Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Porovnanie API tokenov

Bakalárska práca

2023
Jitka Muravská

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Porovnanie API tokenov

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	RNDr. Richard Ostertág, PhD.

Bratislava, 2023
Jitka Muravská

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Jitka Muravská
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Porovnanie API tokenov
Comparison of API Tokens

Anotácia: REST aplikačné rozhrania (API) často nebývajú čisto verejné. V takom prípade treba pri implementácii API riešiť aj jeho bezpečnosť. Väčšina schém zabezpečenia API používa token, ktorý je súčasťou jednotlivých požiadaviek. Tieto tokeny sú nejakým spôsobom spojené s identitou a autorizáciou používateľa. Aplikačné rozhranie prevezme požiadavku, extrahuje token, a podľa pravidiel prístupu rozhodne ako pokračovať.

Cieľom práce je porovnanie rôznych API tokenov (napríklad: OAuth 2.0, JWT, PASETO, Protobuf Tokens, Authenticated Requests, Macaroons, Biscuits, …). Prvým krokom bude zozbieranie a popísanie rôznych v praxi používaných API tokenov. Následne sa vykonajú porovnania ich výhod a nevýhod (napríklad rýchlosť, jednoduchosť použitia) na jednoduchej základnej aplikácii.

Vedúci: RNDr. Richard Ostertág, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: prof. RNDr. Martin Škoviera, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
bez obmedzenia

Dátum zadania: 12.10.2022

Dátum schválenia: 13.10.2022 doc. RNDr. Dana Pardubská, CSc.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie:

Abstrakt

Kľúčové slová:

Abstract

Keywords:

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Kapitola 1

Schémy zabezpečenia API a využitie tokenov

V tejto kapitole uvedieme viaceré známe prístupy riešenia autentifikácie a autorizácie. Stručne objasníme ako fungujú a aké sú ich výhody a nevýhody. Detailnejšie sa pozrieme na prístup využívajúci API tokeny a samotné tokeny a ich rozdelenie, ktorým sa venuje naša práca.

1.1 Zabezpečenie bez autentifikácie

V prípade, že je aplikačné rozhranie plne verejné, nepotrebuje žiadnu schému zabezpečenia. Ľubovoľný používateľ môže volať rozhranie bez predchádzajúcej autentifikácie a neexistuje spôsobom ako obmedziť prístup k volaniam rozhrania.

Jediným identifikátorom autora požiadavky je jeho IP adresa. To je však veľmi slabý identifikátor a nedáva nám veľké možnosti v obmedzení prístupu k rozhraniu.

Hlavnou výhodou tohto riešenia je jednoduchosť implementácie, rozhranie nepotrebuje uchovávať žiadne dáta o používateľoch a všetky požiadavky sú jednoduché, ich rýchlosť závisí len od rýchlosti samotného volania a rýchlosti siete.

Nevýhodou je samozrejme strata kontroly nad prístupom k rozhraniu, ktorá sa obmedzila buď na nejakú kontrolu na základe IP adresy alebo úplne vymizla. A jediným rozumných obmedzením je obmedzenie počtu požiadaviek za nejaký časový interval, či už pre konkrétne IP adresy alebo pre celé rozhranie.

1.2 Zabezpečenie bez schémy

Najjednoduchšie riešenie autentifikácie a autorizácie je posielanie prihlasovacích údajov v každej požiadavke na rozhranie. Klient jednoducho pripojí prihlasovacie údaje ku každej požiadavke a rozhranie si ich overí vo svojej databáze a v prípade úspechu vráti

požadované dáta.

Toto riešenie zjavne nie je vhodné ak sa niekde v rámci komunikácie nachádza nezabezpečené spojenie. Útočník, ktorý by takúto komunikáciu zachytil, by mal jednoduchý prístup k prihlasovacím údajom používateľa.

Používanie nezabezpečeného spojenia je všeobecne nebezpečné bez ohľadu na schému zabezpečenia a typ prenášaného údaju používaného na autorizáciu, preto ďalej v práci budeme predpokladať, že spojenia s každým koncovým bodom sú zabezpečené pomocou SSL/TLS.

