Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Klavdija Piskule Smolnikar

Dvodejavniška avtentikacija

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: dr. Andrej Brodnik
Somentor: prof. dr. Aleksandar Jurišić

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Pri uporabi storitev na internetu je tveganje, ki ga prinaša napačno preverjanje pristnosti, vedno večje. Tradicionalna uporaba uporabniškega imena in gesla ni več zadovoljiva in zahteva dopolnitev. Le-to prinaša tako imenovana dvodejavniška avtentikacija, pri kateri uporabniško ime z geslom nadgradimo z dodatnim dejavnikom avtentikacije. Slednje običajno predstavlja neko udejanjenje bodisi uporabe izziva bodisi enkratnega gesla. V diplomski nalogi preglejte področje vključno z načini avtentikacije in protokoli za avtentikacijo. Na podlagi ovrednotenja možnih rešitev namestite in preskusite praktično rešitev, ki bo uporabljala standardne in odprtokodne implementacije.

Zahvaljujem se mentorju dr. Andreju Brodniku in somentorju prof. dr. Aleksandru Jurišiću za vso pomoč in strokovne napotke pri pisanju diplomskega dela.

Sodelavcem oddelka Register.si se zahvaljujem za pomoč, nasvete, podporo in razumevanje.

Še posebej se zahvaljujem družini in prijateljem za podporo in vzpodbudne besede v času študija.

Kazalo

Seznam uporabljenih kratic

CHAP	(angl. Challenge-Handshake Authentication Protocol) *Avtentikacijski protokol, ki deluje na podlagi izziva*
FIPS	(angl. Federal Information Processing Standards) *Zvezni standardi za procesiranje informacij*
GNU GPL	(angl. GNU General Public License) *Odprtokodna licenca za programsko opremo*
HMAC	(angl. Hash based Message Authentication Code) *Zgoščevalna funkcija, ki uporablja skrivni ključ*
HOTP	(angl. HMAC-based One-time Password Algorithm) *Algoritem za generiranje enkratnega gesla na podlagi vrednosti HMAC*
HTTP	(angl. Hypertext Transfer Protocol) *Protokol za prenos večpredstavnih vsebin*
ID	(angl. Identity) *Identiteta*
IEC	(angl. International Electrotechnical Commission) *Mednarodna komisija za elektrotehniko*
IMEI	(angl. International Mobile Equipment Identity) *Mednarodna identiteta mobilne naprave*
IP	(angl. Internet Protocol) *Internetni protokol*
ISO	(angl. International Organization for Standardization) *Mednarodna organizacija za standardizacijo*
LCD	(angl. Liquid Crystal Display) *Zaslon iz tekočih kristalov*
MD5	(angl. Message Digest Algorithm) *Zgoščevalna funkcija MD5*

MS-CHAP	(angl. Microsoft Challenge-Handshake Authentication Protocol) *Avtentikacijski protokol podjetja Microsoft, ki deluje na podlagi izziva*
NAS	(angl. Network Access Server) *Omrežni dostopovni strežnik*
NFC	(angl. Near Field communication) *Komunikacija s sosednjim poljem*
NIST	(angl. National Institute of Standards and Technology) *Nacionalni inštitut za standarde in tehnologijo*
NSA	(angl. National Security Agency) *Nacionalna varnostna agencija*
OATH	(angl. Open Authentication) *Odprta avtentikacija*
OTA	(angl. Online Trust Alliance) *Iniciativa za varnost na internetu*
OTP	(angl. One Time Password) *Enkratno geslo*
PAM	(angl. Pluggable Authentication Module) *Modul za preverjanje pristnosti*
PAP	(angl. Password Authentication Protocol) *Avtentikacijski protokol, ki za avtentikacijo uporablja geslo*
PIN	(angl. Personal Identification Number) *Osebna identifikacijska številka*
PKI	(angl. Public Key Infrastructure) *Infrastruktura javnih ključev*
SHA-1	(angl. Secure Hash Algorithm) *Zgoščevalna funkcija SHA-1, ki je zasnovana na idejah MD družine*
SIM	(angl. Subscriber Identity Module) *Naročniški identifikacijski modul*
SMS	(angl. Short Message Service) *Sistem kratkih sporočil*
SSL	(angl. Secure Socket Layer) *Kriptografski protokol, ki omogoča varno komunikacijo na medmrežju*
TCP	(angl. Transmission Control Protocol) *Protokol za krmiljenje prenosa*
TLS	(angl. Transport Layer Security) *Kriptografski protokol, ki omogoča varno komunikacijo na medmrežju*
TOTP	(angl. Time One Time Password) *Enkratno geslo na podlagi časa*

UDP	(angl. User Datagram Protocol) *Nepovezovalni protokol za prenos paketov*
URI	(angl. Uniform Resource Identifier) *Enolični identifikator vira*
U2F	(angl. Universal 2nd Factor) *Univerzalni drugi dejavnik*
VPN	(angl. Virtual Private Network) *Navidezno zasebno omrežje*

Povzetek

Naslov: Dvodejavniška avtentikacija

Avtor: Klavdija Piskule Smolnikar

V diplomski nalogi je opisana priprava sistema za dvodejavniško avtentikacijo na Register.si, s čimer bomo zagotovili višjo stopnjo varovanja informacij. Sistem je sestavljen iz strežnika FreeRADIUS, ki skrbi za avtentikacijo uporabnikov v povezavi z orodjem MultiOTP, ter spletne storitve AuthenticationApi, ki sprejema zahteve za avtentikacijo uporabnika.

Predstavili bomo pomen dvodejavniške avtentikacije ter opisali različne avtentikacijske algoritme in protokole ter varnostne naprave. Znotraj dvodejavniške avtentikacije prvi dejavnik predstavljata uporabniško ime in geslo, kot drugi dejavnik pa smo izbrali generator enkratnih gesel, ki je glede na vnaprej določene kriterije najprimernejši.

Ključne besede: Dvodejavniška avtentikacija, varnostne naprave, TOTP, FreeRADIUS.

Abstract

Title: Two-factor authentication

Author: Klavdija Piskule Smolnikar

The purpose of this thesis is to describe the setup of a two-factor authentication system at Register.si, which will ensure a greater protection of information. The system consists of a FreeRADIUS server that takes care of user authentication in connection to the MultiOTP tool, as well as of AuthenticationApi web services that accept user authentication requirements.

The thesis will present an importance of two-factor authentication, as well as offer descriptions of various authentication algorithms and protocols, and security devices. Concerning two-factor authentication, the username and the password cover the first factor, while a one-time password token generator was chosen as the second factor, as the most appropriate one for a predetermined criteria.

Keywords: Two factor authentication, security tokens, TOTP, FreeRADIUS.

Poglavje 1

Uvod

Register.si je del Akademske in raziskovalne mreže Slovenije (ARNES) ter opravlja funkcijo nacionalnega registra za vrhnjo internetno domeno `.si`. Registracija domen poteka preko registrarjev, ki imajo z Arnesom sklenjeno pogodbo o sodelovanju pri registraciji domen. Ob registraciji domene registrar registru sporoči kontaktne podatke nosilca domene ter podatke o imenskih strežnikih. Ta nato podatke vpiše v področje `.si`.

Register.si registrarjem nudi dostop do strežnika za registracijo domen, strežnika `whois` in do portala za registrarje. Slednji jim ponuja možnost pregleda nad registriranimi domenami ter z njimi povezanimi osebami in organizacijami. Trenutno je registrarjem dostop do portala omogočen na podlagi uporabniškega imena in gesla, poleg tega mora biti IP naslovov, s katerega želijo dostopati do sistema, na *seznamu dovoljenih IP naslovov* (angl. IP Whitelisting).

Težava, ki se pojavi pri uporabi seznama dovoljenih IP naslovov, je, da so registrarji omejeni glede tega, od kod lahko dostopajo do sistema. Če jim želimo omogočiti dostop od koderkoli, preprosta uporaba uporabniškega imena in gesla ni dovolj varna, da bi z njo lahko preprečili nepooblaščene dostope, glej Rouse [29]. Zato smo se znotraj Register.si odločili, da vzpostavimo sistem, ki bo omogočal prijavo na podlagi dvodejavniške avtentikacije, rešitev pa naj bi bilo kasneje mogoče razširiti tudi na druge sisteme, ki jih

uporabljamo.

Glavna želja je, da bi registrarji do portala za registrarje lahko dostopali tudi preko mobilne naprave. Zato bo potrebno omejitev dostopov na podlagi IP naslova nadomestiti z drugim dejavnikom, ki ne bo vezan le na določeno napravo. Najprej smo si pogledali, kako imajo urejen dostop do sistemov ostali nacionalni evropski registri za vrhnje internetne domene. Glede na rezultate ankete o dvodejavniški avtentikaciji, opravljene s strani združenja CENTR [2], je razvidno, da kar 20 od 26 registrov uporablja dvodejavniško avtentikacijo. Od tega le 4 registri kot drugi dejavnik uporabljajo fizične varnostne naprave, in sicer generatorje enkratnih gesel na osnovi časa TOTP [25]. Kar 14 se jih posložuje programskih rešitev, od tega so vsi navedli, da uporabljajo Google avtentikator [16].

V nadaljevanju bomo najprej opisali osnove kriptografije, kjer bomo govorili predvsem o zgoščevalnih funkcijah in kriptosistemu z javnim ključem. V tretjem poglavju bomo opisali različne avtentikacijske dejavnike. Osredotočili se bomo na varnostne naprave in primerjavo le-teh na podlagi kriterijev, kot so cena, varnost, skladnost z različnimi operacijskimi sistemi in preprostost uporabe. Opisali bomo tudi algoritme, na podlagi katerih te naprave delujejo, ter opisali njihove prednosti in slabosti. Glede na ugotovitve se bomo nato odločili za uporabo naprave, ki bo najbolj primerna.

V četrtem poglavju bomo predstavili avtentikacijsko orodje MultiOTP in strežnik FreeRADIUS, ki bo opravljal glavno nalogo v našem sistemu za avtentikacijo. Postavitev in arhitekturo celotnega sistema bomo predstavili v petem poglavju. V tem poglavju bomo tudi ovrednotili celotno rešitev glede na funkcionalne in nefunkcionanlne zahteve. Te so dostop do portala za registrarje od koderkoli, varnost in cena. V šestem poglavju bomo podali sklepne misli in ugotovitve.

Poglavje 2

Osnove kriptografije

Kriptografija se ukvarja s tem, kako zagotoviti varno izmenjavo sporočil med dvema osebama (poimenujmo ju Anita in Borut), ne da bi jih bil opazovalec (Oskar) sposoben razvozljati in spremeniti njihovo vsebino. Če želimo nekomu omejiti dostop do informacij, je ena izmed možnosti njihovo šifriranje. Pri tem postopku uporabljamo šifre in jih delimo na zasebne oz. simetrične šifre in šifre z javnimi ključi oz. asimetrične šifre.

Ko govorimo o simetričnih šifrah, gre za to, da si Anita in Borut delita zasebni ključ k, s katerim šifrirata oz. odšifrirata sporočila. Pomankljivost takšnega pristopa je v tem, da se morata Anita in Borut pred pričetkom izmenjave sporočil sestati in izbrati k. Poleg tega se s povečanjem števila oseb (n), ki si želijo varno izmenjevati sporočila v omrežju, povečuje tudi število ključev za vsakega posameznika ($n-1$). Te pomanjkljivosti rešujejo šifre z javnimi ključi, ki ji bomo bolj podrobno opisali v naslednjem podpoglavju.

