifikovaná entita je blokovaná
Výnimky	žiadne

Vlastnosť	Špecifikácia
ID	7:ArchiveEntity
Názov	Archivácia entity
Stručný popis	Archivácia entity
Primárni aktéri	Administrátor
Alternatívni aktéri	AAA subsystém
Vstupné podmienky	žiadne
Scénár	1. Administrátor identifikuje subjekt 2. Systém načíta subjekt 3. Systém nastaví príznak archivácie 4. Systém uloží zmenu subjektu 5. Systém vytvorí auditný záznam
Výstupné podmienky	Identifikovaná entita je archivovaná
Výnimky	žiadne

Vlastnosť	Špecifikácia
ID	8:AuditManagement
Názov	Správa auditných
Stručný popis	Archivácia entity
Primárni aktéri	Administrátor
Alternatívni aktéri	AAA subsystém
Vstupné podmienky	žiadne

Vlastnosť	Špecifikácia
Scenár	1. Administrátor určí počet, kategóriu a vlastnosti auditného záznamu 2. Systém načíta záznamy
Výstupné podmienky	Systém načítal požadované záznamy
Výnimky	žiadne

Dodatok D

OWASP–Session Management analýza

Doporučenie	Stav v jD3A
All applications should share a well–debugged and trusted session management mechanism.	Implementácia obsahuje automatické testy, podrobný a namodelovaný návrh.
All session identifiers should be sufficiently randomized so as to not be guessable.	Ku generovaniu sa používa UUID.
All session identifiers should use a key space of at least XXXX bits.	Ku generovaniu sa používa UUID, ktoré generuje 128 bitové identifikátory.
All session identifiers should use the largest character set available to it.	Závislé na UUID generátoru.
Sessions SHOULD timeout after 5 minutes for high-value applications, 10 minutes for medium value applications, and 20 minutes for low risk applications.	Nastaviteľné cez globálnu konfiguráciu rámca.
All session tokens in high value applications SHOULD be tied to a specific HTTP client instance (session identifier and IP address).	Táto kontrola spadá do základnej ochrany ukradnutia sedenia rámca.
Application servers SHOULD use private temporary file areas per client/application to store session data.	Cookies generované webovou podporou rámca využíva privátne cookies, ktoré sa mažú po zatvorení prehliadača.
All applications SHOULD use a cryptographically secure page or form nonce in a hidden form field.	Implementovaná len podpora vynútenej komunikácia v SSL.
Each form or page nonce SHOULD be removed from the active list as soon as it is submitted.	Stránky ani formuláre nie sú žiadnym spôsobom kontrólované na prijatie/odoslanie.
Session ID values submitted by the client should undergo the same validation checks as other request parameters.	Identifikátor sedenia je validovaný pri načítavaní z keše resp. perzistentného uložišťa.

Doporučenie	Stav v jD3A
High value applications SHOULD force users to re-authenticate before viewing high-value resources or complete high-value transactions.	Táto funkčnosť nie je podporovaná. Architektúra umožnuje ovplyvniť proces autorizácie týmto spôsobom.
Session tokens should be regenerated prior to any significant high value transaction.	Táto funkčnosť nie je podporovaná. Architektúra umožnuje ovplyvniť proces autorizácie týmto spôsobom.
In high-value applications, session tokens should be regenerated after a certain number of requests.	Táto funkčnosť nie je podporovaná. Architektúra umožnuje ovplyvniť proces autorizácie týmto spôsobom.
In high-value applications, session tokens should be regenerated after a certain period of time.	Táto funkčnosť nie je podporovaná. Architektúra umožnuje ovplyvniť proces autorizácie týmto spôsobom.
For all applications, session tokens should be regenerated after a change in user privilege.	Táto funkčnosť nie je podporovaná. Architektúra umožnuje ovplyvniť proces autorizácie týmto spôsobom.
Applications should log attempts to continue sessions based on invalid session identifiers.	Auditný modul rámca zaznamenáva tieto udalosti – ak je k tomu nakonfigurovaný.
Applications should, if possible, conduct all traffic over HTTPS.	Rámec umožňuje vyžadovať len SSL komunikáciu.
If applications use both HTTP and HTTPS, they MUST regenerate the identifier or use an additional session identifier for HTTPS communications.	Táto funkčnosť nie je implementovaná.