Aj v prípade zabezpečeného spojenia však existujú zraniteľnosti [12]. Prihlasovacie údaje sú zriedkavo menený identifikátor a teda po ich odchytení sa dajú dlho zneužívať. Okrem samotného úniku citlivých údajov a z toho vyplývajúcich nepríjemností pre používateľa má tento prístup aj ďalšie nevýhody.

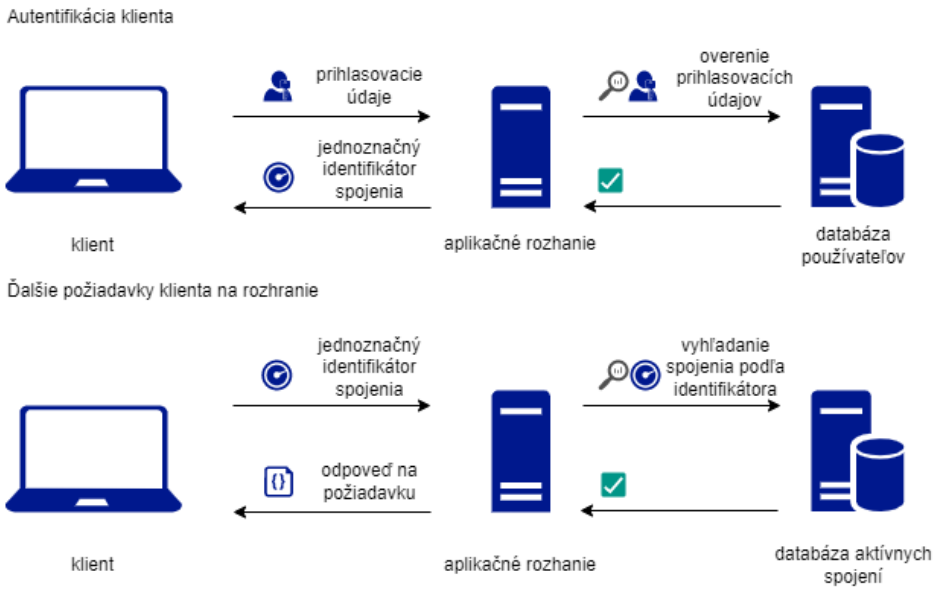
Útočník často nezneužíva získané heslá hneď, ale ukladá ich do databázy odchytených hesiel a následne túto databázu môže použiť pri útokoch. Tento prístup sa nazýva *slovníkový útok* a môže byť veľmi efektívny. [11]. Ľudia tiež často používajú podobné heslá na všetky svoje účty. Potom sa podľa istých vzorov odhalených v sade hesiel jedného používateľa dajú ľahšie uhádnuť ďalšie heslá. [3].

1.3 Schéma zabezpečenia s využitím identifikátora spojenia

Prvé riešenie, kde môžeme hovoriť o nejakej schéme zabezpečenia, nie len o existencii prihlasovacích údajov a ich overení pri každej požiadavke, je sledovanie každého aktívneho spojenia (angl. session) s rozhraním.

Pri prvej požiadavke sa používateľ autentifikuje voči serveru, ten si vytvorí a uloží záznam o spojení. Tento záznam môže obsahovať rôzne základné informácie a o spojení ako čas vytvorenia a informácie o používateľovi, s ktorým je spojenie vytvorené. Dôležitá časť záznamu je jednoznačný identifikátor spojenia v podobe náhodného reťazca. Následne klientovi vráti identifikátor spojenia. Všetky nasledujúce požiadavky klienta budú obsahovať tento identifikátor, podľa ktorého vie server určiť, ktorému používateľovi patria, teda či je autentifikovaný, prípadne či má právo vykonať dané volanie, ak rozhranie implementuje nejakú autorizačnú schému. Popísaná schéma je znázornená na obrázku 1.2.

Oproti predošlému riešeniu pribudla potreba manažovať spojenia, no veľkou výhodou je, že sa už neposielajú citlivé prihlasovacie údaje v každej požiadavke. Rozhranie však stále potrebuje pri každej požiadavke preložiť identifikátor na záznam o používateľovi.