Primer simetrične šifre je DES (Data Encription Standard), ki uporablja ključe dolžine 56 bitov. Zaradi kratke dolžine ključev ga je NIST leta 2001 nadomestil z AES (Advance Encription Standard), ki za šifriranje podatkov lahko uporablja ključe dolžin 128, 192 ali 256 bitov in tako zagotavlja večjo varnost, glej Pfleeger in Margulies [15, Sect. 2.3].

2.1 Kriptosistem z javnim ključem

Leta 1976 sta Whifield Diffie in Martin Hellman iznašla nov način šifriranja sporočil imenovan *kriptografija z javnim ključem*, glej Pfleeger et al. [15, Sect. 2.3]. Gre za to, da ima vsak uporabnik svoj par ključev. Zasebnega, tega obdrži zase, in javnega, ki ga objavi. Čeprav se ključe generira kot matematično povezane pare, iz javnega ključa ne moremo v doglednem času, v katerem želimo zadržati napadalca, izračunati zasebnega ključa.

V praksi komunikacija med Anito in Borutom poteka tako, da Anita najprej objavi svoj javni ključ. S tem ključem Borut zašifrira sporočilo namenjeno Aniti. Na ta način dosežemo zasebnost, saj lahko to pismo odšifrira le Anita.

Če Borut želi Aniti zagotoviti, da je sporočilo res poslal on, ga mora najprej zašifrirati s svojim zasebnim ključem (tj. digitalno podpiše). Podpisano sporočilo nato zašifrira še z Anitinim javnim ključem in ji le-to pošlje. Anita prejeto sporočilo najprej odšifrira s svojim zasebnim ključem, zatem pa preveri, ali je sporočilo res poslal Borut. To naredi tako, da sporočilo odšifrira z Borutovim javnim ključem in na tak način preveri njegov digitalni podpis, glej Jurišić [21].

2.2 Digitalni podpis

Najmočnejše orodje za izkazovanje pristnosti je digitalni podpis. Njegova vloga je podobna podpisu na papirju, le da ga uporabljamo za podpisovanje dokumentov v digitalni obliki. Z njim prejemnika prepričamo, da je podpisnik dokumenta res tisti, za katerega se izdaja, in tako dosežemo avtentičnost.

Ker so dokumenti poljubno veliki in je šifriranje datotek z asimetričnimi algoritmi počasno, si v praksi pomagamo z zgoščevalnimi funkcijami, ki jih bomo opisali v naslednjem podpoglavju. Z uporabo zgoščevalnih funkcij iz dokumenta najprej izračunamo njegovo zgostitev in jo nato podpišemo (tj. jo v primeru RSA zašifriramo z zasebnim ključem).

Pojavi se problem, kako zagotoviti zaupanje med dvema osebama, ki se

nikoli nista srečali. Če Anita podpiše sporočilo s svojim zasebnim ključem, lahko kdorkoli odšifiria sporočilo z njenim javnim ključem in s tem preveri, da ga je res podpisala ona. Vendar kako vemo, ali ključ res pripada Aniti in ne nekomu, ki se za njo izdaja? Ta problem rešuje infrastruktura javnih ključev (anlg. Public Key Infrastructure - PKI), ki združuje nabor tehnologij, postopkov, naprav in predpisov, potrebnih za varno upravljanje zasebnih in javnih ključev.

PKI zagotavlja tudi varno ustvarjanje, shranjevanje in distribucijo digitalnih potrdil. Slednje vsebuje osebne podatke entitete, ki ji je bilo potrdilo izdano s strani zaupanja vrednega organa za overjanje (angl. Certification Authority) in povezuje javni ključ z uporabnikom. Z uporabo digitalnih potrdil uporabniki potrjujejo svojo pristnost [28].

Infrastrukturo PKI lahko v obliki fizičnih naprav implementirajo pametne kartice. Gre za majhne elektronske naprave v velikosti kreditne kartice z vgrajenim mikroprocesorjem ali pomnilniškim vezjem, ki za svoje dolovanje potrebujejo čitalec.

2.3 Zgoščevalne funkcije

Eden izmed temeljnih gradnikov moderne kriptografije so kriptografske zgoščevalne funkcije. To so funkcije, ki poljubno dolgo sporočilo pretvorijo v zaporedje bitov določene dolžine, imenovano *zgostitev* (ali izvleček, t.i. digitalni prstni odtis), glej Stinson [31, Sect. 5.1]. V kombinaciji z asimetričnimi algoritmi se najpogosteje uporabljajo za šifriranje in digitalno podpisovanje.

Moderne zgoščevalne funkcije morajo biti deterministične, računsko učinkovite in morajo izpolnjevati naslednja kriterija:

- so enosmerne, kar pomeni, da iz vhoda izračunajo zgostitev, iz zgostitve pa ni mogoče učinkovito izračunati vhoda v funkcijo,
- trčenja so težko izračunljiva, tj. nemogoče je učinkovito izračunati različni zaporedji nizov, katerih zgostitvi zgoščevalne funkcije sta enaki.

Najpogosteje uporabljene zgoščevalne funkcije pripadajo družini SHA (Secure Hash Algorithm), ki jih bomo za potrebe tega dela tudi podrobneje opisali. Te funkcije imajo dodatno še to praktično lastnost, da majhna sprememba v sporočilu povzroči dobro zaznavno spremembo v zgostitvi (tj. stara in nova zgostitev ne izgledata podobno).

Leta 1993 je Nacionalni inštitut za standarde in tehnologijo (NIST) objavil zgoščevalno funkcijo SHA-0 zasnovano na algoritmih MD4 in MD5. Kmalu jo je zamenjala njena izboljšana različica SHA-1, ki jo je predlagala Nacionalna varnostna agencija (NSA) (dodali so zamik za 1 na vsakem krogu). Dolžina vhodnega sporočila pri SHA-1 je omejena na 2^{64} bitov in kot rezultat vrne zgostitev dolžine 160 bitov. Leta 2005 je bil na zgoščevalno funkcijo SHA-1 odkrit napad, s katerim je možno v manj kot 2^{63} korakih poiskati različni vhodni sporočili z enako vrednostjo zgostitve. Zato je NIST leta 2008 objavil nov standard za zgoščevalne funkcije FIPS 180-3 [26]. Ta definira algoritme zasnovane na SHA algoritmu, ki vračajo zgostitve velikosti od 224 do 512 bitov in jih imenujemo algoritmi SHA-2, glej Pfleeger et al. [15, Sect. 12.4].

Naj omenimo še mehanizem za avtentikacijo sporočil (HMAC) [20], ki igra pomembno vlogo pri algoritmih za generiranje enkratnih gesel opisanih v naslednjem poglavju. Uporablja se v kombinaciji s katerokoli iterativno zgoščevalno funkcijo (npr. SHA-1) in skupnim skrivnim ključem ter služi kot mehanizem za preverjanje izmenjanih informacij med dvema uporabnikoma.

Poglavje 3

Avtentikacija

Ko želimo dostopati do nekega omrežja in do storitev, ki jih to ponuja, je ponavadi prvi korak postopek avtentikacije. To je proces preverjanja pristnosti, tj. trditve o identiteti, ki temelji na nizu poverilnic. Najpogostejši način podajanja poverilnic je v obliki uporabniškega imena in pripadajočega gesla. V primeru, da se podane poverilnice ujemajo s poverilnicami shranjenimi znotraj sistema za avtentikacijo, se običajno sproži še postopek avtorizacije, znotraj katerega poteka preverjanje, do katerih storitev oz. virov je uporabniku dovoljen dostop, glej Van der Walt [19].

Uporabniška imena so pogosto sestavljena le iz posameznikovih začetnic lastnega imena, izbrana gesla pa znajo biti šibka, zaradi česar jih je pogosto preprosto uganiti. Zato avtentikacija, ki temelji zgolj na podlagi vnosa uporabniškega imena in gesla, ne zagotavlja ustrezne varnosti pri dostopu do sistemov z občutljivimi podatki. Temu se do neke mere lahko izognemo z uporabo močnejših gesel z določitvijo določenih pravil, kot so minimalna dolžina znakov in kompleksnost z uporabo različnih simbolobv ter velikih in malih črk. Velikokrat pride do tega, da uporabnik geslo nehote razkrije ali pa ga celo pozabi.

Zaradi zgoraj omenjenih ranljivosti, ki jih prinaša avtentikacija samo na podlagi uporabniškega imena in gesla, je vse bolj pogosta uporaba kombinacije različnih neodvisnih avtentikacijskih dejavnikov. V grobem se ti de-

javniki delijo v tri kategorije, in sicer na nekaj, kar veš, na nekaj, kar imaš in na nekaj, kar si. Pri avtentikaciji z uporabniškim imenom in geslom tako govorimo o dejavniku nekaj, kar veš, saj si mora uporabnik uporabniško ime in geslo zapomniti. V primeru, ko govorimo o dejavniku nekaj, kar imaš, govorimo o tem, kar ima uporabnik v lasti. Največkrat so to razne fizične naprave, kot so npr. fizični generatorji enkratnih gesel in pametne kartice. Tretja kategorija, nekaj, kar si, pa se nanaša na biometrične lastnosti, kamor spadajo npr. prstni odtisi, očesna šarenica in mrežnica ter vzorci glasu in obraza, glej Rouse [29].

Zadnje čase velikokrat govorimo o pojmu močna avtentikacija, ki se največkrat nanaša na dvo- in večdejavniško avtentikacijo. Uporabnik svojo identiteto izkazuje z vsaj dvema neodvisnima avtentikacijskima dejavnikoma, ki pripadata različnima kategorijama. V nadaljevanju se bomo osredotočili na dvodejavniško avtentikacijo na podlagi uporabniškega imena in gesla ter varnostnih naprav, ki zagotavlja višjo raven varnosti storitev. Takšen način avtentikacije napadalcem močno otežuje pridobitev dostopa do uporabnikovih naprav ali spletnih računov, saj samo poznavanje uporabniškega gesla ni več dovolj za uspešno preverjanje identite, glej Rouse [29].

V sklopu dvodejavniške avtentikacije ponavadi govorimo o varnostnih napravah, ki so v lasti uporabnika in se z njimi lahko prijavi v določen sistem, in o programski opremi zadolženi za prepoznavo. Poleg varnostnih naprav v fizični obliki obstajo tudi njihove programske rešitve, ki so preprostejše za uporabo, oboje pa generirajo gesla po nekih vnaprej določenih algoritmih, ki jih bomo opisali v nadaljevanju.

3.1 Avtentikacijski algoritmi

V tem razdelku bomo opisali algoritme za ustvarjanje enkratnih gesel z zgoščevalnimi funkcijami.

3.1.1 Algoritem HOTP

Predstavimo preprost algoritem za generiranje enkratnega gesla, ki deluje na osnovi mehanizma za avtentikacijo sporočil (HMAC). Implementira ga lahko vsak razvijalec strojne ali programske opreme ter s tem ustvari interoperabilno avtentikacijsko napravo in programsko opremo. Postopek mora biti v skladu z naslednjimi zahtevami:

1. Algoritem uporablja števec ali zaporedje, saj je eden izmed ciljev vključitev algoritma za generiranje enkratnega gesla na podlagi vrednosti HMAC (HOTP) v naprave namenjene shranjevanju večje količine podatkov, kot so ključi USB in kartice SIM.

2. Algoritem je učinkovit z vidika implementacije v strojni opremi z minimizacijo zahtev, ki so povezane z baterijo, s številom gumbov, z računsko močjo in velikostjo zaslona v primeru, da ga naprava uporablja.

3. Algoritem deluje z napravami, ki ne podpirajo nobenega numeričnega vnosa, se pa lahko uporablja tudi z bolj naprednimi napravami, ki zahtevajo vnos PIN.