Dodatok E

OWASP–Authentication analýza

Doporučenie	Stav v jD3A a Takeplace
All applications within your organization SHOULD share a well–debugged and trusted authentication mechanism if possible	Aplikovaním rámca jD3A do distribuovanáho prostredia je táto požiadavka splnená.
All secured functions and secured resources SHOULD be protected by a common authentication mechanism within the one application	Rámec podporuje autentifikáciu viac metódami. Obmedzenie prístupu ku špecifikovanému zdroju presne definovanou metódov je súčasťou rozvoja rámca.
All applications MUST use the lowest possible privilege service account to access back end systems such as directories, web services, database and message queues	Závislé na nastaveniu prostredia. Rámec rozširuje možnosti oddelenia jednotlivých priestorov vďaka DAO.
Credentials SHOULD be transmitted only over encrypted links, particularly weak authentication mechanisms such as passwords	Všetky operácia nad rámcom jD3A môžu vyžadovať zabezpečenú komunikáciu.
Credentials MUST be stored after being one-way hashed and salted using acceptable hashing algorithms	Správca subjektov vždy generuje pre nový účet soľ a je aplikovaná hešovacia funkcia s minimálnym počtom iterácií 1.
Credential stores SHOULD implement configurable settings for thresholds, lockouts, password complexity and alerts	jD3A podporuje tvrdý (administrátorský) a mäkký zámok, nastavenie uzamknutia účtu po N neúspešných pokusoch. Výstrahy nie sú podporované v tejto verzii.
Credential stores SHOULD be designed to implement several hashing algorithms as these will be replaced soon and as change is inevitable, your application should plan today for this transition	Použitý hešovací algoritmus je nastaviteľný v konfiguračnom súbore. Funguje na princípe Java MessageDigest, ktorý je nativne rozširovaný s novými verziami platformy.
Applications SHOULD have the facility to alert the user as to failed login attempts and offer to allow them to change their password (if applicable)	Zmena dokladov je závislá na systéme aplikovanom nad rámcom jD3A. Takeplace AAA subystém implementuje samoobslužnú zmenu dokladov.

Doporučenie	Stav v jD3A a Takeplace
Applications SHOULD have the facility to notify the user of their last logged in time, and subsequently report a fraudulent login if they disagree with that date and time	Rámec zaznamenáva údaje o naposledy úspešnom resp. neúspešnom prihlásení. Upozornenie musí byť realizované v systéme nad rámcom. Takeplace AAA nemá zamýšlanú takúto funkčnosť.
Authentication and registration processes, particularly login failures, SHOULD provide no information as to if an account exists or password or is wrong. A single error message for the end user covering both scenarios is more than adequate	Výsledkom autentifikačného procesu je úspech alebo neúspech. Registrácia je realizovaná nad rámcom. Takeplace AAA kontróluje unikátnosť emailu a dáva informáciu o výsledku autentifikácie. Je možné použiť prekladač výnimiek k prispôsobeniu procesu.
All pages SHOULD have an effective logout button on every single page in a common location	UI Takeplace je prispôsobené tejto požiadavke.
Applications SHOULD possess administrative functions to detail and manage never logged in accounts, idle accounts, and accounts that have been administratively- or soft- locked	Dátové štruktúry obsahujú tieto informácie. Tieto funkcie sú naplánované v ďalšom rozvoji nástroja.
Passwords MUST be easily changed. Applications MAY include password strength indicators or provide a random password generator function	Je súčasťou systému Takeplace. Poslaním emailu s kódom platným po obmedzený čas.
There SHOULD be a logical difference between administrative lockout and failed login lockout, so that re-enabling all users en masse does not unlock administratively locked users	Rámec rozlišuje mäkký (soft) a tvrdý (hard, administrative) zámok.
Medium value applications SHOULD and High value applications MUST provide a mechanism to re-authenticate or transaction sign high value transactions	Táto funkčnosť nie je podporovaná. Architektúra umožnuje ovplyvniť proces autorizácie týmto spôsobom.
Applications MUST NOT store any secret part of the credential in the clear (passwords or questions and answers if implemented)	Všetky dôverné doklady sú hašované a solené.
Applications MUST NOT expose the credential in untrusted locations, such as cookies, headers or hidden fields	Doklady sa nikdy nedostanú mimo autentifikačný proces.
Applications MUST NOT implement CAPTCHA as there is case law against them with respect to universal access and ineffective	Takeplace neimplementuje CAPTCHu ako ochranu. Každý nový užívateľ musí potvrdiť svoju registráciu vzhľadom k aktivačnému kódu, ktorý po registrácii obdrží.