Obr. 1.1: Autentifikačná schéma s použitím identifikátora spojenia.

1.4 Schéma zabezpečenia využívajúca API kľuče

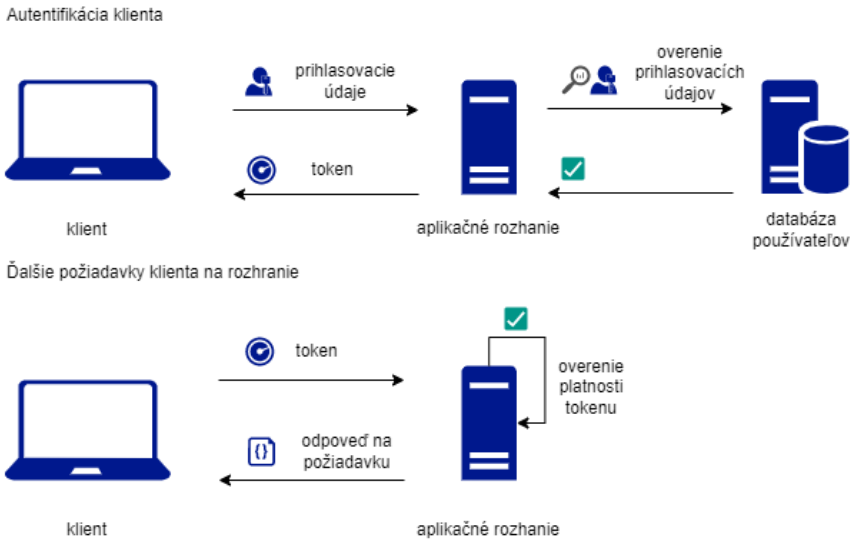
API kľúč je identifikátor používaný na pripojenie k rozhraniu. Identifikuje však aplikáciu alebo službu, nie konkrétneho používateľa. Ak potrebuje aplikácia používať rozhranie, ktoré nie je verejné a na autentifikáciu používa API kľúče, musí si najprv vyžiadať API kľuč a následne ho posielať s každou požiadavkou.

Tento kľúč môže byť zároveň spojený s právami umožňujúcimi vykonávať určité volania, ak dané rozhranie poskytuje rôzne úrovne prístupu.

Použitie API kľúčov nie je veľmi bezpečné, lebo kľúče majú neobmedzenú platnosť a ak príde k ich odchyteniu, jediný spôsob ako zvrátiť ich zneužitie je ich zneplatniť a vygenerovať nové. Ďalšia zraniteľnosť API kľúčov často nastáva pri ich uložení na nesprávne miesto.

Napríklad pri webovej aplikácii nie je bezpečné ukladať kľúč na klientskej strane, ale na serveri a odtiaľ vykonávať API volania. Rovnako pri mobilnej aplikácii nie je bezpečné ukladať kľúč priamo v aplikácii, lebo sa dá získať reverzným inžinierstvom. [2]

Preto sa vo všeobecnosti API kľúče nepoužívajú na ochranu kritických alebo citlivých volaní a zdrojov rozhrania. Naopak vhodné sú na zvýšenie kontroly nad požiadavkami na rozhranie. Je ľahké pomocou nich kontrolovať frekvenciu, či absolútny počet volaní a prípadne monetizovať ich používanie.



Obr. 1.2: Autentifikačná schéma s použitím tokenu.

1.5 Schéma zabezpečenia využívajúca API tokeny

API token (ďalej token) je identifikátor, ktorý slúži na autorizáciu prípadne aj identifikáciu používateľa pri prístupe k rozhraniu. Token môže mať viaceré podoby, jednotlivým typom a formátom sa venujú podkapitoly 1.6 a 1.7.

Tok v rámci schémy zabezpečenia rozhrania funguje podobne ako pri využití identifikátora spojenia, ktorý sme popísali v podkapitole 1.3. Líši sa najmä v tom, že token môže niesť pridanú informáciu a rozhranie vie jeho platnosť overiť bezstavovo. Na dosiahnutie tejto výhody je nutné použiť štruktúrovaný formát tokenu. Možné formáty tokenov detailnejšie popíšeme v podkapitole 1.7.

