Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Využití jednorázových hesel OTP

Daniel Odvárko

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Daniel Odvárko

Systémové inženýrství

Název práce

Využití jednorázových hesel OTP

Název anglicky

One-time passwords

Cíle práce

Cílem této práce je vytvořit ukázkovou webovou aplikaci s autentizací pomocí OPT.
Dílčím cílem je s využitím dostupných zdrojů objasnit problematiku využití jednorázových hesel (OTP).

Metodika

- studium odborných informačních zdrojů

- analýza nástrojů použitelných jak pro běžné uživatele, tak pro firemní zabezpečení

- vytvoření praktické ukázky na základě teoretických znalostí – webové aplikace s přihlašováním pracujícím na principu OTP.

- zhodnocení a formulování závěrů

Doporučený rozsah práce

30-40 stran

Klíčová slova

OTP, jednorázová, hesla, zabezpečení, bezpečnost, počítač, síť, internet, web, ochrana

---

Doporučené zdroje informací

DOSTÁLEK, Libor, Marta VOHNOUTOVÁ a Miroslav KNOTEK. Velký průvodce infrastrukturou PKI a technologií elektronického podpisu. 2., aktualiz. vyd. Brno: Computer Press, 2009, 542 s. ISBN 978-80-251-2619-6.

CHESWICK, William R a Steven M BELLOVIN. Firewalls and Internet security: repelling the wily hacker. Reading, Mass.: Addison-Wesley Publishing Company, c1994, xiv, 306 s. Addison-Wesley professional computing series. ISBN 0201633574.

Předběžný termín obhajoby

2015/16 LS – PEF

Vedoucí práce

Ing. Martin Havránek, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 1. 2. 2016

Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 10. 2. 2016

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 01. 03. 2016

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Využití jednorázových hesel OTP" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 11. 3. 2016 _____

Poděkování

Děkuji svému vedoucímu bakalářské práce Ing. Martinu Havránkovi, Ph.D za odborné vedení a cenné rady i připomínky při konzultacích mé bakalářské práce.

Využití jednorázových hesel OTP

One-time passwords

Souhrn

Tato práce pojednává o jednorázových heslech a jejich využití v praxi. Jednorázová hesla jsou, jak již z jejich názvu vyplývá, hesla, která lze využít pouze jednou, čili když se například někdo zmocní našeho hesla, je již neplatné. Teoretická část práce se zaměřuje jak na běžná hesla, způsoby zabezpečení a napadnutelnost, tak na OTP, jejich jednotlivé druhy a způsoby zasílání. Praktická část sestává z tvorby praktické ukázky na základě nabytých teoretických znalostí - webové aplikace s přihlašováním pracujícím na principu OTP

Summary

This project deals with one-time passwords and their practical usage. One-time passwords are, as their name implies, passwords, which can be used only one time, so, when someone takes over our password, it's already invalid. Theoretical part of this project focuses on both common passwords, security methods and vulnerability to attackers, and OTP, their individual kinds and types of delivery. Practical part consists of creation of practical demo based on acquired knowledge – web application with authentication based on OTP.

Klíčová slova: OTP, jednorázová, hesla, zabezpečení, bezpečnost, internet, web, ochrana

Keywords: OTP, one-time, passwords, security, safety, internet, web, protection

Obsah

1 Úvod

Hesla zde byla v různých podobách již od nepaměti a jsou nedílnou součástí moderního života, ani kontrolu e-mailu, ani převod peněz v internetovém bankovnictví již nelze provést bez autorizace. Jejich hlavním účelem je ověření identity, mají ovšem mnoho slabin, a tak lze uživatelovu identitu snadno ukrást prostřednictvím zneužití hesla. Mnoho lidí ale očividně sílu svého hesla zanedbává, vždyť v žebříčku nejpopulárnějších hesel vede již několik let po sobě prosté „123456“ či „password“. Mnohdy ani mít silné heslo v dnešní době nezabezpečených wifi sítí a stále připojených chytrých telefonů nestačí. Možným řešením tohoto problému jsou jednorázová hesla, která snižují riziko odposlechnutelnosti hesla a zvyšují bezpečnost uživatelů, těmi se zabývá tato práce.
(Condliffe, 2014)

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem této práce je s pomocí dostupných zdrojů analyzovat možnosti využití jednorázových hesel, cílem dílčím pak tvorba praktické ukázky autentizace jednorázovým heslem. Praktická část bude realizována úpravou stávajícího přihlašování a jeho zabezpěčení technologií OTP.

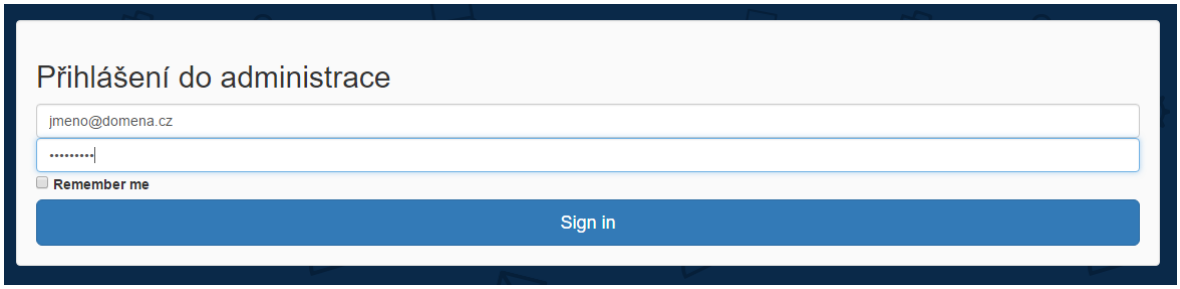
2.2 Metodika

Práce se bude skládat ze dvou částí, analytické, popisující základní principy fungování běžných hesel a rozebírající jednotlivé druhy hesel jednorázových, spolu s jejich definicemi a základními požadavky na ně.
V praktické části bude představena PHP třída multiOTP a implementováno přihlašování jednorázovým heslem za pomocí této třídy do stávající administrace webového portálu. Dále budou popsány jednotlivé části serverové a klientské strany, zhodnoceny možnosti dalšího vylepšení a zabezpečení tohoto přihlašování a na jejich základě zformulován závěr.

