

PCI DSS Standard

I grupa

1. Svi delovi sistema moraju biti zaštićeni od neautorizovanog pristupa

Da bi se pristupilo sistemu i njegovim funkcionalnostima, korisnik mora biti autentifikovan. Za autentifikaciju i autorizaciju postoji poseban mikroservis - **Auth Service** koji izdaje JWT token koji se nalazi u Cookie-u pomoću kojeg je kasnije omogućena autentifikacija korisnika.

Primer login funkcije koja vraća Cookie prikazan na Response objektat:

```
public LoginResponse login(LoginRequest loginRequest, HttpServletResponse response) throws Exception {
    Authentication authentication = authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(
        loginRequest.getEmail(),
        loginRequest.getPassword()
    ));
    User user = (User) authentication.getPrincipal();

    SecurityContextHolder.getContext().setAuthentication(authentication);
    String accessToken = tokenProvider.createAccessToken(authentication);
    Integer tokenExpirationSeconds = appProperties.getAuth().getTokenExpirationSeconds();
    CookieUtils.addCookie(
        response,
        TokenAuthenticationFilter.ACCESS_TOKEN_COOKIE_NAME,
        accessToken,
        tokenExpirationSeconds
    );

    Long expiresAt = tokenProvider.readClaims(accessToken).getExpiration().getTime();

    return new LoginResponse(user.getId().toString(), expiresAt, user.getRole());
}
```

Kada zahtev stigne na PSP, potrebno je autentifikovati korisnika kako bi se zahtev prosledio odgovarajućem mikroservisu za plaćanje.

Verifikacije korisnika na PSP mikroservisu:

```
try {
    authenticate(req, httpRequest);
} catch (Exception e) {
    logger.error("Authentication failed. Message: " + e.getMessage(), new Date().toString(), service: "PaymentServiceProvider", req);
    return ResponseEntity.status(500).body(Map.of("kl": "Error", "vk": "Failed to make the request"));
}
```

```

private void authenticate(HttpServletRequest httpRequest) throws IOException {
    Cookie[] cookies = httpRequest.getCookies();
    Cookie accessCookie = Arrays.stream(cookies).stream<Cookie>
        .filter(cookie -> cookie.getName().equals("access_token"))
        .findFirst().optional<Cookie>()
        .orElseThrow(() -> new RuntimeException("No access token found"));
    String token = accessCookie.getValue();
    OkHttpClient okHttpClient = new OkHttpClient.Builder()
        .addInterceptor(chain -> {
            Request originalRequest = chain.request();
            Request.Builder builder = originalRequest.newBuilder()
                .header("Authorization", "Bearer " + token);
            Request newRequest = builder.build();
            return chain.proceed(newRequest);
        })
        .build();

    Retrofit retrofit = new Retrofit.Builder()
        .baseUrl("http://localhost:8001/auth/verify/")
        .addConverterFactory(GsonConverterFactory.create())
        .client(okHttpClient)
        .build();
    AuthCall auth = retrofit.create(AuthCall.class);
    Call<Map<String, Object>> call = auth.authenticate(url: "http://localhost:8001/auth/verify/");
    Response<Map<String, Object>> response = call.execute();
    if (!response.isSuccessful()) {
        throw new RuntimeException("Authentication failed");
    }
}
}

```

2. Definisanje rola i odgovornosti

U sistemu postoje dve uloge: User i Admin. Uloga se dodaje svakom korisniku.

Primer klase Role:

```

@Table(name = "ROLE")
public class Role implements GrantedAuthority {
    private static final long serialVersionUID = 1L;

    @Id
    @Column(name = "id")
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    Long id;

    @Column(name = "name")
    String name;

    @ManyToMany(fetch = FetchType.EAGER)
    @JoinTable(name = "role_privilege",
        joinColumns = @JoinColumn(name = "role_id", referencedColumnName = "id"),
        inverseJoinColumns = @JoinColumn(name = "privilege_id", referencedColumnName = "id"))
    private Set<Privilege> privileges;

    @JsonIgnore
    @ManyToMany(mappedBy = "roles", fetch = FetchType.LAZY)
    private Set<User> users;
}

```

Svaka ulogu ima privilegiju, dodela privilegija rađena je po principu **least privilege**.

Admini mogu da kreiraju nove korisnike, ažuriraju njihove podatke i vrše plaćanje,, korisnici mogu da vrše plaćanje.