Doporučenie	Stav v jD3A a Takeplace
Applications MUST NOT implement questions and answers as they are contrary to most privacy regimes and ineffective	Takeplace neimplementuje túto funkčnosť. Rámec nepodporuje tento spôsob overenia identity subjektu.
Applications SHOULD NOT rely on infrastructure authentication, such as REFERER headers or the client's DNS or IP address as these can be faked	Autentifikácia nezávisí na žiadnej spomenutej kontrole. Avšak overenie a ochrana voči ukradnutiu sedenia porovnáva IP adresu klienta pri autentifikácii a následnej autorizácii.

Dodatok F

Konfiguračné nastavenia rámca

Globálne nastavenie	Datový typ	Popis
LockAccountTreshold	Integer	Súčasťou dátovej štruktúry subjektu je počet neúspšených pokusov o autentifikáciu. V prípade prekročenia tohto prahu sa účet uzamkne mäkkým zámkom.
FailedAttemptTimetouts	Integer[]	Pole hodnôt predstavujúcich časový interval v sekundách medzi neúspešnými pokusami o autentifikáciu, ktorý musí byť dodržaný. Príklad: 5,10,15. Prvý neúspech možné prihlásenie po 5 sekundách, druhý neúspech po 10 sekunách, každý ďalší neúspech po 15 sekundách.
BlockIPTreshold	Integer	Hodnota prahu udavajúci počet neúspešných pokusov o autentifikáciu po ktorom sa v rámci aplikačného firewallu začnú odmietať všetky ďalšie pokusy o autentifikáciu z IP adresy.
DefaultSessionTimeout	Integer	Východzie nastavenie doby aktívneho sedenia v sekundách. Rámec nevyužíva vstavanej funčnosti JEE a ich sedení.
DefaultHashAlgorithm	String	Východzia hešovacia funkcia. Musí byť podporovaná verziou Java platformy `MessageDigest`. Zmena počas životného cyklu aplikácie môže znefunkčniť doterajšie existujúce doklady.
DefaultHashIterations	Integer	Východzí počet iterácií hešovacej funkcie - minimálne 1. Zmena počas životného cyklu aplikácie môže znefunkčniť doterajšie existujúce doklady.

Nastavenie prístupového bodu	Datový typ	Popis
ID	String	Identifikátor prístupového bodu.