4. Vrednost, ki se izpiše na napravi, je preprosta za branje ter vnos in tako razumne dolžine, vendar ne manj kot 6 znakov. Prav tako je zaželeno, da je vrednost HOTP izključno numerična, kar omogoča preprost vnos v naprave, kot so telefoni.

5. Obstaja uporabniku prijazen mehanizem za ponovno usklajevanje števca, saj se le-ta na napravi poveča vsakič, ko to uporabnik od nje zahteva, na strežniku pa se števec poveča le ob uspešni avtentikaciji na podlagi veljavne vrednosti HOTP in je tako v tem primeru potrebno števca na eni in drugi strani ponovno uskladiti.

6. Algoritem uporablja močno skupno skrivnost, predstavljeno z nizom znakov, ki je znana le avtentikacijskemu strežniku in odjemalcu in mora

biti dolžine vsaj 128 bitov. Za zagotavljanje večje varnosti pred *napadom z uporabo grobe sile* (angl. brute force attack) je priporočiljiva uporaba skupne skrivnosti dolžine 160 bitov.

Kot že omenjeno, algoritem HOTP generira enkratna gesla na podlagi mehanizma za avtentikacijo sporočil HMAC. Za generiarnje enkratnih gesel se uporablja naraščajoči števec in statični simetrični ključ, ki je znan le varnostni napravi in strežniku za preverjanje. Za vrednost HOTP uporabimo algoritem HMAC-SHA-1 [22], ki sprejme poljubno velik niz podatkov in vrne vrednost določene dolžine 160 bitov:

$$\text{HOTP}(K, C) = \text{HMAC-SHA-1}(K, C), \tag{3.1}$$

kjer je K skupna skrivnost med odjemalcem in strežnikom, C pa 8-bitni števec, ki mora biti usklajen med generatorjem HOTP (odjemalcem) ter se poveča po vsakem generiranju vrednosti ter overiteljem HOTP (strežnikom). Dobljeno vrednost je nato potrebno okrajšati, da dobimo uporabniku prijazno vrednost za vnos, glej M'Raihi, Bellare, Hoornaert, Naccache in Rannen [24].

Algoritem HOTP je zaradi uporabe okrajšane vrednosti HMAC-SHA-1 ranljiv za napad z uporabo grobe sile, zaradi česar mora biti avtentikacijski strežnik sposoben zaznati in preprečiti tak napad. Prva možnost je uporaba omejitve števila zahtev z nastavitvijo parametra dušenja D, ki definira največje število možnih poskusov potrjevanja enkratnega gesla. Druga možnost je implementacija zakasnitvene sheme, ki deluje tako, da po i-tem neuspelem poskusu avtentikacijski strežnik počaka naraščajoče število sekund $D \cdot i$. V primeru, da je torej $D = 5$, bi po prvem poskusu strežnik moral počakati npr. 5 sekund, po drugem neuspelem poskusu pa $5 \cdot 2 = 10$ sekund itd.

Avtentikacija med odjemalcem in overiteljem mora potekati preko varnih kanalov ter z uporabo ustreznih varnostnih mehanizmov, kot je npr. uporaba identifikatorja seje, z namemon zaščite uporabnikove zasebnosti in izogibanja *napada s ponavljanjem* [24].

Glavni problem algoritma HOTP je, da je ustvarjeno geslo veljavno dolgo časa oz. vsaj do naslednjega poskusa avtentikacije, kar pomeni, da če napadalec geslo presteže, ga lahko uporabi kadarkoli. To pomanjklivjost rešuje algoritem TOTP, znotraj katerega imajo gesla omejeno veljavnost.

3.1.2 Algoritem TOTP

Pri Algoritmu TOTP gre za različico algoritma HOTP, ki deluje na podlagi časa in se tako izračuna po skupni funkciji (3.1), le da v tem primeru števec C nadomestimo z vrednostjo T, ki izhaja iz časovne reference. Za izračun enkratnega gesla se lahko namesto algoritma HMAC-SHA-1 uporablja algoritma HMAC-SHA-256 ali HMAC-SHA-512. Algoritem TOTP mora biti v skladu z naslednjimi zahtevami:

1. Za generiranje vrednosti TOTP morata odjemalec (fizični ali programski generator enkratnih gesel) in overitelj (avtentikacijski ali validacijski strežnik) poznati, ali sta sposobna pridobiti trenutni unix čas (tj. število pretečenih sekund od pričetka leta 1970 naprej).

2. Odjemalec in overitelj si ali delita skupno skrivnost ali pa posedujeta informacijo, na podlagi katere generirata skupno skrivnost.

3. Vrednosti tega algoritma so ustvarjene z uporabo funkcije (3.1).

4. Odjemalec in overitelj uporabljata enako vrednost časovnega koraka X, ki ponazarja trajanje veljavnosti gesla v sekundah.

5. Za vsakega odjemalca obstaja enolično določen skrivni ključ.

6. Ključi so naključno generirani.

7. Ključi so lahko shranjeni na napravah z *neprepustnimi lastnostmi* (angl. tamper resistance), ki so zaščitene pred nepooblaščenim dostopom in uporabo.

V osnovi je TOTP definiran kot

$$\text{TOTP} = \text{HOTP}(K, T), \tag{3.2}$$

kjer T predstavlja število časovnih korakov od začetnega časovnega števca T_0 in trenutnega unix časa. Bolj specifično je T določen kot

$$T = (\text{Trenutni unix čas} - T_0)/X, \tag{3.3}$$

kjer X predstavlja število časovnih korakov v sekundah (privzeta vrednost za X je 30 sekund) in je sistemski parameter, T_0 pa je unix čas, pri katerem začnemo šteti časovne korake (privzeta vrednost je 0) [25].

Ko avtentikacijski strežnik prejme enkratno geslo (OTP), ne ve, ob katerem času točno je bilo to ustvarjeno. Zaradi omrežnih zakasnitev je lahko presledek (št. časovnih korakov od časa T_0 naprej) med časom, ko je bil OTP ustvarjen, in časom, ko sistem za preverjanje prejme OTP, prevelik. Zato mora sistem za preverjanje tipično nastaviti politiko za sprejemljiv zakasnitveni interval prenosa OTP. Večji kot je ta interval, bolj je izpostavljen možnostim napada. Zato je priporočljivo, da je časovni zamik manjši od velikosti enega časovnega koraka. Kriptografi M'Raihi, Bellare, Hoornaert, Naccache in Rannen [24, 25] so z analizo pokazali, da je najpogosteje uporabljen način napada na HOTP funkcijo napad z uporabo grobe sile.

3.2 Avtentikacijski protokoli

V tem podpoglavju bomo opisali tri najpogosteje uporabljene avtentikacijske *protokole od točke do točke* (angl. Point to point protocol) (PPP), ki delujejo na podlagi uporabniškega imena in gesla.

3.2.1 PAP

Protokol PAP skrbi za avtentikacijo z geslom. Ne uporablja šifriranja podatkov, uporabniško ime in geslo pa se strežniku pošljeta v čistopisu. Zato velja

za enega izmed najmanj varnih protokolov. Je ranljiv za *napad s ponavljanjem* (angl. replay attack), kjer gre za to, da napadalec prestreže uporabniško ime in geslo ter ju uporabi za prijavo v sistem. Izpostavljenost uporabniškega imena in gesla preko medsebojne povezave je mogoče omejiti s tem, da se izognemo pošiljanju gesla v obliki golega besedila preko celotnega omrežja. Če je PAP uporabljen znotraj varnega kanala, je varen toliko kot kanal sam, glej Lloyd in Simpson. [23].

3.2.2 CHAP

Protokol CHAP je bil zasnovan kot izboljšava protokola PAP v smislu, da ne pošilja gesla v čistopisu. Deluje na podlagi skupne skrivnosti, ki je znana le overitelju in uporabniku [30].

Preverjanje pristnosti uporabnika poteka tako, da overitelj pošlje uporabniku naključno ustvarjen izziv. Uporabnik na izziv odgovori z izračunom kriptografske zgoščevalne funkcije, ki kot vhod prejme vrednost identifikatorja izziva združeno s skupno skrivnostjo in vrednostjo izziva. Overitelj odgovor preveri z lastnim izračunom pričakovane vrednosti zgoščevalne funkcije. Če se vrednosti ujemata, je preverjanje pristnosti potrjeno.

Vsako vrednost izziva lahko uporabimo le enkrat, saj bi v nasprotnem primeru ponovitev izziva v kombinaciji z enako skrivnostjo napadalcu dovolila odgovor s prej zajetim odgovorom. Prav tako mora biti vrednost izziva nepredvidljiva, da napadalec uporabnika ne more zavesti v odgovor na predviden prihodnji izziv. Ker se vsakič ustvari nov izziv, je možnost za uspešen napad s ponavljanjem zelo majhna.

3.2.3 Microsoft PPP CHAP

MS-CHAP je Microsoftova različica protokola CHAP. V principu deluje na precej podoben način kot CHAP. Glavna razlika med njima je v tem, da paket z odgovorom vsebuje polja, ki so posebej določena za MS-CHAP. Takšno polje je npr. *NT Response*, ki vsebuje uporabniško ime in geslo v posebnem

šifriranem formatu.

Protokol MS-CHAP za razliko od protokola CHAP zagotavlja še nadzorovane mehanizme za spremembo gesla s strani overitelja. Definira tudi množico kod z razlogi za neuspešno prijavo, ki se pošlje kot ločen paket [33].

MS-CHAP obstaja v različicah MS-CHAPv1 (z nekaj ranljivostmi povezanimi s pomankljivostmi lastne zgoščevalne funkcije) in MS-CHAPv2, kjer so bile odpravljene omenjene pomankljivosti. Novejša različica uporablja t.i. metodo *medsebojnega preverjanja pristnosti* (angl. mutual authentication), ki jo imenujemo tudi *dvosmerna avtentikacija* (anlg. two-way authentication). Odjemalec in strežnik morata drug drugemu dokazati svojo identiteto, preden se prične pošiljati promet preko povezave odjemalec - strežnik [32].

3.3 Varnostne naprave in njihove programske rešitve

Varnostne oz. avtentikacijske naprave, ki jih uporabljamo za pooblaščen dostop do omrežnih storitev, so lahko pametne kartice, naprave PKI ali generatorji enkratnih gesel, ki jih bomo opisali v nadaljevanju. Tako fizične naprave kot tudi njihove programske rešitve lahko uporabimo kot enega izmed dejavnikov v sklopu dvodejavniške avtentikacije in tako zagotovimo dodatno raven varnosti pri preverjanju pristnosti.

3.3.1 Programske rešitve ali fizične varnostne naprave, kaj izbrati?

Ko se odločamo za to, ali bomo dostop do storitev zaščitili z varnostnimi napravami ali njihovimi programskimi rešitvami, moramo imeti ves čas v mislih izbiro pravega ravnotežja med varnostjo in priročnostjo. Programske rešitve so bistveno lažje za uporabo, saj lahko digitalno potrdilo ali aplikacijo preprosto namestimo na napravo. Na ta način se izognemo stroškom nakupa dodatne naprave, vendar to za seboj potegne kar nekaj slabosti s stališča

varnosti.

Glavna razlika med strojnimi in programskimi rešitvami je v tem, kje je shranjena sama aplikacija oz. podatki. Pri programskih rešitvah sta aplikacija OTP ali digitalno potrdilo shranjena na napravi, ki ni bila posebej zasnovana za varovanje občutljivih podatkov. Primeri takšnih naprav so npr. mobilni telefon, tablični računalnik ali celo namizni računalnik, ki ga ponavadi želimo zaščititi. S shranjevanjem aplikacije in digitalnega potrdila na isti napravi dosežemo, da dejavnika nekaj, kar veš (geslo), in nekaj, kar imaš (varnostna naprava), nista več fizično ločena. Pri uporabi fizične varnostne naprave so vsi podatki shranjeni v sami napravi, ki je zasnovana tako, da so vse informacije znotraj nje na varnem.