Primer zaštite ruta izgleda ovako:

```

Andela
@GetMapping()
@PreAuthorize("hasRole('ADMIN')")
public List<UserInfoResponse> getLawyers() { return userService.getLawyers(); }

Andela
@GetMapping("/{userEmail}")
@PreAuthorize("hasRole('ADMIN')")
public UserDetailsResponse getLawyer(@PathVariable String userEmail) { return userService.getLawyer(userEmail); }

Andela
@PutMapping()
@PreAuthorize("hasRole('ADMIN')")
public UserDetailsResponse editLawyer(@RequestBody UserDetailsRequest user) { return userService.editLawyer(user); }

Andela
@DeleteMapping("/{userEmail}")
@PreAuthorize("hasRole('ADMIN')")
public ResponseOk deleteUser(@PathVariable String userEmail) {
    userService.deleteUser(userEmail);
    return new ResponseOk("User logically deleted.");
}

```

3. Ne koristiti protokole za koje je ustanovljeno da nisu bezbedni – poput SSL v1, FTP

Ne koriste se prokoli poput SSL v1 i FTP, za Web komunikaciju koristi se HTTPS (SSL TLS) prokol i prilikom komunikacije komponentata unutar sistema, koristi se HTTP prokol.

4. Ne koristiti podrazumevane naloge ili lozinke

U sistemu se ne koriste podrazumevani nalozi niti lozinke. Lozinka korisnika mora sadržati velika i mala slova, specijalne karaktere i biti minimum 12 karaktera dugačka.

```
public Boolean checkPasswordStrength(String password) {
    if (password.length() < 12) {
        return false;
    }
    boolean hasUppercase = !password.equals(password.toLowerCase());
    boolean hasLowercase = !password.equals(password.toUpperCase());
    boolean hasNumber = password.matches(regex: ".*\\d.*");
    boolean hasSpecialChar = password.matches(regex: ".*[!@#$%^&*()_+].*");
    return hasUppercase && hasLowercase && hasNumber && hasSpecialChar;
}
```

II grupa

1. Zastititi sacuvane podatke naloga

Sve lozinke korisnika se heširaju. Heširanje lozinke postiže se korišćenjem *PasswordEncoder* interfejsa iz Spring Security biblioteke, specifično kroz **BCryptPasswordEncoder** koji se specificira u konfiguraciji.

Heširanje lozinke:

```
Lawyer.setPassword(passwordEncoder.encode(registrationRequest.getPassword()));
```

Pored lozinke, potrebno je i enkriptovati korisnikove podatke o kartici. Ono što se enkriptuje jeste PAN, SecurityCode i CardholderName.

Enkripcija se radi prilikom čuvanja podataka u bazu na sledeći način:

```

public static String encrypt(String value) {
    try {
        IvParameterSpec iv = new IvParameterSpec(initVector.getBytes(StandardCharsets.UTF_8));
        SecretKeySpec skeySpec = new SecretKeySpec(key.getBytes(StandardCharsets.UTF_8), algorithm: "AES");

        Cipher cipher = Cipher.getInstance(algo);
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec, iv);

        byte[] encrypted = cipher.doFinal(value.getBytes());
        return Base64.encodeBase64String(encrypted);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    return null;
}

```

Ovde se radi o simetričnoj enkripciji koristeći AES algoritam u CBC (Cipher Block Chaining) modu sa PKCS5 paddingom.

2. Zaštitite podatke vlasnika kartice snažnom kriptografijom prilikom prenosa preko otvorene javne mreže

III grupa

1. Periodično skeniranje ranjivosti

Skeniranje ranjivosti vrši se uz pomoć **SonarQube** alata. Potrebno je pozicionirati se u svaki od projekta i izvršiti komandu:

```

Unset
mvn clean verify sonar:sonar \
-Dsonar.projectKey=sep \
-Dsonar.projectName='sep' \
-Dsonar.host.url=http://localhost:9000 \
-Dsonar.token=sqp_77f770af99b1ad42b0564c047ae059bd6ddcf62f

```

U SonarQube-u mogu se videti potencijalni bagovi, kvalitet koda (pokrivenost testovima, dupliciran kod), sigurnosni propusti i još mnogo toga.