3 Teoretická část

3.1 Hesla

Funkcionalitu běžných hesel je možno demonstrovat na přihlašování do webové aplikace, například do administrace nějakého portálu.



Obr. 1 - Přihlášení, zdroj: autor

Přihlášení probíhá uživatelovým zadáním jeho přihlašovacích údajů, uživatelského jména a hesla. Na serveru není uložena textová podoba hesla, pouze jeho hash, neboli jeho jednocestná funkce.

955db0b81ef1989b4a4dfeae8061a9a6

Obr. 2 - Ukázka md5 hashe, zdroj: autor

Když chce uživatel provést autentizaci, jím zadané heslo je zahashováno a porovnáváno s hashem hesla na serveru, například v databázi SQL. Kromě rizika odposlechnutí hesla je zde i jistá pravděpodobnost, že se někdo dostane do naší databáze a povede se mu heslo rozklíčovat, jelikož k tomu je na internetu dostupná řada nástrojů. Tento způsob zabezpečení je jednoduchý a hojně využívaný, s bezpečností ale na tom není nejlépe. (Dostálek, Vohnoutová, Knotek, 2009)

Standardní hesla jsou typicky statická, běžně se nemění, dokud není uživatel vyzván k tvorbě hesla nového. A jelikož se příliš často nemění, dává to útočníkům velmi dlouhý časový úsek, během kterého mají příležitost heslo prolomit a zneužít. Trendem je přecházet ze statických hesel na hesla dynamická, která se pravidelně mění, tato jsou známa jako jednorázová hesla, nebo OTP. Systémy pracující s OTP vytváří unikátní heslo, které není znovu použitelné. Útočníkovi, který toto heslo ukradne, bude již k ničemu.
(Ciampa, 2011)

3.2 Jednorázová hesla - OTP

Jednorázové heslo je automaticky generovaný číselný, nebo alfanumerický řetězec znaků, který autentizuje uživatele pro určitou činnost nebo relaci. Jednorázové heslo je silnější, než heslo stálé (zvláště to vytvořené uživatelem) a může být použito místo normální uživatelské autentizace, nebo jako její doplněk a přidání další ochranné vrstvy. Autentizace, kombinující klasické údaje a jednorázové heslo, či další faktory, se nazývá *vícefaktorová autentizace*. (Rouse, 2013)

Jednorázová hesla řeší problém odposlechu hesla během jeho přenosu sítí a následným použitím odposlechnutého hesla. Pokud je jednorázové heslo využito pouze pro počáteční autentizaci, pak ještě po úspěšně proběhlé autentizaci existuje nebezpečí v převzetí relace útočníkem. Proto se jednorázová hesla zpravidla ještě následně využívají pro další zabezpečení relace (např. pomocí doplňování MAC k blokům přenášených dat či jako součást symetrických klíčů pro šifrování relace).
(Dostálek, Vohnoutová, Knotek, 2009)

Na rozdíl od běžného hesla, fungujícího na principu „něco, co víte“ (a pamatujete si), jednorázová hesla fungují na principu „něco, co máte“.
(Woodford, 2015)

Bezpečnost systému OTP je založena na neinverzovatelnosti bezpečné hashovací funkce. Takovou funkci musí být možno spočítat v dopředném směru, a výpočetně neproveditelná ve směru inverzním.
(A One-Time Password System, 1998)

3.2.1 Využití OTP

Jednorázová hesla dnes nachází širokou škálu využití: SMS klíč při autorizaci platby kartou, vygenerované heslo v aplikaci Battle.net autenticator, nebo přes USB token pro přístup k hernímu účtu Battle.net, k přihlašování na Google účet apod.

3.3 Rozdělení OTP

OTP lze rozdělit do několika kategorií – OTP pracující na principu matematického algoritmu (na výzvě), HOTP (HMAC-based one-time password), a z něho vycházející TOTP (Time-based one-time password).

3.3.1 OTP založené na výzvě

Uživatel si na začátku vytvoří, nebo mu je přidělen řetězec znaků, neboli seed, a je určen počet použití daného hesla. Tento seed *(s)* se poté zahashuje pomocí funkce *f(s)*, a to tolikrát, kolikrát má být heslo použito *(n)*. Poslední zahashování *fn(s)* bude využito jako první heslo. Výhoda tohoto druhu jednorázových hesel je, že i v případě odposlechnutí hesla útočníkem je výpočet inverzní funkce velmi obtížný a téměř nemožný.
(Iacona, 2009)

Seed musí být složen z 1 až 16 alfanumerických znaků. Je to řetězec znaků, který nesmí obsahovat žádné mezery a měl by být složen pouze z alfanumerických znaků dle ISO-646. Všechna velká písmena musí být vnitřně převedana na malá před zpracováním. Pořadové číslo a seed tvoří dohromady větší celek nazývaný challenge (výzva). Výzva pošle generátoru parametry, které potřebuje ke spočítání správného jednorázového hesla. Výzva musí být ve standardní syntaxi, aby ji mohly automatizované generátory rozpoznat a parametry z ní získat. Syntaxe výzvy je:

otp - <identifikátor algoritmu> <integer pořadového čísla> <seed>

Tyto tři tokeny musí být odděleny mezerou (jakýkoli počet mezer, nebo tabulátorů), a celý řetězec výzvy musí být oddělen mezerou, nebo odřádkován. Před zpracováním je řetězec převeden na malá písmena. Momentálně definované identifikátory algoritmů jsou *md4*, *md5* a *sha1*. Příkladem výzvy je *otp-md5 487 dog2*
(A One-Time Password System, 1998)

Jednorázové heslo vygenerované tímto postupem má délku 64 bitů. Zadávání takto dlouhého hesla je složité a náchylné na chyby. Proto může být převedeno na sekvenci šesti krátkých