V zadnjem času so telefoni in tablični računalniki postali naši osebni računalniki. Zaposleni velikokrat dostopajo do občutljivih podatkov podjetja, kot so npr. elektronska sporočila, kar z mobilne naprave in nanjo nalagajo zasebne dokumente. Ko so avtentikacijski podatki shranjeni na napravi, s katere želimo pridobiti dostop, le-ti niso več zavarovani z dvodejavniško avtentikacijo. To za uporabnika in za podjetje odpre celo vrsto ranljivosti, kot so kraja kartice SIM, izvoz semena OTP, ki služi kot osnova za generiranje vrednosti OTP, uporaba zlonamernih aplikacij in čas nerazpoložljivosti mobilne naprave.

Velik problem pri uporabi programskih rešitev predstavljajo zlonamerne aplikacije, ki jih uporabnik lahko naloži z interneta. Te aplikacije lahko pridobijo podatke iz telefona, vključno s podatki o kartici SIM, česar se uporabnik običajno ne zaveda. Kar 82% aplikacij, ki so na voljo, prebere ID naprave, medtem ko 26% mobilnih aplikcij pozna številko kartice SIM, ki je lahko uporabljena za pridobitev dostopa do številke za uporabo SMS OTP [17].

Težava, ki se pojavi pri uporabi telefona ali tabličnega računalnika, je čas nerazpoložljivosti, saj se baterija lahko nepričakovano izprazni. Tudi najzmogljivejše baterije niso sposobne zdržati več kot dan ali dva v primerjavi s fizičnimi napravami, pri katerih baterija lahko zdrži leta.

Poleg programskih rešitev v obliki generatorjev OTP obstajajo tudi programske rešitve naprav PKI, ki imajo poleg pomanjkljivosti aplikacij OTP še nekaj dodatnih pomankljivosti. Medtem ko je redkost, da je aplikacija OTP nameščena na namiznem računalniku, je to za digitalna potrdila pogosto. Mnogo uporabnikov namesti potrdila s svojimi javnimi in zasebnimi ključi neposredno na svoj namizni računalnik. To pomeni, da so ranljivi za krajo, izgubo in zlorabo podatkov z zlonamernimi aplikacijami kot tudi za vdor med procesom tvorjenja para ključev. Pri uporabi PKI mora biti zasebni ključ varno shranjen, saj potrjuje posameznikovo identiteto. Potrebujemo ga za odšifriranje in podpisovanje dokumentov. Računalnikov velikokrat ne uporabljamo daljše časovno obdobje in jih v tem času ne ugasnemo, ampak damo v spanje, kar pusti odprto okno za zlonamerne aktivnosti. Hkrati so računalniki ranljivi za *napade beležnikov tipkanja* (angl. keylogger attacks) in lahko pride do kraje gesla, ki se ga uporablja za dostop do PKI. Temu se lahko izognemo z uporabo fizičnih naprav PKI [17].

Kljub tveganju, ki ga predstavljajo programske rešitve, se mnogi zaradi priročnosti še vedno odločajo zanje. Ob tem je potrebno upoštevati specifične potrebe posameznika. Za nekatere so programske rešitve vse, kar potrebujejo, spet drugim se uporaba dodatnih fizičnih naprav zdi predraga. V resnici lahko pridobljena varnost z uporabo fizičnih naprav krepko presega finančno tveganje v primeru zlorabe podatkov. Po podatkih združenja OTA naj bi se kar 89% vseh varnostnih incidentov lahko izognili z uporabo primernih varnostnih ukrepov [17]. Ljudje so na splošno nagnjeni k izgubljanju mobilnih naprav in pozabljanju odjave iz računalnika, mnogo je takšnih, ki gesla še vedno zapisujejo na kose papirj, in jih nato pustijo na delovnem mestu. Z uporabo fizičnih varnostnih naprav, s katerimi zagotavljamo pravo dvodejavniško avtentikacijo, lahko bistveno zmanjšamo vpliv človeških napak.

3.3.2 Pregled generatorjev enkratnih gesel

Generatorje enkratnih gesel delimo na HOTP in TOTP generatorje, ki implementirajo ustrezne algoritme, opisane v prejšnjem poglavju. Delimo jih

(a) (b)

Slika 3.1: (a) OTP generator zaščiten s PIN [8] in (b) OTP generator brez PIN [11]

na fizične naprave in programske generatorje, kjer fizične naprave predstavljajo običajno manjše naprave, ki ob pritisku na gumb ali po vnosu kode PIN ustvarijo enkratno geslo in ga izpišejo na zaslon (slika 3.1). Uporabnik se nato z uporabo tega gesla prijavi. Programski generatorji delujejo po enakem principu, le da v tem primeru aplikacijo, kot je npr. Google Authenticator, ki oponaša vedenje fizičnega generatorja, namestimo na napravo, kot je npr. mobilni telefon.

Generatorji enkratnih gesel so danes eden izmed najpogosteje uporabljenih avtentikacijskih dejavnikov, saj je njihova uporaba preprosta, prav tako pa jih je dokaj preprosto implementirati v sisteme. Cena za fizične naprave se giblje nekje med 15 in 30 € in je bolj ali manj odvisna od količine kupljenih naprav. Ne glede na to, kako varna se na prvi pogled zdi uporaba takšnih naprav, ima takšen način dvodejavniške avtentikacije nekaj ranljivosti.

Ena izmed glavnih težav je možnost *napada s posrednikom* (angl. Man in the middle attack) v realnem času, kjer napadalec v realnem času zlorabi podatke uporabnika, ki se želi prijaviti v neko storitev. Napadalec posnema izgled ciljne spletne strani in tako pridobi vse potrebne avtentikacijske po-

datke uporabnika, ko se le-ta prijavi v storitev na ponarejeni spletni strani. Ti podatki se nato takoj prepošljejo na legitimno spletno stran in napadalcu omogočijo prijavo v imenu žrtve [14]. Pred takšnimi napadi se v veliki meri lahko ubranimo tako, da uporabnik za dostop do občutljivh podatkov nikoli ne uporablja javnega omrežja, še boljšo varnost pa omogoča uporaba VPN, saj komunikacija poteka preko varnega tunela. Prav tako je potrebno nadzorovati vse dostope do sistemov, ki vsebujejo zaupne podatke.

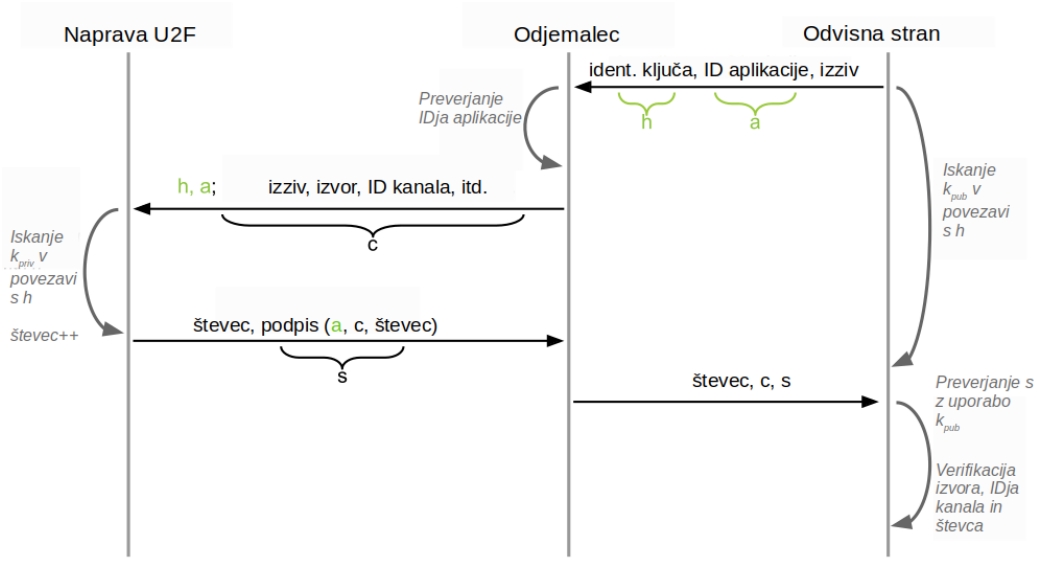
3.3.3 Naprave U2F

U2F je odprtokodni avtentikacijski standard, ki deluje po principu izziv - odgovor (primer je protokol CHAP opisan v podpoglavju 3.2.2). Internetnim uporabnikom omogoča varen dostop do katerekoli spletne strani, ki jo bomo imenovali odvisna stran, z uporabo samo ene naprave, pri tem pa ni potrebna dodatna uporaba gonilnikov ali programske opreme na strani odjemalca. Standard sta ustvarili podjetji Google in Yubico, danes pa zanj skrbi združenje FIDO [4].

Razširjen je z zaščito proti *napadom z ribarjenjem* (angl. phishing) (tj. napad z lažnim predstavljanjem, z namenom pridobivanja osebnih podatkov) in napadom s posrednikom. Avtentikacijski tok izziv-odgovor, v sklopu algoritma U2F, deluje skupaj z asimetričnim kriptosistemom, ki smo ga opisali v podpoglavju 2.1. Naprava U2F ima v lasti zasebni ključ k_z, *odvisni strani* (angl. relying party) pa je dodeljen ustrezen javni ključ k_p. Par ključev je ustvarjen v okolju, zavarovanem pred posegi, ki ga ključ k_z ne more zapustiti.

Za zaščito pred napadom z ribarjenjem in napadom s posrednikom je poskrbljeno tako, da odjemalec zbere vse podatke o tem, kar ve o trenutni povezavi HTTP, in sicer izvor (URI) in ID kanala TSL. Ta informacija je nato podpisana s strani naprave U2F in poslana odvisni strani, ki preveri, ali so podatki pravilni, kar je razvidno iz slike 3.2.

Da se odvisni strani onemogoči sledenje napravam med različnimi uporabniškimi računi, se uporabljajo posebni *aplikacijski ključi* (angl. application specific keys), kar pripomore k temu, da neka spletna stran ne more



Slika 3.2: U2F protokol [9].

vedeti, ali si dva različna uporabnika delita isto napravo. V ta namen naprava U2F ustvari nov par ključev in *identifikator ključa* (angl. key handle) ob vsaki registraciji. Identifikator ključa se shrani na odvisni strani, ki je po uspešnem preverjanju pristnosti poslan nazaj na napravo. Na ta način naprava ve, kateri ključ povezati z določenim uporabnikom [9].

Preprečevanje kloniranja naprav poteka z uporabo avtentikacijskega števca, ki se poveča ob vsaki avtentikaciji, odvisna stran pa preveri, ali je števec ob ponovni avtentikaciji višji od predhodnega ali ne.

Naprava U2F (slika 3.3) se vstavi v režo USB in deluje s katerokoli spletno stranjo, ki podpira U2F protokol. Z dotikom na gumb se sproži protokol, ki deluje takoj, uprabniku pa tako ni potrebno ponovno vnašati gesla iz naprave. Nekatere naprave U2F podpirajo tudi povezavo preko povezave bluetooth ali NFC, kar omogoča uporabo z mobilnimi napravami [9].

Uporaba naprav U2F trenutno predstavlja enega izmed najvarnejših dejavnikov avtentikacije, saj je protokol zasnovan tako, da ne dopušča možnosti za napade z ribarjenjem, napade s posrednikom, kloniranje naprav in ugrabitev seje. Ker pa je protokol še dokaj nov, je trenutno glavna težava v tem,

Slika 3.3: Naprava U2F podjetja Yubico [9].

da imata podporo zanj le brskalnika Google Chrome in Opera ter se tako naprav U2F v ostalih brskalnikih ne da uporabljati [9].