Na slici je prikazana analiza za **payment-service-provider** mikroservis:

My IssuesAll

Filters

Issues in new code

Clean Code Attribute

Consistency13

Intentionality7

Adaptability19

Responsibility0

Software Quality

Security0

Reliability8

Maintainability38

Severity

Type

Scope

OpenNot assignedMaintainabilityCode SmellMajor10min effort • 26 days ago

Adaptability issue

Replace this use of System.out by a logger.bad-practicecert

OpenNot assignedMaintainabilityCode SmellMajor10min effort • 1 month ago

Adaptability issue

Replace this use of System.out by a logger.bad-practicecert

OpenNot assignedMaintainabilityCode SmellMajor10min effort • 1 month ago

src/.../controller/PaymentRegistryController.java

Consistency issue

Remove this field injection and use constructor injection instead.No tags

OpenNot assignedMaintainabilityReliabilityCode SmellMajor5min effort • 2 months ago

src/.../paymentserviceprovider/logger/Log.java

Adaptability issue

Make logLevel a static final constant or non-public and provide accessors if needed.cwe

OpenNot assignedMaintainabilityCode SmellMinor10min effort • 2 days ago

Adaptability issue

Make message a static final constant or non-public and provide accessors if needed.cwe

OpenNot assignedMaintainabilityCode SmellMinor10min effort • 2 days ago

Adaptability issue

Provera bezbednosti:

0.0% Security Hotspots Reviewed

To reviewAcknowledgedFixedSafe

2 Security Hotspots

Review priority: Medium

Weak Cryptography1

Make sure that using this pseudorandom number generator is safe here.

Review priority: Low

Insecure Configuration1

Make sure this debug feature is deactivated before delivering the code in production.

2 of 2 shown

Make sure that using this pseudorandom number generator is safe here.

Using pseudorandom number generators (PRNGs) is security-sensitive java:S2245

Status: To review

This security hotspot needs to be reviewed to assess whether the code poses a risk.

Review

Where is the risk?What's the risk?Assess the riskHow can I fix it?Activity

src/.../paymentserviceprovider/service/PaymentService.java

Open in IDE

```
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Make sure that using this pseudorandom number generator is safe here.

Pregled citavog sistema:

	Lines of Code	Bugs	Vulnerabilities	Code Smells	Security Hotspots	Coverage	Duplications
sep	—	—	—	—	—	—	—
src	392	1	0	38	2	0.0%	0.0%
pom.xml	91	0	0	0	0	—	0.0%

2. Upravljanje zakrpama softvera

Uz pomoć **Maven dependency management**. Potrebno je pozicionirati se u odgovarajući mikroservis i pokrenuti dve komande:

```
Unset
mvn versions:display-dependency-updates
mvn versions:display-plugin-updates
```

Komande se koriste za proveru verzija zavisnosti i pluginova koje se nalaze u pom.xml paketu.

Primer pokrenute komanda za upravljanje zavisnostima u **bank** mikroservisu:

```
[INFO]
pavle@pop-os $ [~/Desktop/sep/backend/bank]: mvn versions:display-dependency-updates
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.example:bank >-----
[INFO] Building bank 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- versions-maven-plugin:2.15.0:display-dependency-updates (default-cli) @ bank ---
[INFO] The following dependencies in Dependency Management have newer versions:
[INFO] ch.qos.logback:logback-access ..... 1.4.11 -> 1.4.14
[INFO] ch.qos.logback:logback-classic ..... 1.4.11 -> 1.4.14
[INFO] ch.qos.logback:logback-core ..... 1.4.11 -> 1.4.14
[INFO] co.elastic.clients:elasticsearch-java ..... 8.7.1 -> 8.12.0
[INFO] com.couchbase.client:java-client ..... 3.4.11 -> 3.5.2
```

Primer pokrenute komanda za upravljanje pluginova u **bank** mikroservisu:

```
pavle@pop-os $ [~/Desktop/sep/backend/bank]: mvn versions:display-plugin-updates
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.example:bank >-----
[INFO] Building bank 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- versions-maven-plugin:2.15.0:display-plugin-updates (default-cli) @ bank ---
[INFO]
[INFO] All plugins with a version specified are using the latest versions.
[INFO]
[INFO] All plugins have a version specified.
```

IV grupa

1. Ograničiti pristup podacima vlasnika kartice

Pristup podacima se ograničava uz pomoć definisanih uloga sa **least privilege principom**. Takođe, prilikom svakog plaćanja vrši se autentifikacija korisnika uz pomoć JWT tokena.

2. Identifikacija i potvrda pristupa komponentama sistema

Svi entiteti u sistemu su jednoznačno određeni svojim identifikatorom. S obzirom na to da se koristi MongoDB baza podataka, svakom korisniku pri čuvanju u bazu, automatski se kreira objekat koji je tipa **ObjectId**. Autentifikacija korisnika se vrši na osnovu identifikatora.

V grupa

1. Pratiti i nadgledati sav pristup mrežnim resursima i podacima o vlasnicima kartica

Svaka akcija u sistemu se loguje. Logovi se čuvaju u MongoDB.

Za potrebe logovanja napravljena je klasa Logger koja poseduje tri funkcije kako bi se označile tri vrste logova:

```
public class Logger {  
  
    @Autowired  
    private LoggerRepo loggerRepo;  
  
    13 usages  ▲ Aleksandra Nedic  
    public void info(String message, String timestamp, String service, Object payload) {  
        Log log = new Log(LogLevel.INFO, message, timestamp, service, payload);  
        loggerRepo.save(log);  
    }  
  
    no usages  ▲ Aleksandra Nedic  
    public void warn(String message, String timestamp, String service, Object payload) {  
        Log log = new Log(LogLevel.WARN, message, timestamp, service, payload);  
        loggerRepo.save(log);  
    }  
  
    19 usages  ▲ Aleksandra Nedic  
    public void error(String message, String timestamp, String service, Object payload) {  
        Log log = new Log(LogLevel.ERROR, message, timestamp, service, payload);  
        loggerRepo.save(log);  
    }  
}
```

Kao što se može videti iz priloženog, logovi sadrže vreme kreiranja loga, servis u kom se log kreirao i dodatni payload kako bi se bliže opisala data akcija.

Primer u kom se čuva log jeste kada se kreira transakcija, izvrši plaćanje, ili kada se dogodi neka greška, kao što je na primer nedostatak sredstava na računu ili nemogućnost pronalaska korisnika.

Primer nekih od logova koji se nalaze u bazi:

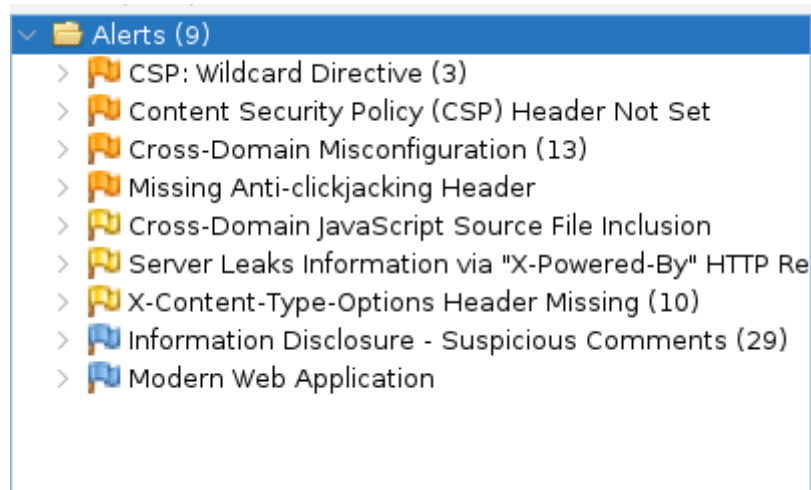
```
_id: ObjectId('65b5381fc28003210de9f259')
logLevel: "ERROR"
message: "Payment failed. Message: Not Found"
timestamp: "Sat Jan 27 18:06:39 CET 2024"
service: "PaymentServiceProvider"
payload: Object
_class: "com.example.paymentserviceprovider.logger.Log"
```

```
_id: ObjectId('65b6db281a5e2d5757908051')
logLevel: "ERROR"
message: "Authentication failed. Message: Use JsonReader.setLenient(true) to acc..."
timestamp: "Sun Jan 28 23:54:32 CET 2024"
service: "PaymentServiceProvider"
payload: Object
_class: "com.example.paymentserviceprovider.logger.Log"
```

```
_id: ObjectId('65b6dbaec42c6268b71525c9')
logLevel: "INFO"
message: "Payment successful"
timestamp: "Sun Jan 28 23:56:46 CET 2024"
service: "PaymentServiceProvider"
payload: Object
_class: "com.example.paymentserviceprovider.logger.Log"
```

2. Redovno testiranje bezbednosnih sistema i procesa

Skeniranje bezbednosti frontend aplikacije izvršeno je OWASP ZAP alatom. Pri skeniranju, dobijeni su sledeći rezultati:



Možemo primetiti da alat nije uspeo da pronađe rizik visokog prioriteta. Kao što je prikazano na slici, za prve dve stavke potrebno je postaviti odgovarajuća CSP zaglavlja. Cross-Domain configuration dešava se zato što je Access-Allow-Origin zaglavlje postavljeno na vrednost *, što svakako ne bi trebalo biti slučaj u produkcionom okruženju. Za Missing Anti-clickjacking

Header potrebno je dodati zaglavlje koje onemogućava da se sajt integriše u okviru iframe elementa. Ostala upozorenja su niskog prioriteta.

Za testiranje sigurnosti korišćen je još jedan alat, a to je DirBuster koji pronalazi rute koje mogu sadržati osetljive informacije. Pronađena je samo ruta /error.