(od 1 do 4 písmene) jednoduchých slov, které používají pouze znaky z normy ISO-646. Každé slovo je vybíráno ze slovníku obsahujícího 2048 slov, všechna jednorázová hesla tak mohou být kódována při 11 bitech na slovo.
(A One-Time Password System, 1998)

Dva přebývající bity v kódování jsou využity k uchování kontrolního součtu. 64 bitů klíče je rozděleno do párů po bitech a tyto páry jsou sečteny dohromady. Dva nejméně významné bity tohoto součtu jsou zakódovány v posledních dvou bitech šestislovné sekvence s tím nejméně významným bitem součtu jako posledním zakódovaným bitem. Všechny generátory jednorázových hesel musí být schopny tento kontrolní součet ověřit jako součást operace dekódování jednorázového hesla.
(A One-Time Password System, 1998)

Generátory, tvořící šestislovný formát musí tato slova zobrazovat s velkými písmeny oddělenými vždy jednou mezerou. Všechny servery musí tento šestislovný formář přijmout bez ohledu na velikost písmen a počet mezer a musí využívat stejný slovník.
(A One-Time Password System, 1998)

K usnadnění implementace menších generátorů je hexadecimální výstup také přijatelnou alternativou reprezentace jednorázového hesla, všechny servery musí umět přijímat jak šestislovnou interpretaci, tak hexadecimální formát. U hexadecimálního formátu je vyžadováno, aby servery ignorovaly všechny mezery. Příkladem hexadecimálního formátu je *3503785b369cda8b*.
(A One-Time Password System, 1998)

Aplikace na serveru, vyžadující ověření vyšle výzvu. Na základě parametrů z tého výzvy a tajného hesla spočítá (nebo vyhledá) generátor jednorázové heslo, které je poté zasláno na server k ověření. Systém na serveru disponuje databází obsahující pro každého uživatele jednorázové heslo z poslední úspěšné autentizace, nebo první jednorázové heslo z nové sekvence. Aby ověřil uživatele, server dekóduje obdržené jednorázové heslo do formy 64 bitového klíče a ten poté zahashuje pomocí bezpečnostní hashovací funkce. Pokud je

výsledek této operace shodný s uloženým předchozím OTP, autentizace je úspěšná a přijaté jednorázové heslo je uchováno pro budocí využití.
(A One-Time Password System, 1998)

Protože je počet aplikací hashovací funkce omezený, a každým použitím generátoru, tedy každou autentizací se zmenšuje o jednotku, na konci „životnosti“ hashe musí být uživateli vytvořen nový seed.

3.3.2 HOTP

HOTP, neboli HMAC based one-time password je druh jednorázového hesla, založený na dvou hlavních věcech, sdíleném tajemství, a pohybujícím se faktoru (počítadlu). Při každém generování hesla se počítadlo zvýší o jedna a výsledné heslo bude tedy pokaždé jiné. (One-Time Passwords – HOTP and TOTP, 2014)

Tento algoritmus je založen na zvyšující se hodnotě počítadla a statickém symetrickém klíči, známému pouze validačnímu tokenu a validační službě. K vytvoření hodnoty klíče HOTP je využit algoritmus HMAC-SHA-1.
(HOTP: An HMAC-Based One-Time Password Algorithm, 1998)

Tento algoritmus vyžduje kryptografickou hashovací funkci, označenou *H,* a tajný klíč *K.* Předpokládá se, že *H* je funkcí, kde data jsou hashována iterací základní kompresní funkce na blocích dat. Písmenem *B* se označuje délka takových bloků v bytech (u běžně používaných hashovacích funkcí – MD5, SHA-1 je *B=64*) a písmenem *L* délka výstupů hashe (*L=16* pro MD5, *L=20* pro SHA-1). Platí, že $K \leq B$. Aplikace využívající klíč delší, než *B* zahashují nejprve klíč pomocí *H*, a až poté využívají výsledný řetězec *L* jako tajný klíč. Ve všech případech je doporučena minimální délka *K,* a to *L* bytů.
Jsou definovány 2 fixní a rozdílné řetězce ipad a opad (i jako inner – vnitřni, o jako outer – vnější)
(HMAC: Keyed-Hashing for Message Authentication, 1997)

Ipad = byte 0x36 opakovaný B-krát
Opad = byte 0x5C opakovaný B-krát.

Ke spočítání HMAC na řetěžci ‘text’, je tedy provedena funkce:

H(K XOR opad, H(K XOR ipad, text))

Přesný postup:

1) Přidání nul na konec *K*, vznikne řetězec o *B* bytech
2) Exkluzivní disjunkce (XOR) řetězce *B* (spočítaného v kroku 1) a *ipad*
3) Připojení dat *‘text’* k výslednému řetězci *B* z kroku 2
4) Aplikace funkce H na data generovaná ve 3. kroku
5) Exkluzivní disjunkce (XOR) řetězce *B* (spočítaného v kroku) 1 a *opad*
6) Připojení výsledku *H* z kroku *4* k výslednému řetězci *B* z kroku 5
7) Aplikace H na data generovaná v 6. roku. Výstupem je výsledek

(HMAC: Keyed-Hashing for Message Authentication, 1997)

Výstupem tohoto algoritmu je 160 bitů, tato hodnota musí být tedy osekána, aby mohla být snadno zadána uživatelem.
(HOTP: An HMAC-Based One-Time Password Algorithm, 1998)

Operace generování hodnoty HOTP může být popsána ve 3 krocích:

1) Generování hodnoty HMAC-SHA-1,

HS = HMAC-SHA1(K,C)

výsledkem je 20-bytový řetězec

2) Z výsledné hodnoty z 1. kroku je pomocí funkce DT (Dynamic Truncation)

S = DT(HS)

vyříznut řetězec o délce 4 byty (začátek 4 bytů je vybrán podle nejméně významného bitu ve 160bitů dlouhém řetězci), výsledkem je 31-bitový řetězec.