3.3.4 Naprave PKI

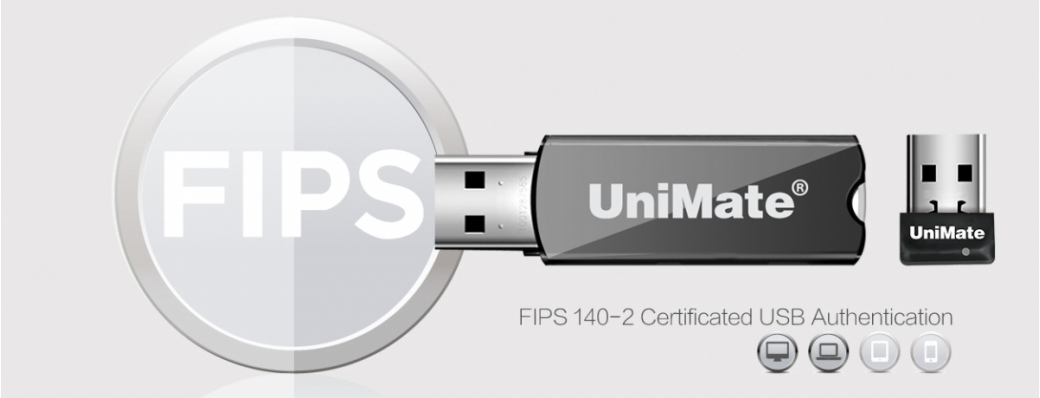
Za varno shranjevanje digitalnih potrdil in zasebnih ključev se uporabljajo naprave PKI. Omogočajo digitalno podpisovanje brez tveganja uhajanja informacij o zasebnem ključu. Šifriranje, odšifriranje in podpisovanje potekajo znotraj varnega vezja in tako nikoli ne more priti do kraje ključev [28].

Naprave PKI v obliki ključev USB (slika 3.4) združujejo pametno kartico in čitalnik v eni sami napravi, tako da nakup dodatne opreme ni potreben, uporaba pa je preprosta [13].

3.4 Pomen dvodejavniške avtentikacije

Register.si nudi več različnih storitev, izmed katerih so določene zelo pomemebne za pravilno delovanje interneta v Sloveniji in tudi širše skupnosti. Nekatere izmed teh storitev uporabnikom omogočajo dostop do osebnih podatkov, te storitve so:

- portal za registrarje,



Slika 3.4: Ključ PKI USB ponudnika SecuTech [10].

- VPN za zaposlene v Registru,
- dostop do strežnikov Registra.

Kot je bilo že omenjeno, so storitve trenutno zaščitene na več ravneh, in sicer z uporabo uporabniškega imena ter gesla, z uporabniškimi digitalnimi potrdili, s seznamom dovoljenih IP naslovov in s pravili na požarni pregradi. Čeprav na prvi pogled to izgleda kot visoka raven zaščite, še vedno obstaja možnost, da napadalec pridobi vse potrebne informacije za uspešno prijavo. Do njih lahko pride z zlonamerno programsko opremo, vdorom ali krajo uporabniških imen in gesel, kar je lahko posledica uporabe šibkih gesel uporabnikov. Da bi odpravili to ranljivost, smo se odločili za uporabo dvodejavniške avtentikacije. Enega izmed dejavnikov bi še vedno predstavljala uporabniško ime in geslo, kot drugi dejavnik bi uporabili eno izmed varnostnih naprav, ki smo jih opisali v poglavju 3.3.

Z uvedbo dvodejavniške avtentikacije se poveča raven varnosti storitev, saj mora napadalec pridobiti napravo, ki predstavlja drugi avtentikacisjski dejavnik. Predvsem se izloči možnost nepooblaščenega oddaljenega dostopa do storitev, saj napadalec potrebuje fizični stik z uporabnikom, da mu ukrade napravo, ki generira podatke za drugi dejavnik. Uporabnik, ki mu je naprava odtujena, bo po vsej verjetnosti krajo hitro prijavil, saj ne bo imel več dostopa

do storitev.

Poglavje 4

Orodja in tehnologije

V tem poglavju bomo predstavili orodja in tehnologije, ki smo jih uporabili pri realizaciji sistema za avtentikacijo. Najprej bomo opisali močno avtentikacijsko orodje multiOTP in avtentikacijski strežnik FreeRADIUS. Zatem bomo predstavili javansko ogrodje Dropwizard v kombinaciji z orodjem za upravljanje projektov Apache Maven.

4.1 MultiOTP

MultiOTP je odprtokodno orodje, ki se uporablja za vzpostavitev močnega avtentikacijskega sistema. Preverjanje pristnosti uporabnikov poteka na osnovi uporabniškega imena in enkratnega gesla, ustvarjenega s strani generatorja gesel. MultiOTP podpira različne programske in fizične generatorje enkratnih gesel (npr. Google avtentikator, Mobile-OTP, Yubikey), ki morajo biti ceritificirani za HOTP/TOTP s strani iniciative za odprto avtentikacijo (OATH) [3].

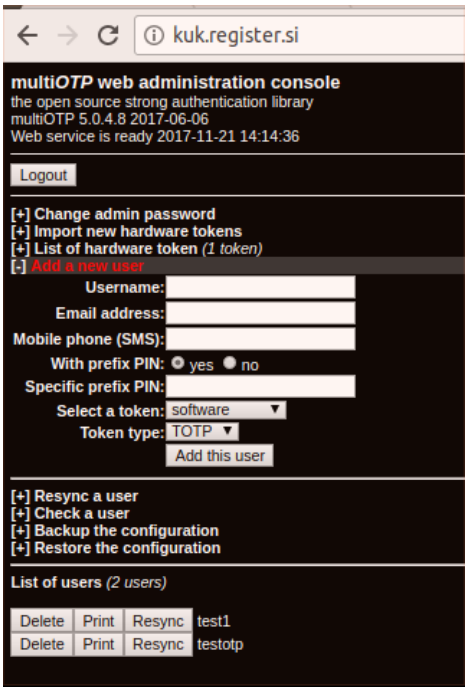
Orodje multiOTP se lahko uporablja samostojno (npr. kot močna avtentikacija za PHP spletne aplikacije), preko ukazne vrstice, kot spletna storitev ali skupaj s strežnikom RADIUS, kot je npr. TekRADIUS (za Windows naprave) ali FreeRADIUS (za unix/linux naprave). Namestitev je preprosta. Vse, kar potrebujemo, je spletni strežnik Apache in zadnja verzija multi-

OTP [6]. Odpakiran direktorij `multiotp`, ki smo ga naložili s spleta [7], prestavimo v korenski direktorij spletnega strežnika Apache `var/www/html/`, kar nam omogoča uporabljati multiOTP kot spletno storitev.

Shramba podatkov je privzeto urejena z zapisom v datoteke, kot zaledni strežnik se lahko definira strežnik MySQL. MultiOTP nato sam poskrbi za ustvarjanje in posodabljanje tabel. Možna je povezava s storitvijo LDAP ali Active Directory.

4.1.1 Uporaba orodja multiOTP

Orodje multiOTP 5.0.4.8 smo namestili na sistem CentOS 6.9. MultiOTP lahko v vlogi administratorja opravljamo kot spletno storitev, kjer preko brskalnika dostopamo do preprostega uporabniškega vmesnika (slika 4.1). Ta omogoča le uporabo osnovnih funkcionalnosti, kot so dodajanje, brisanje



Slika 4.1: Avtentikacija uporabnikov s TOTP napravo.

ter preverjanje uporabnikov in varnostnih naprav ter odklepanje uporabnika,

če je ta prevečkrat vnesel napačno geslo.

Večino stvari je mogoče skonfigurirati le preko ukazne vrstice, za uporabo katere je potrebno predhodno na sistem namestiti PHP. MultiOTP preko ukazne vrstice omogoča dodajanje ter urejanje uporabnikov, varnostnih naprav, urejanje shrambe podatkov, nastavitve sinhronizacije z LDAP ali Active Directory in nastavitve povratnih sporočil. Iz dodatka A je razvidno, kako uredimo shranjevanje podatkov v bazo in dodajanje varnostnih naprav.

Uporabnike dodamo na več različnih načinov, s predpono PIN ali brez nje, in sicer tako, da jim podamo ID želene varnostne naprave. Predpona PIN je dodatno varovalo pri prijavi uporabnika z enkratnim geslom, saj mora uporabnik pred enkratno geslo vpisati še dodeljen PIN.

V naslednjem koraku preverimo, ali avtentikacija uporabnikov s TOTP napravo deluje pravilno (slika 4.2). V primeru, da uporabnik trikrat vnese

```
[root@kuk multiotp]# ./multiotp.php -display-log  test1 327828

LOG 2017-11-21 14:31:25 notice (user test1) User OK: User test1 successfully logged in w
ith TOTP token

[root@kuk multiotp]# ./multiotp.php -display-log  test2 1234129624

LOG 2017-11-21 14:36:50 notice (user test2) User OK: User test2 successfully logged in w
ith TOTP token
```

Slika 4.2: Avtentikacija uporabnikov s TOTP napravo.

napačno geslo, se račun začasno zaklene za 300 sekund, po 6 neuspešnih poskusih se račun zaklene dokončno. Uporabnik svoj račun odklene le z uporabo dveh zaporednih enkratnih gesel ali za to poskrbi administrator. V naslenjem koraku se lotimo konfiguracije povezave strežnika FreeRADIUS z multiOTP.

4.2 FreeRadius

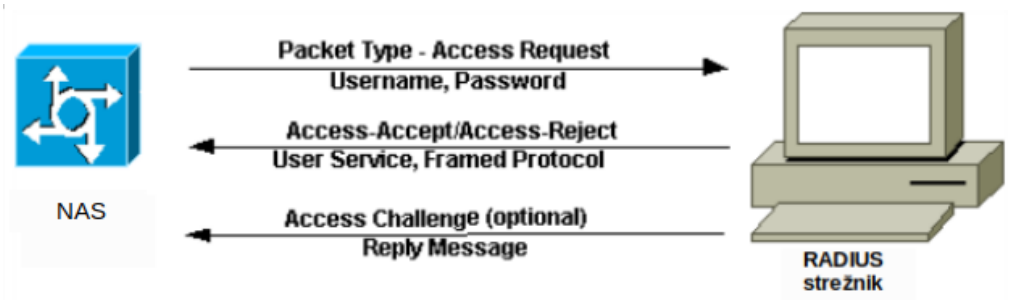
FreeRADIUS je brezplačen produkt, namenjen unixu oz. unixu podobnim operacijskim sistemom, ki deluje na osnovi protokola RADIUS [27] in je

distribuiran pod GNU GPL licenco. Vsebino paketa sestavlja strežnik RADIUS, knjižnica za odjemalca RADIUS, knjižnica PAM, modul Apache in nekaj ostalih orodij ter knjižnjic. V večini primerov se ime FreeRADIUS nanaša kar na brezplačen odprtokodni strežnik RADIUS iz tega paketa. Gre za trenutno najbolj razširjen strežnik RADIUS, uporabljen je za več kot tretjino vseh avtentikacij uporabnikov na internetu in podpira vse najpogosteje uporabljene avtentikacijske protokole [19].

Odjemalec RADIUS je v večini primerov *omrežni dostopovni strežnik* (angl. network access server) (NAS) in strežnik RADIUS je običajno pritajen proces, ki teče na unix ali napravi Windows NT. Preden je lahko povezava med odjemalcem in strežnikom uspešno vzpostavljena, vsak izmed njiju definira skupno skrivnost, ki se uporablja za avtentikacijo odjemalca.