1.6 Typy tokenov

Tokeny využívané v schéme zabezpečenia rozhrania sa dajú rozdeliť podľa ich generického využitia na niekoľko typov. Nie v každom scenári máme explicitne dané aký token využiť, no niektoré sú často vhodnejšie ako iné. V tejto kapitole predstavíme viaceré typy tokenov a to prístupový (angl. access), nositeľský (angl. bearer), obnovovací (angl. refresh) a identifikačný token.

1.6.1 Prístupový token

Prístupový token je najzákladnejší a najčastejšie používaný typ tokenu. Tento token vygeneruje autentifikačný server po autentifikácii používateľa a ďalej slúži na autorizáciu v rozsahu ako sa určil v rámci autentifikácie konkrétneho používateľa. Prístupový

token môže mať rôzne formáty podľa toho ako funguje schéma zabezpečenia rozhrania.

Token je spojený s istými údajmi o používateľovi ako napríklad identifikátor a prístupové práva. Tieto údaje môžu byť uložené priamo v tokene alebo na serveri, v závislosti od toho aký formát tokenu je použitý. Zároveň je s tokenom spojený aj časový limit platnosti tokenu, ktorý býva kvôli bezpečnosti krátky. Po odchytení tokenu totiž môže útočník vystupovať v mene používateľa, ktorého údaje sú uložené v tokene.

1.6.2 Nositeľský token

Nositeľský token reprezentuje najčastejší spôsob použitia prístupového tokenu. Pri jeho použití má nositeľ tokenu prístup k rozhraniu bez ohľadu nato, kto je. Stačí aby v požiadavke poslal platný nositeľský token.

Následne rozhranie nekontroluje identitu používateľa, ale iba platnosť tokenu. Preto pri jeho využití treba prikladať väčší dôraz na bezpečnosť tokenu napríklad tak, že jeho platnosť bude veľmi krátka, napríklad 5 minút.

1.6.3 Obnovovací token

Ako sme už spomínali vyššie má prístupový token často relatívne krátky časový limit platnosti. Ak chce používateľ využívať rozhranie aj po jeho uplynutí, je potrebné aby získal nový prístupový token.

Na riešenie tohto problému existujú dva jednoduché spôsoby. Buď sa používateľ znova autentifikuje a server mu vygeneruje nový prístupový token alebo si klient, cez ktorého používateľ komunikuje s rozhraním bude pamätať prihlasovacie údaje používateľa a využije ich na znovuautentifikáciu používateľa. Oba prístupy nie sú ideálne, prvá možnosť je nekomfortná pre používateľa a nie je možná ak je klientom nejaký servis alebo iná aplikácia. Druhá možnosť je nebezpečná, lebo klient musí mať niekde uložené prihlasovacie údaje používateľa a tým sa zvyšuje riziko ich získania útočníkom.

Obnovovací token adresuje problém s krátkou platnosťou prístupových tokenov. Pri prvotnej autentifikácii používateľa server okrem prístupového tokenu vygeneruje aj obnovovací token. Oba tokeny vráti klientovi. Následne keď uplynie platnosť prístupového tokenu, klient pošle požiadavku o nový prístupový token pomocou obnovovacieho tokenu.

Schému s obnovovacími tokenmi využíva napríklad populárny autentifikačný protokol OAuth 2.0 [4].

1.6.4 Identifikačný token

Identifikačný token je špeciálny typ prístupového tokenu. V tomto prípade musí ísť o štruktúrovaný formát tokenu. Token obsahuje identifikátor používateľa a autorizačné

údaje. Navyše môže token obsahovať doplnkové údaje ako vydavateľa tokenu, čas platnosti tokenu, čas vydania tokenu a podobne.

Identifikačný token využíva napríklad protokol OpenID Connect (OIDC), čo je nadstavba protokolu OAuth 2.0 o identitu používateľa [13]. OIDC dokonca presne špecifikuje použitie JWT (JSON Web Token), ktorý viac predstavíme v kapitole 2.