3) Je spočítána hodnota HOTP

HOTP = StToNum(S) % 10^X

Řetězec je převeden na číselnou řadu a je spočítáno modulo, kde *X* záleží na tom, kolikamístné má být OTP, například pokud má být jednorázové heslo 6místné, za X se dosadí 6.

(HOTP: An HMAC-Based One-Time Password Algorithm, 1998)

3.3.3 TOTP

K realizaci autentizace pomocí TOTP je nezbytné, aby na obou stranách, jak na straně klienta, tak na straně serveru, synchronizovaný čas. TOTP jsou totiž generovány hashem aktuálního času a tajného kódu, který je sdílený klientem i serverem. Je samozřejmé, že při prvním využití této metody je čas synchronní, ovšem časem může dojít ke značnému posunu, pokud k tomuto dojde, server klientem zadané heslo porovná s několika vypočítanými hesly dopředu i dozadu a pokud dojde ke shodě s některým s daných, klientovi je umožněn přístup a na serveru se uloží údaje o časovém posunu. Platnost takovýchto hesel je většinou 30 vteřin, výjimkou ale nejsou ani hesla s minutovou platností.

TOTP je varianta protokolu HOTP, založená na čase, platí, že:

TOTP = HOTP(K, T)

kde *T* je integer a reprezentuje počet časových kroků, které uběhly od počátečního času *T0* k momentálnímu Unixovému času, respektive

$$T = \frac{Momentální\ Unixový\ čas - T0}{X},$$

kde X reprezentuje časový krok v sekundách (výchozí hodnota je 30 sekund)
(TOTP: Time-Based One-Time Password Algorithm, 2011)

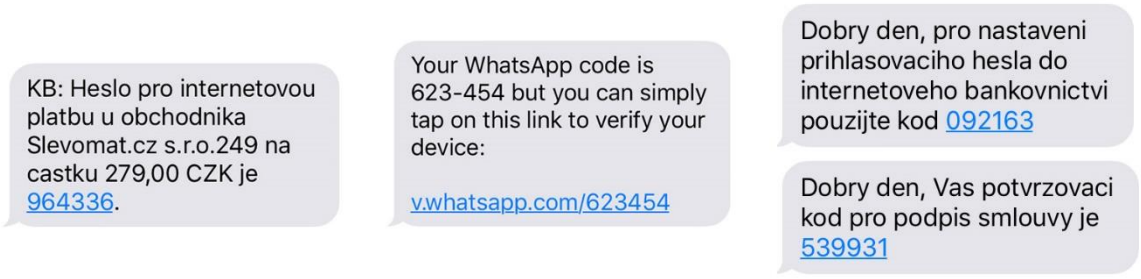
Jednorázové heslo generované během stejného časovém kroku budou stejná. Jakmile validační systém obdrží OTP, neví přesný čas, kdy klient vygeneroval OTP. Pro porovnání je používán čas, kdy je OTP serverem obdrženo k porovnání. Z důvodu zpoždění sítě může být velký rozdíl mezi časem, kdy bylo OTP generováno a kdy je validováno. Pokud je zadáno například heslo vygenerované vteřiny před koncem časového kroku, je toto pak porovnáváno s heslem vygenerovaným v dalším časovém kroku. Validační systém by měl mít nastavenou toleranci zpozdění pro validaci a heslo by mělo být porovnáváno nejen s heslem v čase validace, ale také s hesly předchozími v rámci přenosového zpoždění. Doporučuje se povolit jeden časový krok jako zpoždění sítě.
(TOTP: Time-Based One-Time Password Algorithm, 2011)

Časový krok má dopad na bezpečnost a použitelnost. Větší časový krok znamená větší čas pro validační systém na ověření OTP, což má své nevýhody. Při velkém časovém kroku je zde větší prostor pro útok, když se třetí strana zmocní hesla ihned po jeho vygenerování, může se snadno autentizovat za nás. Doporučenou délkou časového kroku, kompromisem mezi použitelností a bezpečností, je 30 vteřin.
(TOTP: Time-Based One-Time Password Algorithm, 2011)

Při přihlášení uživatele pomocí TOTP musí být pro další přihlášení vygenerovaný nový klíč v novém časovém kroku, uživatel musí počkat na vypršení času u stávajícího kroku. Validační systém nesmí po úspěšném přihlášení přijmout druhý pokus se stejným OTP.
(TOTP: Time-Based One-Time Password Algorithm, 2011)

3.3.4 Způsoby doručení OTP

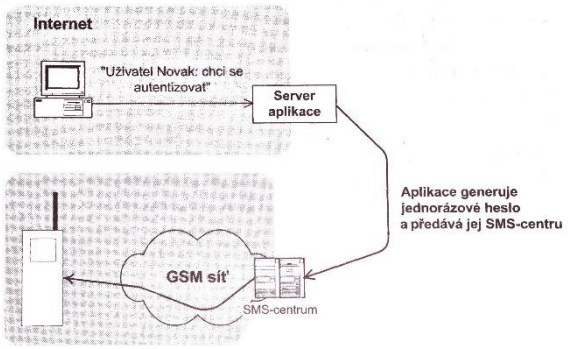
Jednorázová hesla jsou doručitelná mnoha způsoby, i dopis ve schránce obsahující aktivační kód je svým způsobem doručené jednorázové heslo s účelem autentizovat uživatele. Nejrozšířenějším způsobem zasílání hesel je SMS kanál, a to hlavně díky využití v bankovním sektoru.