RADIUS je protokol odjemalec - strežnik, ki za prenos uporablja protokol UDP in deluje na podlagi viskonivojskega arhitekturnega varnostnega modela AAA [18]. To pomeni, da nudi avtentikacijo, avtorizacijo in beleženje.

Komunikacija med NAS in strežnikom FreeRADIUS tekom postopka avtentikacij in avtorizacije, je podrobno prikazana na sliki 4.3. Ko želi uporabnik dostop do omrežja, NAS posreduje poizvedbo *Access-Request* strežniku RADIUS. Paket RADIUS vsebuje uporabniško ime in geslo, IP naslov odjemalca NAS in številko vrat 1812, ki je določena za RADIUS [27]. Ko strežnik FreeRADIUS prejme *Access-Request* od odjemalca NAS, v bazi naredi poizvedbo za podano uporabniško ime. Če uporabniško ime v bazi ne obstaja, se naloži privzeti profil ali pa strežnik FreeRADIUS nemudoma pošlje sporočilo *Access-Reject*. To zavrnitveno sporočilo lahko spremlja tudi besedilo z obrazložitvijo razloga zavrnitve. Če sta uporabniško ime in geslo pravilna, strežnik kot odgovor vrne *Acces-Accept* s seznamom atributov, ki bodo uporabljeni v seji. V primeru, da preverjanje pristnosti poteka z uporabo izziva, FreeRADIUS odgovori z naključnim izzivom v obliki paketa *Acces-Challenge*. Uporabnik odgovori s paketom *Access-Request*, ki vsebuje rezultat izračunanega izziva. FreeRADIUS glede na prejeti rezultat odgovori z *Access-Reject* ali *Acces-Accept* [1].



Slika 4.3: Komunikacija med NAS in srežnikom RADIUS [1].

4.2.1 Namestitev

Strežnik FreeRADIUS smo namestili na napravo s prednaloženim operacijskim sistemom CentOS 6.9. Ker smo želeli do strežnika dostopati preko odjemalca napisanega v programskem jeziku Java, je bilo potrebno v samo namestitev vključiti tudi modul JRadius. Kako je potekala namestitev strežnika FreeRADIUS ter konfiguracija strežnika z MySQL in multiOTP, je opisano v dodatku A.

Poglavje 5

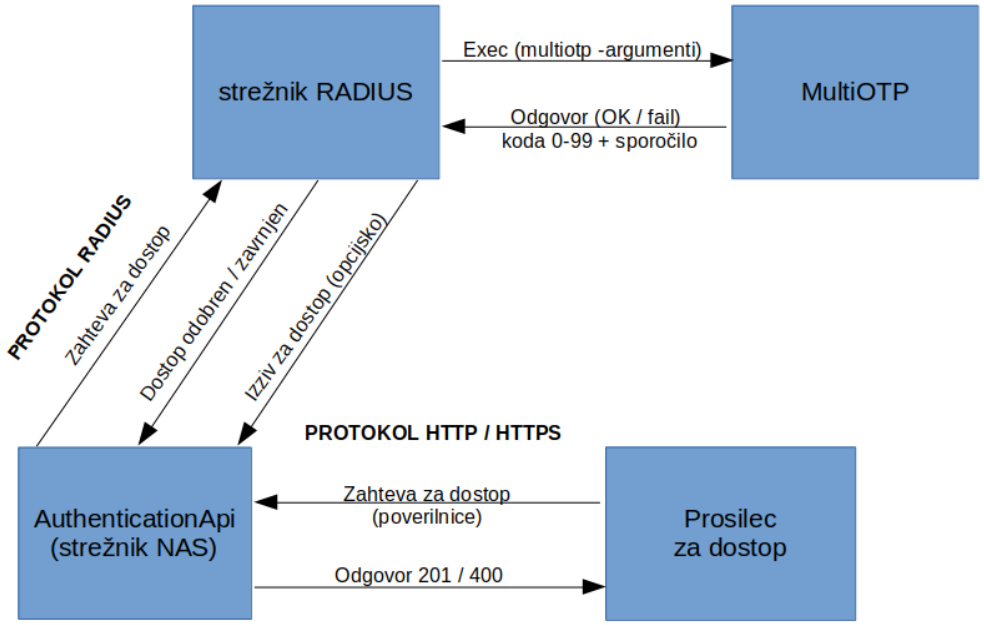
Sistem za dvodejavniško avtentikacijo

Glavni rezultat diplomske naloge je preprost sistem za dvodejavniško avtentikacijo, sestavljen iz strežnika RADIUS ter spletne storitve REST, imenovane AuthenticationApi, ki igra vlogo odjemalca strežniku RADIUS. V nadaljevanju bomo opisali arhitekturo sistema in izbiro dodatnega dejavnika za dvodejavniško avtentikacijo. Na podlagi tega smo se odločili za izbiro uporabljenih orodij in tehnologij. Na koncu bomo še ovrednotili rešitev glede na zahteve, ki smo jih definirali v uvodu.

5.1 Arhitektura in opis sistema

Najpomembnejši del celotnega sistema, ki je prikazan na sliki 5.1, predstavlja avtentikacijski strežnik FreeRADIUS z dodanim modulom multiOTP, ki skrbi za avtentikacijo uporabnikov. Avtentikacija uporabnikov poteka na podlagi podanega uporabniškega imena in gesla, ki je zgeneriran s strani enkratnega generatorja gesel.

Strežnik RADIUS smo uporabili, ker zaradi svoje AAA arhitekture hkrati omogoča tako avtentikacijo kot tudi avtorizacijo. Poleg tega omogoča še beleženje, kar nam daje dodaten nadzor nad aktivnostmi uporabnika. V



Slika 5.1: Arhitektura sistema

našem primeru za avtentikacijo in avtorizacijo skrbi modul multiOTP. To pomeni, da je dostop uporabniku odobren le na podlagi uporabniškega imena in enkratnega gesla, ki je ustvarjen s strani ključka TOTP.

S strežnikom FreeRADIUS preko protokola RADIUS komunicira spletna storitev AuthenticationApi, ki igra vlogo strežnika NAS. AuthenticatioApi s strani prosilca za dostop (npr. uporabnik, ki se želi prijaviti v portal MATsi) prejme zahtevo za dostop (slika 5.1), ki jo posreduje strežniku FreeRADIUS. AuthenticationApi od prosilca za dostop zahteva vnos poverilnic, ki so predstavljene z uporabniškim imenom in enkratnim geslom. Komunikacija med strežnikom FreeRADIUS in AutheticationApijem je mogoča, če si oba delita enako skupno skrivnost, in če zahteva za dostop prihaja z IP naslova, s katerega FreeRADIUS dovoljuje dostop. V našem primeru komunikacija med AuthenticatioApijem in strežnikom FreeRADIUS poteka preko avtentikacijskega protokola MS-CHAP.

Ko FreeRADIUS prejme zahtevo za dostop s strani AuthenticatioApija,

najprej v svoji podatkovni bazi preveri, ali uporabnik, ki se želi avtenticirati, že obstaja. V primeru, da ga ne najde, pošlje zahtevo multiOTP-ju. To naredi tako, da zažene modul multiotp, ta pa z uporabo ukaza exec zažene program *multiotp.php* z ustreznimi argumenti. V našem primeru so to uporabniško ime, geslo ter atributi vezani na protokol MS-CHAP.

V primeru, da AuthenticatioApi s strani strežnika RADIUS dobi odgovor *Access-Accept*, prosilcu za dostop pošlje odgovor v obliki kode *201 Accepted*, kar pomeni, da mu je odobren dostop. V nasprotnem primeru vrne kodo *400 Unauthorized* in zavrne dostop.

Z uporabniki, varnostnimi napravami ter nastavitvami upravljamo preko MultiOTP orodja, dostopnega preko ukazne vrstice, neposredno na strežniku, kjer je nameščen. Do strežnika dostopamo preko tunela SSH z uporabo para ključev SSH. Ko je tunel vzpostavljen, MutiOTP uporabljamo na način, ki je opisan v poglavju 4.1. MultiOTP je prav tako dostopen preko brskalnika v obliki spletne strani, kamor se lahko prijavi le administrator.

5.1.1 Implementacija spletne storitve Authentication-Api

Po postavitvi strežnika RADIUS smo se lotili implementacije spletne storitve z uporabo Dropwizarda. Gre za odprtokodno javansko ogrodje, ki se uporablja za razvoj RESTful spletnih storitev. Prednost uporabe Dropwizarda je v tem, da združuje tehnologije Jetty, Jersey in Jackson v enem paketu ter tako omogoča zelo hitro in preprosto implementacijo spletne storitve. Za upravljanje in izgradnjo projekta se največkrat uporablja orodje Apache Maven, katerega temelj predstavlja *objektni projektni model* (angl. Project Object Model) (v nadaljevanju POM) [12].

POM je xml predstavitev projekta Maven v datoteki imenovani *pom.xml*. Datoteka POM vsebuje vse potrebne informacije o projektu kot tudi konfiguracijske vtičnike, ki so potrebni za izgradnjo projekta. Glavni del datoteke POM predstavlja *seznam odvisnosti* (angl. dependency list), na osnovi katerih poteka izgradnja in zagon projekta [5]. Povezavo s strežnikom RADIUS

smo omogočili z uporabo knjižnjice jRadius, ki smo jo dodali na seznam odvisnosti.

Za shranjevanje nastavitev projekta bomo uporabili človeku prijazen standard za postrojenje objektov YAML, ki omogoča uporabo poljubnih podatkovnih tipov. Na sliki 5.2 je prikazan primer datoteke YAML, v kateri so navedene vse nastavitve, ki jih potrebujemo v našem projektu. Znotraj razdelka *radiusServer* so vsi potrebni podatki za povezavo s strežnikom RADIUS. Podatki o tem, na kakšen način bomo dostopali do storitve, se nahajajo znotraj razdelka *applicationConnectors*. Dostop do storitve je mogoč preko protokla http ter njegove zavarovane različice https, ki uporablja SSL. Le-ta omogoča komunikacijo med strežnikom in odjemalcem v šifrirani obliki in tako zagotavlja varno komunikacijo.

```
radiusServer:
    radiusHost: kuk.register.si
    radiusAuthPort: 1812
    radiusAcctPort: 1813
    radiusSecret: testmultiotp
server:
    applicationConnectors:
        - type: http
          port: 8080
        - type: https
          port: 8083
          keyStorePath: ./src/test/resources/keystore.jks
          keyStorePassword: abc123
          keyStoreType: JKS
```

Slika 5.2: Primer datoteke YAML

Storitev je sestavljena iz ene *končne točke* (angl. endpoint) imenovane `/login` in predstavlja referenco na naslov, ki sprejema poizvedbe s strani odjemalca. V našem primeru do nje dostopamo preko naslova `localhost/authentication/login`.

Spletna storitev za dostop do končne točke `/login` zahteva vnos poverilnic v obliki uporabniškega imena in gesla. V kodi smo to zagotovili z implementacijo *osnovne avtentikacije HTTP* (angl. HTTP basic authentication) z uporabo Dropwizardowega vmesnika *Authenticator*, ki sprejme poverilnice

tipa *BasicCredentials*. Znotraj metode `authenticate()` (slika 5.3) smo z uporabo odjemalca RADIUS preverili, ali je uporabniku, ki se želi avtenticirati, odobren dostop ali ne.