1.7 Formáty tokenov

Už sme spomenuli, že existujú rôzne typy tokenov, no iba pri identifikačnom tokene je striktne daný formát tokenu. Pri ostatných typoch sa stretávame s rôznymi formátmi tokenu. Vo všeobecnosti rozoznávame dva formáty tokenov a to nepriehľadné (angl. opaque) a štruktúrované. Existujú aj hybridné formáty, ktoré kombinujú výhody obidvoch spomínaných typov, no pri nich sa dá hovoriť skôr o istých vzoroch ako o formátoch. Najpoužívanejšie sú fantómové a rozdelené (angl. split) tokeny.

1.7.1 Nepriehľadný token

Ide o najjednoduchší formát tokenu. Nepriehľadný token je náhodný reťazec znakov. Jednoduchý je v tom, že nenesie žiadnu pridanú informáciu pre rozhranie. Všetky metadáta o tokene si musí pamätať rozhranie.

Takýto prístup so sebou nesie významné výhody aj nevýhody. Výhodou je, že samotné vytvorenie tokenu je veľmi rýchle, rozhranie jednoducho vygeneruje náhodný reťazec. Navyše vďaka tomu, že nenesie žiadne pridané informácie, nenesie ani citlivé informácie o používateľovi, teda jeho zachytenie útočníkom nie je také nebezpečné. Nevýhodou však je, že rozhranie ho nevie bezstavovo overiť, teda ho musí napríklad vyhľadať v databáze platných tokenov a to je časovo náročné.

1.7.2 Štruktúrovaný token

Na rozdiel od nepriehľadného tokenu štruktúrovaný token obsahuje pridanú informáciu napríklad identifikáciu používateľa, jeho prístupové práva, čas platnosti tokenu, čas vydania tokenu a podobne.

Aby sa predišlo úniku citlivých informácií z tokenu, je token šifrovaný alebo zahešovaný. Využívajú sa rôzne symetrické aj asymetrické algoritmy. Všetky konkrétne protokoly, ktoré rozoberáme v tejto práci, používajú štruktúrované tokeny a v kapitole 2 sa venujeme ich vytváraniu a s ním spojenému šifrovaniu alebo hešovaniu tokenov.

Najväčšou výhodou štruktúrovaného tokenu je, že rozhranie ho môže bezstavovo overiť, lebo pozná kľúč, ktorým bol token zašifrovaný alebo zahešovaný, teda ho vie

dešifrovať a prečítať informácie, ktoré nesie. Zo získaných informácií vie rozhranie overiť platnosť tokenu a autorizovať používateľa.

1.7.3 Fantómový token

Ako sme naznačili v úvode podkapitoly, fantómový token je hybridný formát tokenu. Podľa tohto vzoru autentifikačný server vygeneruje dva tokeny, nepriehľadný token pre klienta a štruktúrovaný token pre rozhranie. Ďalej sa využíva API brána alebo reverzný proxy server (RPS), v ktorom sa uloží dvojica tokenov ako kľúč a hodnota. Kľúčom je nepriehľadný token, hodnotou je štruktúrovaný token.

Následne, keď klient pošle požiadavku na rozhranie, tak pridá nepriehľadný token. API brána alebo RPS ho preloží na štruktúrovaný token a tento štruktúrovaný token poskytne rozhraniu. Rozhranie ho môže bezstavovo overiť a využiť pridané informácie, ktoré nesie.

Brána alebo RPS síce musí vyhľadať záznam s dvojicou tokenov, kde je kľúčom poslaný nepriehľadný token, ale môže si výsledok uložiť do vyrovnávacej pamäte pre ďalšie požiadavky. Tým sa zvýši priepustnosť brány alebo RPS a zároveň ubudnú nároky na výkonnosť rozhrania.

Výhodou oproti obyčajnému štruktúrovanému tokenu je, že klient, teda prehliadač alebo iná aplikácia nedržia v pamäti štruktúrovaný token obsahujúci, aj keď zašifrované, citlivé informácie.