Obr. 3 – Ukázka jednorázových hesel doručených přes SMS kanál, zdroj: autor

Principem této metody je server, který vygeneruje dotaz a doručí jej uživateli nezávislým kanálem (v tomto případě SMS-zprávou). Druhým kanálem ale může být i fax, e-mail apod. Každý z použitých kanálů přitom nemusí být příliš bezpečný, útočník by musel prolomit dva na sobě nezávislé komunikační kanály současně, což je velice těžko uskutečnitelné. Tento princip se také označuje jako princip dvou zámků.
(Dostálek, Vohnoutová, Knotek, 2009)

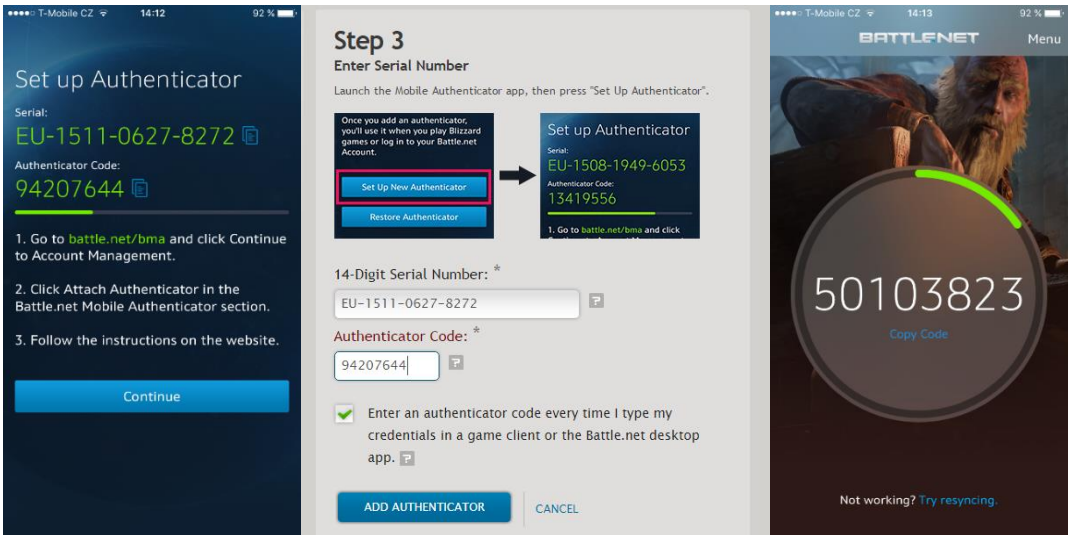
V okamžiku, kdy se má klient autentizovat, oznámí aplikacei své přihlašovací jméno. Aplikace vygeneruje jednorázové heslo a v databázi uživatelů najde číslo jeho mobilního telefonu, na které pomocí SMS-zprávy jednorázové heslo zašle.
(Dostálek, Vohnoutová, Knotek, 2009)



Obr. 4 – Jednorázové heslo zaslané SMS-zprávou – (Dostálek, Vohnoutová, Knotek, 2009)

Dalším, stále se rozšiřujícím způsobem doručení jednorázového hesla ke klientovi je mobilní aplikace, existuje celá řada mobilních aplikací, například aplikace s názvem OTP Auth, umožňující správu více jednorázových klíču. Lze se také setkat s konkrétními aplikacemi daných společností. Například klienti ČSOB mohou použít pro autorizaci svých plateb aplikaci s názvem ČSOB Smart klíč, hráči online her od společnosti Blizzard Entertainment

pak zase Battle.net Mobile Authenticator. Aplikace vygeneruje tajný klíč, který uživatel zadá do webového rozhraní správy svého Battle.net účtu spolu s prvním vygenerovaným heslem z aplikace.



Obr. 5 - Nastavení jednorázového hesla s aplikací Battle.net authenticator, zdroj: autor, https://eu.battle.net/

Po nastavení aplikace je uživatelovo přihlášení dvoufaktorově zabezpečeno a každý přístup do jeho účtu požaduje heslo a po jeho zadání dále i vygenerované jednorázové heslo.

Jednorázové heslo lze použít také k ověření identity fyzické, popřípadě právnické osoby. Při registraci firmy na Google je třeba vyplnit její adresu, na tu je poté zaslán tajný kód, tím se ověří, zda je zadaná adresa opravdu sídlem firmy. Při tvorbě online identity na webu Mojeid.cz, je uživateli opět zaslána obálka s ověřovacím kódem, který je také jednorázovým heslem.

Relativně bezpečným, ale nákladnějším řešením jsou osobní tokeny, mezi jejichž nevýhody patří zejména velké riziko ztráty a u tokenů pouze zavěšených na klíčích, či na platebních kartách také omezenou kapacitu baterie.

4 Praktická část

Praktická část je ukázkou využití jednorázových hesel OTP v praxi, bylo upraveno existující přihlášení do administrace webového portálu, a jeho následné dvoufaktorové zabezpečení. Bylo využito dostupných technologií, konkrétně open-source řešení MultiOTP.

4.1 Multiotp

MultiOTP je PHP třída vyvíjená společností SysCo systèmes de communication sa, pro účely poskytnutí relativně bezplatného a snadného řešení pro implementaci dvoufaktorové autentizace, nezávislého na operačním systému.

K dispozici je také jako spustitelná aplikace pro Windows, či jako zdrojový kód připravený ke kompilaci pro Linux nebo minipočítač Rapsbery Pi. Umožňuje vytvářet jednorázová hesla pomocí všech běžných standardů a doručovat je řadou metod, včetně SMS a e-mailové zprávy. K zabezpečení tohoto portálu, kterým se zabývá tato práce, byla však pro svou technologickou a finanční nenáročnost využita pouze základní funkcionalita a na klientské straně generováno heslo dle standardu RFC 6238 – TOTP aplikací na chytrém telefonu a na serverové straně ověřováno touto PHP třídou.