Ustvarili smo seznam atributov *AttributeList*, ki vsebuje podatke o naslovu, kjer se nahaja strežnik RADIUS, in uporabniško ime ter geslo uporabnika. Strežniku RADIUS posredujemo atribute v obliki poizvedbe za dostop *AccessRequest*, ta pa nam odgovori z *AccessAccept* ali *AccessReject*. Če je uporabniku odobren dostop, nam AuthenticatioApi vrne odgovor *202 Accepted*, v nasprotnem primeru vrne *401 Unauthorized*. To smo zagotovili z

```
@Override
public Optional<User> authenticate(BasicCredentials credentials) throws AuthenticationException {
    AttributeFactory.loadAttributeDictionary("net.jradius.dictionary.AttributeDictionaryImpl");
    boolean isAuthenticatedUser = false;
    try {
        InetAddress address = InetAddress.getByName(config.getRadiusServer().getRadiusHost());
        final RadiusClient rc = new RadiusClient(address,
                config.getRadiusServer().getRadiusSecret(),
                config.getRadiusServer().getRadiusAuthPort(),
                config.getRadiusServer().getRadiusAcctPort(), 5);

        final AttributeList attributes = new AttributeList();
        attributes.add(new Attr_NASIdentifier(config.getRadiusServer().getRadiusHost()));
        attributes.add(new Attr_UserName(credentials.getUsername()));
        attributes.add(new Attr_UserPassword(credentials.getPassword()));

        final AccessRequest accessRequest = new AccessRequest(rc, attributes);
        final RadiusAuthenticator auth = new MSCHAPv1Authenticator();
        RadiusPacket reply = rc.authenticate(accessRequest, auth, 0);
        isAuthenticatedUser = reply instanceof AccessAccept;

    } catch (UnknownHostException ex) {
        LOG.error("Can not find host.", ex);
        throw new AuthenticationException("Not able to authenticate user");
    } catch (RadiusException | IOException | NoSuchAlgorithmException ex) {
        LOG.error("Radius Client exception.", ex);
        throw new AuthenticationException("Not able to authenticate user");
    }

    if(isAuthenticatedUser) {
        return Optional.of(new User(credentials.getUsername()));
    } else {
        return Optional.empty();
    }

}
```

Slika 5.3: Metoda `authenticate()`, ki skrbi za avtentikacijo uporabnika

uporabo Dropwizardove možnosti zavarovanja virov, kjer v metodi `login()` (slika 5.4) parameter `user`, ki predstavlja uporabnika, označimo z besedo `@Auth`. S tem smo dosegli, da je klic metode `login()` dovoljen le, ko gre za predhodno avtenticiranega uporabnika (slika 5.3).

```
@Path("/authentication")
@Consumes({MediaType.APPLICATION_JSON})
@Produces({MediaType.APPLICATION_JSON})
public class AuthResource {

    @GET
    @Path("login")
    public Response login(@Auth User user) {

        return Response.accepted().build();
    }
}
```

Slika 5.4: Zavarovana metoda `login()`

5.2 Izbira dodatnega dejavnika za dvodejavniško avtentikacijo

Na izbiro dodatnega dejavnika za dvodejavniško avtentikacijo so vplivali naslednji kriteriji:

- varnost uporabe,
- cenovna sprejemljivost,
- razširljivost na storitve celotnega Arnesa in
- preprosta uporaba.

Glede na zgoraj navedene kriterije smo se odločili, da se izognemo vsem programskim rešitvam (npr. Google Authenticator, SMS sporočila). Čeprav so daleč najcenejše in preproste za uporabo, imajo z vidika varnosti precej ranljivosti, o katerih smo govorili v poglavju 3.3.

Zatem smo se osredotočili na naprave PKI, namenjene varnemu shranjevanju ter uporabi digitalnih potrdil in zasebnih ključev, ki omogočajo visoko raven zaščite. Pametne kartice za svoje delovanje potrebujejo čitalec, zaradi česar se za njihovo uporabo nismo odločili. Zanimiva se nam je zdela ideja uporabe naprav PKI v obliki ključev USB. Po testiranju nekaj ključev

USB različnih proizvajalcev smo ugotovili, da se pojavi težava pri podprtosti na sistemih linux in je tako uporaba ključev na nekaterih distribucijah nemogoča.

Najbolj zanimive so se nam zdele naprave U2F, saj so preproste in varne za uporabo. Težava, ki se pojavi pri uporabi protokola U2F, je, da še ni podprt s strani vseh brskalnikov, med drugim tudi Firefoxa, zaradi česar se za njegovo uporabo nismo odločili.

Nato smo se poglobili še v strojne generatorje enkratnih gesel. Večina jih deluje na podlagi algoritma HOTP, pri čemer nas je zmotilo neznano dolgo trajanje veljavnosti gesla. To predstavlja grožnjo v primeru, da geslo nekdo prestreže, saj je lahko uporabljeno kadarkoli. Odločili smo se, da preizkusimo enega izmed generatorjev TOTP, saj je geslo veljavno le določeno število sekund. Izbrali smo ključ TOTP UniOTP500 (slika 3.1), ki je produkt podjetja SecuTech. V to nas je prepričala ugodna cena ključev in možnost brezplačnega testiranja.

5.3 Ovrednotenje rešitve

V tem razdelku bomo ovrednotili rešitev glede na funkcionalne in nefunkcionalne zahteve, ki smo jih definirali v uvodnem poglavju.

5.3.1 Funkcionalne zahteve

Dostop do portala za registrarje od koderkoli

Dostop do spletnega portala za registrarje MATsi je trenutno omejen s seznamom dovoljenih IP naslovov. Registrar pred sklenitvijo pogodbe o sodelovanju pri registraciji domen navede, s katerega IP naslova bo dostopal do MATsija. Ta naslov se doda na seznam dovoljenih IP naslovov in tako lahko registrar do strani, kjer se nahaja MATsi, pride le z navedenega IP naslova. Znotraj portala lahko registrar doda še do 5 IP naslovov, ki jih lahko poljubno spreminja.

Z uporabo nove rešitve bodo registrarji imeli možnost dostopa do portala od koderkoli, saj dostop ne bo vezan na IP naslov. Za dostop bo potrebno, da imajo registrarji v lasti ustrezen generator TOTP, z uporabo katerega bodo dostopali do svojega uporabniškega računa.

5.3.2 Nefunkcionalne zahteve

Varnost

Pomemben vidik celotne arhitekture je varnost. Kot že opisano v poglavju 5.2, smo se odločili, da bo poleg uporabniškega imena in gesla pri prijavi potrebno še geslo, ustvarjeno s strani generatorja TOTP. Kljub temu, da bi z uporabo naprave U2F zagotovili višjo varnost, je bil za nas pomemben dejavnik to, da je uporaba podprta s strani vseh brskalnikov.

Zaradi uporabe generatorja TOTP se v arhitekturi pojavi tveganje za napad s posrednikom v realnem času. Napadu se poskušamo izogniti s temeljito analizo obnašanja odjemalca, in sicer s preverjanjem, s katerih IP naslovov se uporabnik ponavadi prijavlja. V primeru, da bi šlo za prijavo s sumljivega naslova ali za prijavo različnih uporabnikov z istega IP naslova, bi se ustrezno odzvali.

Na strani odjemalca lahko pride do DDOS napada, katerega namen je preobremeniti sistem s pošiljanjem večje količine poizvedb. V tem primeru je potrebno preučiti, od kod prihajajo poizvedbe, in odreagirati z ustreznimi blokadami IP naslovov.

Povezava med odjemalcem in AuthenticatioApijem poteka preko protokola HTTPS, kar ščiti pred napadom s posrednikom. Če napadalec pridobi podatke poslane s strani odjemalca, se z njimi ne more prijaviti v sistem, saj so podatki šifrirani. Sistem je zaščiten tudi pred napadom z golo silo z omejitvijo neuspešnih prijav za uporabnika.

Drugih tveganj za napade ni, saj bosta tako avtentikacijski strežnik kot AuthenticationApi zaščitena s požarno pregrado in vsak v svojem omrežju. Omejitev dostopov med strežniki bo urejena z uporabo seznama dovoljenih

IP naslovov.

Cena

Izbira ustreznega dejavnika je odvisna od pravega razmerja med ceno in varnostjo. Cena izbranega generatorja TOTP je med 20 in 27 € odvisno od izbrane količine, kar se nam zdi sprejemljivo. Ostale uporabljene tehnologie so odprtokodne, kar pomeni, da so brezplačne.

Poglavje 6

Sklepne ugotovitve

V diplomskem delu smo se ukvarjali s tem, kako postaviti varen avtentikacijski sistem, ki bi omogočal dvodejavniško avtentikacijo. Povod za to je bila želja registrarjev, ki ne želijo biti omejeni glede prijave v portal za registrarje s seznamom dovoljenih IP naslovov.

Predstavili smo avtentikacijske algoritme in protokole, ki so pomembni za postavitev sistema. Poudarek je bil na predstavitvi varnostnih naprav in njihovih programskih rešitev. Med seboj smo jih primerjali z namenom, da izberemo najprimernejšo in jo uporabimo kot dodatni dejavnik pri prijavi uporabnikov. Glede na izbrane kriterije smo ocenili, da je najprimernejši generator enkratnih gesel podjetja SecuTech.

Rešitev sistema predstavlja spletna storitev AuthenticationApi, do katere dostopamo preko poizvedb HTTP. Storitev nastopa v vlogi odjemalca strežnika FreeRADIUS, ki skrbi za avtentikacijo uporabnikov z uporabo orodja MultiOTP.

Naš cilj v prihodnosti je vzpostaviti sistem, kjer bo avtentikacija uporabnikov potekala z uporabo odprtega avtentikacijskega standarda U2F. Zato bomo v prihodnje uporabili rešitev podjetja Yubico, in sicer Yubikey 4. generacije. Ta implementira različne avtentikacijske protokole, med drugim tudi FIDO U2F, Yubico OTP, TOTP ter HOTP. Tako bomo na začetku uporabili TOTP, kasneje, ko bo U2F nekoliko bolj razširjen, bomo uporabili

tega. Uporaba Yubikey 4. generacije predstavlja cenejšo rešitev, saj nam ne bo potrebno ločeno kupovati naprave TOTP in U2F.

Literatura

[1] How Does RADIUS Work? *Cisco Systems, Inc.* Dosegljivo: `https://www.cisco.com/c/en/us/support/docs/security-vpn/remote-authentication-dial-user-service-radius/12433-32.html`, 2006. [Dostopano 31.01.2018].

[2] About CENTR. *CENTR.* Dosegljivo: `https://centr.org/about/about-centr.html`, 2018. [Dostopano 31.01.2018].

[3] About Oath. *OATH.* Dosegljivo: `https://openauthentication.org/about-oath/`, 2018. [Dostopano 31.01.2018].

[4] About the FIDO Alliance. *FIDO Alliance.* Dosegljivo: `https://fidoalliance.org/about/overview/`, 2018. [Dostopano 31.01.2018].

[5] Introduction to POM. *The Apache Software Foundation.* Dosegljivo: `https://maven.apache.org/guides/introduction/introduction-to-the-pom.html`, 2018. [Dostopano 20.08.2018].

[6] multiOTP open source dokumentacija. *SysCo systèmes de communication sa.* Dosegljivo: `https://github.com/multiOTP/multiotp/wiki`, 2018. [Dostopano 20.08.2018].

[7] multiOTP open source izvorna koda. *SysCo systèmes de communication sa.* Dosegljivo: `https://download.multiotp.net/`, 2018. [Dostopano 20.08.2018].

[8] Navodila za uporabo novih generatorjev gesel. Dosegljivo: https://www.unicreditbank.si/si/prebivalstvo/digitalno-bancnistvo/navodila-zeton.html, 2018. [Dostopano 31.01.2018].

[9] U2F Technical Overview. *Yubico*. Dosegljivo: https://developers.yubico.com/U2F/Protocol_details/Overview.html, 2018. [Dostopano 31.01.2018].