1.7.4 Rozdelený token

Rozdelený token má podobnú schému a výhodu ako fantómový token, no navyše ombedzuje štruktúrovaný token vydaný autentifikačným serverom tým, že musí obsahovať podpis chrániaci jeho autenticitu.

V schéme rozdeleného tokenu vydá autentifikačný server len jeden štruktúrovaný token. Podpis z tohto tokenu pošle klientovi a do vyrovnávacej pamäte brány alebo RPS zapíše celý token so zahešovaným podpisom ako kľúčom.

Požiadavky od klienta budú obsahovať ako token získaný podpis a brána alebo RPS ho zahešuje a preloží na štruktúrovaný token pomocou vyrovnávacej pamäte. Token následne spolu s požiadavkou pošle rozhraniu.

Kapitola 2

Šprecifikácia konkrétnych API tokenov

V tejto kapitole predstavíme v praxi využívané API tokeny, ktorými sa zaoberá naša práca. Uvedieme ich formát a základané charakteristiky ako ich vytvorenie, či validácia.

2.1 JSON Web Token

Prvý token, ktorým sa budeme zaoberať je JSON web token (JWT) [8]. JWT vznikol ako súčasť JOSE [1] (JSON object signing and encryption) štandardov, čo je dokument vypracovaný pracovnou skupinou IETF (Internet Engineering Task Force) na základe aplikácií bezpečnostných mechanizmov v rámci vývoja softvéru. Tieto štandardy popisujú využitie a definujú požiadavky na objekty formátu JSON, ktoré sú bezpečné.

Definujú štandard pre bezpečný prenos JSON objektov medzi stranami, ktoré sú schopné ich overiť a dešifrovať. Zavádzajú tri základné formáty JSON objektov a to JWS (JSON Web Signature), JWE (JSON Web Encryption) a JWK (JSON Web Key), ktorým sa detailnejšie venujú ďalšie štandardy [7, 9, 6]. Prvé dva sú formáty zabezpečujúce bezpečnostné vlastnosti JSON objektov. Oba zabezpečujú autentickosť a integritu pomocou elektronických podpisov alebo hešovaného autentifikačného kódu (HMAC), ďalej budeme hovoriť v oboch prípadoch o podpise tokenu. JWE navyše zabezpečuje aj dôvernosť a to šifrovaním obsahu JSON objektu. Posledný formát JWK je formát pre reprezentáciu kľúčov, ktoré sú použité v kryptografických algoritmoch využitých v JWS a JWE. Kryptografické algortimy a ich identifikátory sú definované JSON Web Algorithms (JWA) štandardom [5].

2.1.1 Formát a vytvorenie JWT

Samotný JWT je vpodstate iba serializácia JSON objektu chráneného JWS alebo JWE. Podľa štandardu JWT obsahuje tri samostatné časti oddelené bodkami - hlavičku, telo a podpis. Hlavička a telo sú serializované JSON objekty obsahujúce oprávnenia

(angl. claim) vo forme dvojíc kľúč, hodnota. Niektoré kľúče niektorých oprávnení sú definované v štandarde a teda by sa nemali používať pre žiadne iné hodnoty.

A to konkrétne v hlavičke najdôležitejšie sú *typ* a *alg*. Prvý určuje typ tokenu a druhý algoritmus, ktorý bol použitý na vytvorenie podpisu alebo vrámci šiforvania obsahu tokenu. Rôzne možnosti pre algortimy sú definované v JWA štandardoch.

Telo obsahuje oprávnenia týkajúce sa kontrétnej autentifikácie a používateľa, pre ktorého bol token vydaný. Dôležité Štandardom popísané kľúče sú napríklad *iss*, *sub*, *exp*, *nbf*, *iat*. Popisujú postupne vydavateľa tokenu, identifikátor používateľa, čas vypršania platnosti tokenu, čas, kedy sa token začne považovať za platný a čas vydania tokenu.

Do tela aj hlavičky sa môžu vkladať ľubovoľné iné oprávnenia, naprílad *admin*, *role*, *permissions*, určujúce oprávnenia používateľa.