4.1.1 Popis třídy Multiotp

Nejdříve je potřeba nadefinovat nastavení, která budou využita pro tuto implementaci. Jednou z jeho nejdůležitějších částí konfigurace je řádek:

```
'backend_type'              => "varchar(255) DEFAULT 'files'",
```

Backend_type slouží k nastavení typu backendu, možnými hodnotami jsou ‘files’ a ‘mysql‘, určuje, zda budou ukládány informace o jednotlivých uživatelích ve speciálních souborech s koncovkou .db, nebo v MySQL databázi. Z důvodu většího zabezpečení dat o uživatelích byla vybrána varianta MySQL, ke kterému byly v následujících řádcích konfiturace nastaveny přístupy:

```
'sql_server'              => "varchar(255) DEFAULT "",
'sql_username'             => "varchar(255) DEFAULT "",
'sql_password'             => "varchar(255) DEFAULT "",
'sql_database'             => "varchar(255) DEFAULT "",
'sql_config_table'         => "varchar(255) DEFAULT 'multiotp_config'",
'sql_cache_table'          => "varchar(255) DEFAULT 'multiotp_cache'",
'sql_devices_table'        => "varchar(255) DEFAULT 'multiotp_devices'",
```

```
'sql_groups_table'          => "varchar(255) DEFAULT 'multiotp_groups'",
'sql_log_table'             => "varchar(255) DEFAULT 'multiotp_log'",
'sql_tokens_table'          => "varchar(255) DEFAULT 'multiotp_tokens'",
'sql_users_table'           => "varchar(255) DEFAULT 'multiotp_users'",
```

Nejdříve byly nastavoveny údaje o adrese SQL serveru, přístupové údaje k němu, a název databáze. Ta musí být předem vytvořena, vytvářet jednotlivé tabulky není potřeba, o to se aplikace postará, jejich názvy jsou také nastavitelné, a vyplývají z dalších řádků konfiguračního souboru. Pokud je zvolen backend_type mysql, je konfigurace dále ukládána do tabulky na serveru dle sql_config_table, v případě backend_type files, je vytvořena složka config, a v ní konfigurační soubor, obsahující tato data.

V každém procesu je nejprve volána funkce Multiotp s řetězcem, sloužícím jako klíč pro šifrování uložených dat. Výchozím je MyPersonalEncryptionKey.

```
$multiotp = new Multiotp('MyPersonalEncryptionKey')
```

Poté je nastaven uživatel, se kterým bude v daném procesu pracováno, pokud je tento uživatel neexistující, je v tomto procesu vytvořen a na jeho konci jsou jeho data zapsána do databáze, v opačném případě jsou jeho data z databáze načtena a je s nimi pracováno.

```
$multiotp->SetUser('username');
```

Při ověřování je volána funkce CheckToken() s jedním parametrem, uživatelem zadaným jednorázovým heslem, které je porovnáváno s heslem vygenerovaným serverem na základě zvoleného typu algoritmu a tajného klíče (seedu). Při volání se dvěma parametry oddělenými čárkou provede aplikace synchronizaci jednorázových hesel zvoleného uživatele s přesností na sekundy. Návratovými hodnotami funkce CheckToken() jsou 0 při úspěšném přihlášení, 0 nebo 14 při úspěšné synchronizaci a hodnoty větší, než 20 pro chybové hlášky, například 26 při pokusu o autentizaci již použitým tokenem, či 27 při selhání synchronizace.

```
$multiotp->CheckToken('token1');
$multiotp->CheckToken('token1', 'token2');
```

Vytvoření uživatele probíhá nastavením uživatelského jména, algoritmu, tajného seedu, počtu znaků tokenu, časové platnosti tokenu a zapsáním uživatelských dat do databáze. Pokud zadaný uživatel již existuje, jsou tímto procesem jeho data přepsána nově zadanými.

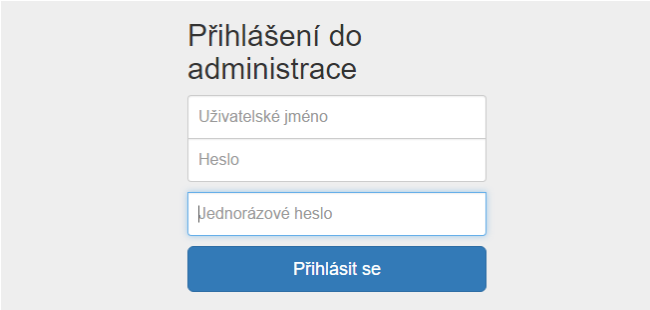
```
$multiotp->SetUser($username);
$multiotp->SetUserAlgorithm('TOTP');
$multiotp->SetUserTokenSeed($seed);
$multiotp->SetUserTokenNumberOfDigits($znaky);
$multiotp->SetUserTokenTimeInterval($sec);
$multiotp->WriteUserData();
```

4.2 Portál (část server)

Cleverjobs portál, dostupný na adrese http://www.cleverjobs.cz/, byl uveden do provozu v září 2015 s cílem pomoci lidem nalézt lepší práci a firmám lepší zaměstnance. K jeho tvorbě byly použity technologie HTML, CSS, PHP, JavaScript, jQuery a AJAX. Nezbytnou součástí portálu je backendová administrační část, komunikující s databází a umožňující přidávání, editaci a mazání jednotlivých pozic a referencí, tyto se poté vypisují na frontend. Reálně tuto část aplikace využívá méně, než 5 osob se stejnými pravomocemi. Toto rozhraní bylo samozřejmě třeba zabezpečit proti neoprávněným přístupům, a využití jednorázového hesla k přihlašování do administrace se, vzhledem k velice malé šanci odposlechnutí hesla útočníkem, a zároveň k snadné použitelnosti uživateli, jevilo jako nejvhodnější řešení.

4.2.1 Přihlašování do portálu

Přihlašování do portálu je realizované běžným formulářem obsahujícím pole pro uživatelské jméno, heslo a jednorázové heslo, odesílaným ajaxem a zpracovávaným v PHP. Proběhne připojení k MySQL serveru, vybrání databáze a porovnávání uživatelského jména a md5 hashe zadaného hesla, s hodnotami uloženými v databázi. Pokud je nalezena shoda - řádek s tímto jménem a heslem, je ve třídě MultiOtp ve funkci SetUser() nastaveno dané uživatelské jméno a funkce CheckToken() porovnává vygenerované jednorázové heslo se zadaným. Při shodě je uživateli nastaven session_login a session_id, při neúspěchu je PHP echem vypsána odpovídající chybová hláška a vykreslena na přihlašovací stránce.