[10] UniMate USB. *SecuTech Solutions PTY LTD*. Dosegljivo: https://www.esecutech.com/products/unimate/unimate-usb, 2018. [Dostopano 31.01.2018].

[11] UniOTP 500. *SecuTech Solutions PTY LTD*. Dosegljivo: https://www.esecutech.com/products/uniotp/uniotp-500-510, 2018. [Dostopano 31.01.2018].

[12] What is Maven? *The Apache Software Foundation*. Dosegljivo: https://maven.apache.org/what-is-maven.html, 2018. [Dostopano 20.08.2018].

[13] What is so smart about smart cards? *Gemalto*. Dosegljivo: https://www.gemalto.com/companyinfo/smart-cards-basics, 2018. [Dostopano 31.01.2018].

[14] M. Boodaei. Real-Time Phishing Takes Off. Dosegljivo: https://securityintelligence.com/real-time-phishing-takes-off/, 2010. [Dostopano 31.01.2018].

[15] S. L. Pfleeger C. P. Pfleeger and J. Margulies. *Security in Computing*. Prentice Hall, 5 edition, 2015.

[16] CENTR. 2 Factor Authentication Survey. Dosegljivo: https://centr.org/statistics/surveys.html, 2018. [Dostopano 23.1.2019].

[17] N. Dan. Soft vs Hard Tokens. *Hypersecu Information Systems, Inc.* Dosegljivo: `https://hypersecu.com/blog/90-soft-vs-hard-tokens`, 2015. [Dostopano 31.01.2018].

[18] C. dee Laat, G. Gross, L. Gommans, J. Vollbrecht, and D. Spence. Generic AAA Architecture. RFC 2903, RFC Editor, 2000.

[19] D. Van der Walt. *FreeRADIUS Beginner's Guide*. Packt Publishing Ltd, 2011.

[20] R. Canetti H. Krawczyk, M. Bellare. HMAC: Keyed-Hashing for Message Authentication. RFC 2104, RFC Editor, 1997.

[21] A. Jurišić. Diffie-hellmanov dogovor o ključu, 2. del. *Presek*, 34(1):25–30, 2006.

[22] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104, RFC Editor, 1997.

[23] B. Lloyd and W. Simpson. PPP Authentication Protocols. RFC 1334, RFC Editor, 1992.

[24] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Rannen. HOTP: An HMAC-Based One-Time Password Algorithm. RFC 4226, RFC Editor, 2005.

[25] D. M'Raihi, S. Machani, M. Pei, and J. Rydell. TOTP: Time-Based One-Time Password Algorithm. RFC 6238, RFC Editor, 2011.

[26] NIST. FIPS 180-3. Dosegljivo: `https://csrc.nist.gov/CSRC/media/Publications/fips/180/3/archive/2008-10-31/documents/fips180-3_final.pdf`, 2008. [Dostopano 31.01.2019].

[27] C. Rigney, S. Willens, A. Rubens, and W. Simpson. Remote Authentication Dial In User Service (RADIUS). RFC 2865, RFC Editor, 2000.

[28] M. Rouse. PKI (public key infrastructure) *TechTarget*. Dosegljivo: `http://searchsecurity.techtarget.com/definition/PKI`, 2014. [Dostopano 31.01.2018].

[29] M. Rouse. Two-factor authentication (2FA). *TechTarget*. Dosegljivo: `http://searchsecurity.techtarget.com/definition/two-factor-authentication`, 2015. [Dostopano 31.01.2018].

[30] W. Simpson. PPP Challenge Handshake Authentication Protocol (CHAP). RFC 1334, RFC Editor, 1994.

[31] D.R. Stinson. *Cryptography: theory and practice*. CRC press, 4 edition, 2018.

[32] G. Zorn. Microsoft PPP CHAP extensions, Version 2. RFC 2759, RFC Editor, 2000.

[33] G. Zorn and S. Cobb. Microsoft PPP CHAP Extensions. RFC 2433, RFC Editor, 1998.

Dodatek A

Namestitev in konfiguracija strežnika FreeRADIUS

Strežnik FreeRADIUS smo namestili na linux operacijski sistem CentOS 6.9. Ker modul JRadius ni vključen v sistemskem inštalacijskem paketu rpm strežnika FreeRADIUS, ga je bilo potrebno dodati ročno. To smo storili tako, da smo z interneta naložili izvorno kodo paketa FreeRADIUS, ki vsebuje modul JRadius, in potem ročno naredili rpm paket z vsebovanim modulom ter ga namestili, kar je razvidno iz naslednjih ukazov:

```
# wget http://ftp.redhat.com/pub/redhat/linux/enterprise/
  6Server/en/os/SRPMS/freeradius-2.2.6-7.el6_9.src.rpm
# rpm2cpio freeradius-2.2.6-7.el6_9.src.rpm | cpio -i
  -make-directories
# tar xvjf freeradius-server-2.2.6.tar.bz2
# yum install rpm-build
# yum install redhat-rpm-config
# yum install make
# yum install gcc
# mkdir -p ~/rpmbuild/{BUILD,RPMS,SOURCES,SPECS,SRPMS}
# echo '%_topdir %(echo $HOME)/rpmbuild' > ~/.rpmmacros
# echo rlm_jradius >> $RPM_BUILD_DIR/freeradius-server-
  %{version}/src/modules/stable
# cd ~/rpmbuild/
# rpmbuild -v -bb --clean SPECS/freeradius.spec
# rpm -Uvh *.rpm.
```

Po namestitvi se vse konfiguracijske datoteke nahajajo v direktoriju `/etc/raddb`.

Konfiguracija strežnika FreeRADIUS z MySQL

Za shranjevanje podatkov bomo uporabili bazo MySQL. Najprej je potrebno glede na zahteve FreeRADIUSa ustvariti podatkovno bazo z vsemi tabelami, ki so potrebne za shranjevanje zapisov s strani strežnika FreeRADIUS, kar smo storili z naslednjimi ukazi:

```
#mysql -u root -p
>CREATE DATABASE radius;
>GRANT ALL ON radius.* TO radius@localhost IDENTIFIED BY
 "radpass"; exit

#mysql -u root -p radius < /etc/raddb/sql/mysql/mysql.sql
#mysql -u root -p radius < /etc/raddb/sql/mysql/admin.sql
#mysql -u root -p radius < /etc/raddb/sql/mysql/schema.sql
```

Zatem dodamo v datoteko `/etc/raddb/radiusd.conf` vrstico z besedilom `$INCLUDE sql.conf`, ki pove, kje naj FreeRADIUS poišče vse podatke, ki jih potrebuje za dostop do podatkovne baze. Nato v datoteko dodamo naslednje zapise:

```
database = "mysql"
driver = "rlm_sql_${database}"
server = "localhost"
login = "radius"
password = "radpass"
radius_db = "radius"
```

Nato še v vse konfiguracijske datoteke strežnika FreeRADIUS dodamo ustrezne vrstice, ki nakazujejo, da se avtorizacija, avtentikacija ter beleženje izvajajo preko modula sql.

Konfiguracija strežnika FreeRADIUS z MultiOTP

Najprej je bilo potrebno znotraj MultiOTP urediti shranjevanje podatkov v vnaprej ustvarjeno podatkovno bazo, saj nismo želeli, da se uporabniki zapisujejo v datoteke, kar naredimo z naslednjimi ukazi:

```
#./multiotp.php -debug -congif backend-type=mysql
 sql_server=localhost sql_database=multiotp
 sql_username=multiotp sql_password:=620650
#./multiotp.php -debug -initalize-backend
```

Sedaj lahko pričnemo z dodajanjem uporabnikov in varnostnih naprav. Naprave bomo uvozili v obliki datoteke xml, ki jo pridobimo s strani proizvajalca. Format datoteke je vnaprej definiran tako, kot je razvidno iz slike A.1. Za vsako konkretno napravo torej potrebujemo podatke o tipu algoritma *Key Algorithm*, ki bo uporabljen za ustvarjanje gesel, intervalu osveževanja *TimeInterval*, skrivnosti *Secret* in ID naprave *SerialNo* ter tip in dolžino izpisa *ResponseFormat*, ki ga naprava vrača.

```
<?xml version="1.0" encoding="UTF-8"?>
<KeyContainer Version="1.0"
    xmlns="urn:ietf:params:xml:ns:keyprov:pskc"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
    <KeyPackage>
        <DeviceInfo>
            <SerialNo>STTS11001738</SerialNo>
            <Model>UniOTP500</Model>
        </DeviceInfo>
        <Key Algorithm="urn:ietf:params:xml:ns:keyprov:pskc:totp" Id="1">
            <AlgorithmParameters>
                <ResponseFormat Length="6" Encoding="DECIMAL"/>
            </AlgorithmParameters>
            <Data>
                <Secret>
                    <PlainValue>DA6JksJaPd1n1CEZNljPLotjkg4=</PlainValue>
                </Secret>
                <Time>
                    <PlainValue>0</PlainValue>
                </Time>
                <TimeInterval>
                    <PlainValue>60</PlainValue>
                </TimeInterval>
                <TimeDrift>
                    <PlainValue>0</PlainValue>
                </TimeDrift>
            </Data>
        </Key>
    </KeyPackage>
</KeyContainer>
```

Slika A.1: Primer datoteke xml s podatki o varnostni napravi.

Strežnik FreeRADIUS je potrebno ustrezno skonfigurirati z orodjem MultiOTP, saj želimo, da se preko njega izvaja avtentikacija uporabnikov. Strežniku FreeRADIUS dodamo MultiOTP kot modul, kar pomeni, da FreeRADIUS z MultiOTP-jem komunicira direktno preko klicev ustreznih funkcij.

Najprej ustvarimo datoteko `/etc/raddb/modules/multiotop`, katere vsebina je prikazana spodaj.

```
exec multiotp {
  wait = yes
  input_pairs = request
  output_pairs = reply
  program = "/var/www/html/multiotp/multiotp.php '%{User-Name}'
    '%{User-Password}' -request-nt-key
    -src=%{Packet-Src-IP-Address}
    -chap-challenge=%{CHAP-Challenge}
    -chap-password=%{CHAP-Password}
    -ms-chap-challenge=%{MS-CHAP-Challenge}
    -ms-chap-response=%{MS-CHAP-Response}
    -ms-chap2-response=%{MS-CHAP2-Response}"
  shell_escape = yes
}
```

Iz vsebine datoteke je razvidno, da komunikacija z MultiOTP-jem poteka s klicem razreda *multiotp.php* z ustreznimi argumenti, ki so potrebni za avtentikacijo uporabnika.

Zatem na začetek razdelka *policy* dodamo spodnjo kodo, ki *Auth-Type* nastavi v multiop.

```
multiotp.authorize {
  if (!control:Auth-Type) {
    update control {
      Auth-Type := multiotp
    }
  }
}
```

Nato v datoteki `/etc/raddb/sites-available/default` znotraj sklopa `authorize{}` dodamo vrstico z besedilom `multiotp`, v sklop `authenticate{}` pa dodamo sledeče:

```
authenticate {
  Auth-Type multiotp {
    multiotp
  }
  ...
}
```

To pomeni, da bosta tako avtentikacija kot tudi avtorizacija potekali preko modula multiotp. Zakomentiramo še vse vrstice, ki vsebujejo chap ali mschap, saj teh dveh modulov ne bomo uporabljali. Na enak način popravimo še datoteko `/etc/raddb/sites-available/inner-tunnel`.

V datoteki `/etc/raddb/client.conf` definiramo še, iz katerih IP naslovov bomo dostopali do strežnika, in določimo skupno skrivnost za vsakega odjemalca posebej.