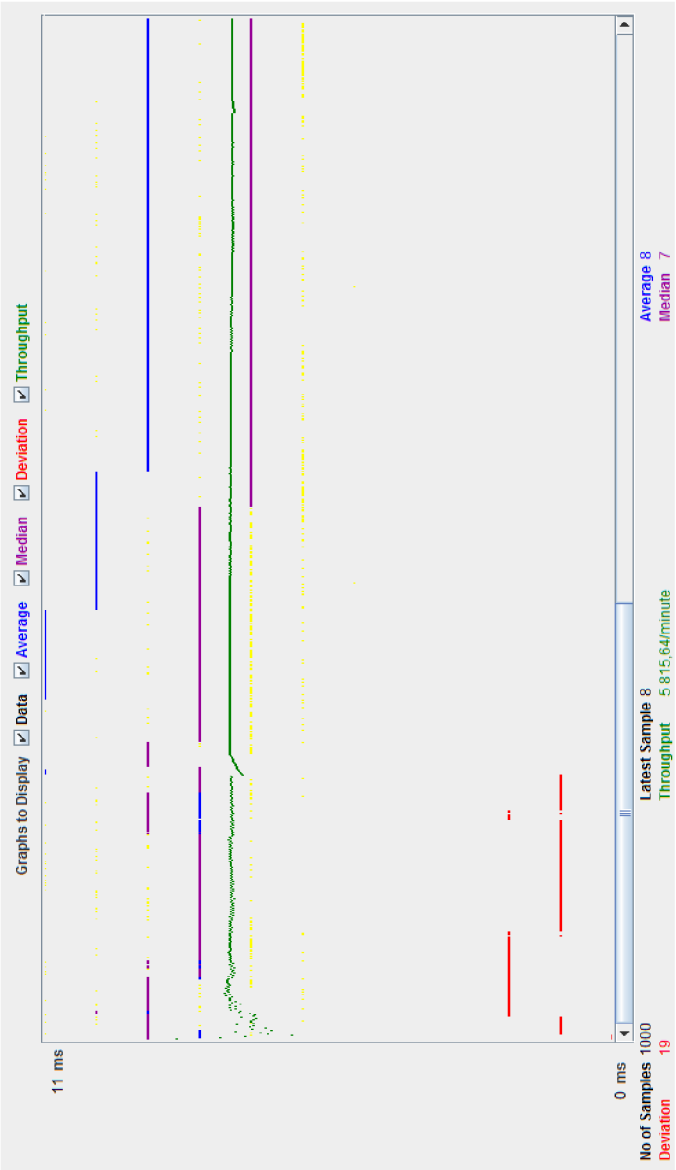
Nastavenie prístupového bodu	Datový typ	Popis
RealmForceSsl	Boolean	Prístupový bod bude prijímať len požiadavky zabezpečenej komunikácie.
RealmContextPath	String	Kontextová časť URL, ktorá je namapovaná na prístupový bod.
RealmName	String	Názov autentifikačnej metódy. Vyhodnotenie použitia je závislé od použitého `RealmProvider` implementácie.
*RealmSessionTimeout	Integer	Nastavenie doby aktívneho sedenia v sekundách.
*RealmHashAlgorithm	String	Nastavenie hešovacej funkcie. Musí byť podporovaná verziou Java platformy `MessageDigest`. Zmena počas životného cyklu aplikácie môže znefunkčniť doterajšie existujúce doklady.
*RealmHashIterations	Integer	Nastavenie počtu iterácií hešovacej funkcie - minmálne 1. Zmena počas životného cyklu aplikácie môže znefunkčniť doterajšie existujúce doklady.
RealmSuccessRedirect	String	Kontextová URL na ktorú bude presmerovaný subjekt po úspešnej autentifikácii.
RealmFailureRedirect	String	Kontextová URL na ktorú bude presmerovaný subjekt po neúspešnej autentifikácii
* – Ak nie je uvedené použije sa globálne nastavenie.		

Dodatok G

Graf autentifikačného testu

Dodatok H

Graf autorizačného testu



Dodatok I

Obsah CD

Súčasťou odovzdanej dokumentácie je priložené CD so súborovou štruktúrov:

- `bin` – obsahuje batch a shell skripty k vystavaniu (build) rámca s použitím knižníc z adresára `lib`.
- `doc` – adresár obsahuje dokumentáciu k rámcu:
  - `Tex` – adresár so zdrojovým kódom tejto dokumentácie vo formáte tex.
  - `UML` – súbor všetkých UML diagramov vzťahujúcich sa k práci.
  - `dp-print.pdf` – dokumentácia vo formáte PDF určeného k tlačeniu.
  - `dp.pdf` – dokumentácia vo formáte PDF s aktívnymi odkazmi.
- `lib` – adresár obsahuje všetky knižnice potrebné k vystavaniu (build) rámca jD3A. Knižnice je nutné zaviesť do lokálneho Maven repozitára, alebo použiť nástroje z adresára `bin`.
- `src` – obsahom adresára je zdrojový kód k verzii rámca 1.0.0. Aktuálne zdrojové kódy rámca je možné získať podľa inštrukcií v [16].
- `test` – súčasťou je konfiguračný súbor k aplikácii jMeter `jD3A-Test-Plan.jmx` obsahujúci testovací plán k automatizovanému testovaniu autentifikácie a webovej služby autorizácie.
- `README.txt` – detailný popis súborovej štuktúry média.