Obr. 6 – Přihlášení do administrace, zdroj: autor

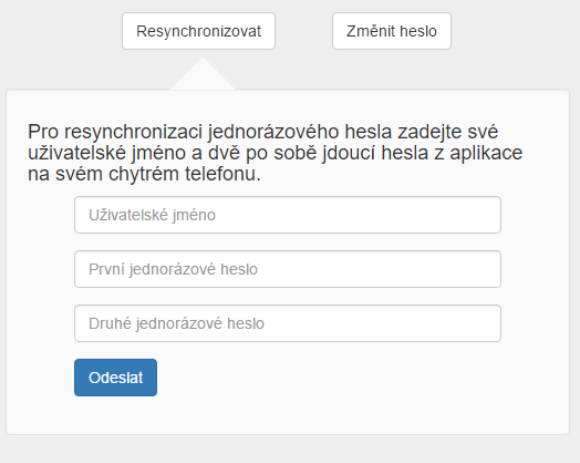
adminlogin.php

```
error_reporting(0);
include "./db.php";
if (!$pripoj)
{
  echo'pripojeni';
  die();
}
$vyberdb = mysql_select_db('cleverjobs', $pripoj);
if (!$vyberdb)
{
  echo'databaze';
  die();
}
$login = mysql_real_escape_string($_POST["inputEmail"]);
$heslo = mysql_real_escape_string($_POST["inputPassword"]);
$md5heslo = md5($heslo);
$dotaz = mysql_query("select * from users where mail = '$login' and pass_hash =
'$md5heslo'");
$overeni = mysql_num_rows($dotaz);
$row = mysql_fetch_array($dotaz);
if($overeni == 1)
 {
require_once('/multiotp/multiotp.class.php');
 $multiotp = new Multiotp('MyPersonalEncryptionKey');
 $multiotp->SetUser($_POST["inputEmail"]);
 if (0 == $multiotp->CheckToken($_POST["inputOTPassword"])) {
    session_start();
    $_SESSION['login'] = $_POST["inputEmail"];
    $_SESSION['id'] = $row["id"];
    echo'true';
   }
   else
   {
    echo'otp';
   }
}
```

```
else
{
  echo'false';
}
Mysql_Close($pripoj);
```

4.2.2 Resynchronizace hesla

Pro demonstraci možností jednorázových hesel byla na přihlašovácí stránku přidána ještě tlačítka pro resynchronizaci jednorázového a změnu stálého hesla. Resynchronizace je opět odesílána ajaxem a zpracovávána PHP, které nejdříve ověří úplnost odeslaných dat a dotazem do databáze zkontroluje existenci uživatele. Třída MultiOtp, konkrétně funkce CheckToken(), je tentokrát volána se dvěma parametry, prvním a druhým jednorázovým heslem. Uživatel je resynchronizován s přesností na vteřiny a o této okolnosi informován. Při neúspěchu je v PHP vypsána a zpět do formuláře vykreslena odpovídající chybová hláška. Tato možnost je k dispozici i v administračním rozhraní po přihlášení, u této varianty odpadá nutnost vyplňování uživatelského jména.



Obr. 7 - Resynchronizace jednorázového hesla, zdroj: autor

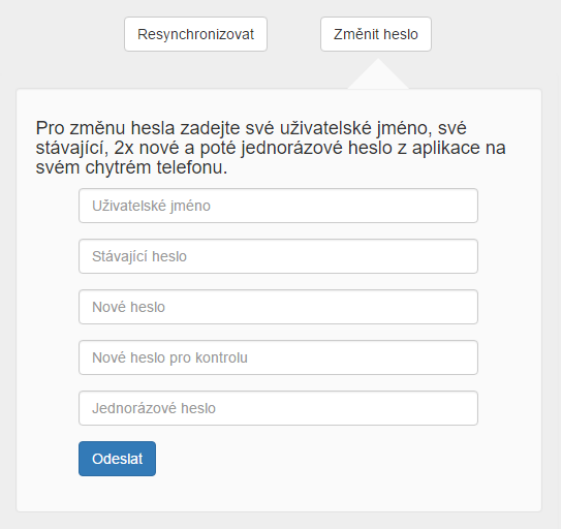
V případě, že se uživatel pomocí TOTP přihlašuje pravidelně, není třeba tak často heslo resynchronizovat, ovšem pro případ, že by přihlášení se správným heslem nefungovalo, je resynchronizace z důvodu časového posunu nutná ke správnému sladění generujících algoritmů a tedy funkci celého přihlašovacího cyklu.

doresync.php

```
error_reporting(0);
include "./db.php";
$resyncusername = ($_POST["resyncusername"]);
$resyncprvni = ($_POST["resyncprvni"]);
$resyncdruhe = ($_POST["resyncdruhe"]);
if (($resyncusername == '') || ($resyncprvni == '') || ($resyncdruhe == ''))
{
echo('chybne odeslani formulare!');
}
else if ((mysql_num_rows(mysql_query("select * from users where mail =
'$resyncusername'"))) != '1')
{
echo('neexistence');
}
else
{
require_once('/multiotp/multiotp.class.php');
$multiotp = new Multiotp('MyPersonalEncryptionKey');
$multiotp->SetUser($resyncusername);
if (0 == $multiotp->CheckToken($resyncprvni , $resyncdruhe))
{
echo('true');
} else {
echo('false');
}
}
Mysql_Close($pripoj);
```

4.2.3 Změna hesla

Další z funkcionalit je možnost změny stálého hesla. Po odeslání ajaxem je nejprve PHP skriptem zkontrolována vyplněnost všech údajů, kontrola původního hesla a uživatelského jména probíhá stejným způsobem jako v případě přihlášení, tedy databázovým dotazem. Dvě nová hesla jsou porovnána a při jejich shodě je volána funkce CheckToken() s jedním parametrem, zadaným jednorázovým heslem. Při úspěchu má uživatel nastavené nové stálé heslo a formulář je vyresetován. Při neúspěchu je v PHP zobrazena a na stránce vykreslena odpovídající chybová hláška.