Hlavička a telo sa serializujú pomocou base64url kódovania [10]. V prípade JWE sa telo ešte telo šifruje. Následne sa obe časti podpíšu algoritmom definovaným v hlavičke a podpis sa zreťazí s hlavičkou a telom. Výsledný reťazec sa používa ako token.

2.1.2 Overenie platnosti JWT

Na overenie platnosti tokenu treba zvalidovať podpis. Podpis je vytvorený pomocou algoritmu definovanom v hlavičke. Teda pri validácii tokenu sa ako prvé dekóduje hlavička tokenu a z nej sa prečíta hodnota v kľúči *alg*.

Na základe hodnoty v kľúči *alg* sa určí algoritmus, ktorý bol použitý na podpis tokenu. Následne sa podpis zvaliduje.

Ak bol token vytvorený pomocou JWE na prečítanie tela je potrebné ho najprv dešifrovať. V prípade využitia JWS je telo po dekódovaní priamo čitateľné.

Kapitola 3

Teoretické porovnanie API tokenov

V tejto kapitole porovnáme teoretické parametre rôznych typov API tokenov ako bezpečnosť, škálovateľnosť, jednoduchosť implementácie a flexibilita.

Kapitola 4

Návrh a implementácia jednoduchého rozhrania

V tejto kapitole navrhneme a implementujeme jednoduché rozhranie s použitím viecerých API tokenov pre autentifikáciu a autorizáciu.

Kapitola 5

Porovnanie na jednoduchom rozhraní

V tejto kapitole porovnáme jednotlivé API tokeny na nami implementovanom rozhraní. Výsledkom budú naše pozorovania pri implementacii a použití rozhrania s jednotlivými typmi API tokenov.

Záver

Literatúra

[1] Richard Barnes. Use Cases and Requirements for JSON Object Signing and Encryption (JOSE). RFC 7165, Apríl 2014.

[2] Fallible. We reverse engineered 16k apps, here's what we found, Január 2017. `https://hackernoon.com/we-reverse-engineered-16k-apps-heres-what-we-found-51bdf3b456bb#.io6e11q6n`.

[3] Weili Han, Zhigong Li, Minyue Ni, Guofei Gu, and Wenyuan Xu. Shadow attacks based on password reuses: A quantitative empirical analysis. *IEEE Transactions on Dependable and Secure Computing*, 15(2):309–320, 2018.

[4] Dick Hardt. The OAuth 2.0 Authorization Framework. RFC 6749, 2012.

[5] Michael Jones. JSON Web Algorithms (JWA). RFC 7518, Máj 2015.

[6] Michael Jones. JSON Web Key (JWK). RFC 7517, Máj 2015.

[7] Michael Jones, John Bradley, and Nat Sakimura. JSON Web Signature (JWS). RFC 7515, Máj 2015.

[8] Michael Jones, John Bradley, and Nat Sakimura. JSON Web Token (JWT). RFC 7519, Máj 2015.

[9] Michael Jones and Joe Hildebrand. JSON Web Encryption (JWE). RFC 7516, Máj 2015.

[10] Simon Josefsson. The Base16, Base32, and Base64 Data Encodings. RFC 4648, Október 2006.

[11] Arvind Narayanan and Vitaly Shmatikov. Fast dictionary attacks on passwords using time-space tradeoff. In *Proceedings of the 12th ACM Conference on Computer and Communications Security*, CCS '05, page 364–372, New York, NY, USA, 2005. Association for Computing Machinery.

[12] Agathoklis Prodromou. Tls security 6: Examples of tls vulnerabilities and attacks, Marec 2019. https://www.acunetix.com/blog/articles/tls-vulnerabilities-attacks-final-part/.

[13] N. Sakimura, NRI, J. Bradley, Ping Identity, Microsoft, M. Jones, B. de Medeiros, Google, and C. Mortimore. Openid connect core 1.0 incorporating errata set 1, November 2014. https://openid.net/specs/openid-connect-core-1_0.html#IDToken.