Obr. 8 – změna stálého hesla, zdroj: autor

dochange.php

```
error_reporting(0);
include "./db.php";
$vyberdb = mysql_select_db('cleverjobs', $pripoj);
  if (!$vyberdb)
  {
    echo'databaze';
  }
else
{
  $changeusername = mysql_real_escape_string($_POST["changeusername"]);
  $changepassword = mysql_real_escape_string($_POST["changepassword"]);
  $changenewpassword = mysql_real_escape_string($_POST["changenewpassword"]);
  $changenewpassword2 = mysql_real_escape_string($_POST["changenewpassword2"]);
  $md5changepassword = md5($changepassword);
  $changeotp= ($_POST["changeotp"]);
  if (($changeusername == '') || ($changepassword == '') || ($changenewpassword == '') ||
($changenewpassword2 == '') || ($changeotp == ''))
  {
  echo('chybne odeslani formulare!');
  }
  else if ((mysql_num_rows(mysql_query("select * from users where mail =
'$changeusername' and pass_hash = '$md5changepassword'"))) != '1')
  {
  echo('neexistence');
  }
  else if ($changenewpassword != $changenewpassword2)
```

```
 {
 echo('neshoda');
 }
 else
 {
 require_once('/multiotp/multiotp.class.php');
 $multiotp = new Multiotp('MyPersonalEncryptionKey');
 $multiotp->SetUser($changeusername);
 if (0 == $multiotp->CheckToken($changeotp))
 {
     $md5changenewpassword = md5($changenewpassword);
     $vloz = mysql_query("UPDATE users SET pass_hash='$md5changenewpassword'
     WHERE mail='$changeusername'");
     if ($vloz == 1)
       {
     echo'true';
       }
     else
       {
      echo'failed';
       }
   }
   else
   {
   echo'otp';
   }
  Mysql_Close($pripoj);
 }
}
```

4.2.4 Přidání nového uživatele

Pravomocí administrátora je mimo správy pozic a referencí také možnost přidávat nové uživatele backendu. Formulář je odesílán ajaxem a zpracováván PHP skriptem. Po ověření úplnosti a správnosti údajů ve formuláři a připojení k MySQL databázi je pomocí databázového dotazu zjištěno, zda již požadované uživatelské jméno neexistuje. Stálé heslo je zahashováno md5 hashem a data vložena do databáze.

Seed je generován náhodně pomocí PHP funkce:

```
bin2hex(openssl_random_pseudo_bytes(10))
```

Tato funkce generuje řetězec náhodných binárních, kryptograficky silných bytů a převádí je do hexadecimální soustavy. Těchto 20 znaků zadává uživatel jako tajný seed do aplikace generující jednorázová hesla. Je volána třída MultiOtp a metoda WriteUserData() zapíše uživatelské jméno, typ algoritmu, seed, počet znaků a dobu trvání do tabulky s uživateli v databázi, připadně do výchozího databázového souboru. Ověření neexistence uživatele v databázi zamezí případnému přepsání uživatele stávajícího, které, při jeho existenci, metoda WriteUserData() provede.

Obr. 9 – Tvorba nového uživatele. zdroj: autor

doadduser.php

```
include "./db.php";
if(!isset($_SESSION['login']))
{
header("Location: index.html");
}
else
{
$username = ($_POST["username"]);
$prvni = ($_POST["prvni"]);
$druhe = mysql_real_escape_string($_POST["druhe"]);
$seed = ($_POST["seed"]);
$sec = ($_POST["sec"]);
```

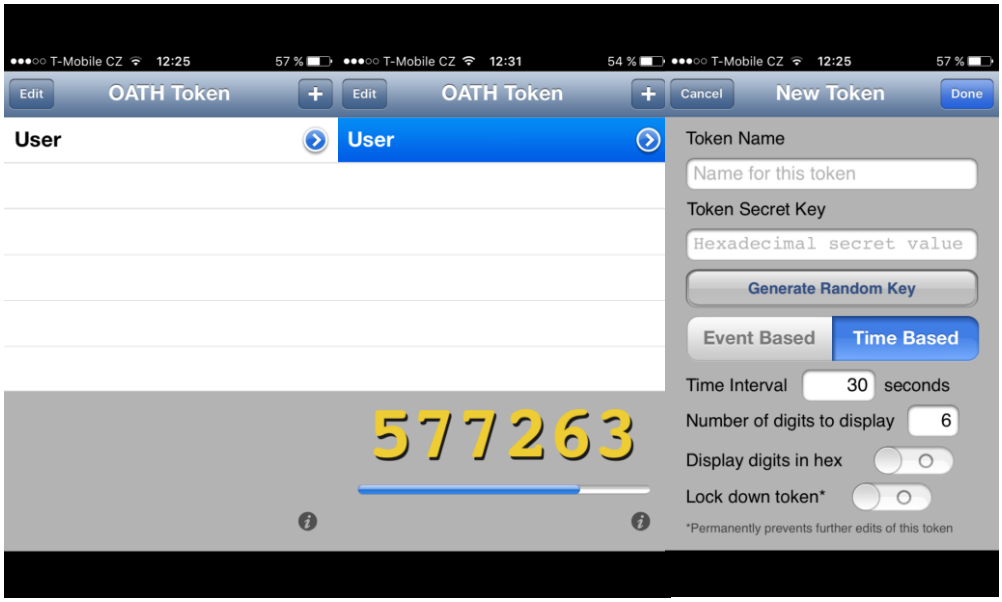
```
$znaky = ($_POST["znaky"]);

if (($username == '') || ($prvni == '') || ($druhe == '') || ($seed == ''))
{
        echo('chybne odeslani formulare!');
}
else if ((mysql_num_rows(mysql_query("select * from users where mail = '$username'")))
== '1')
{
        echo('existence');
}
else if ($prvni != $druhe)
{
echo('neshoda');
}
else
{
$md5heslo = md5($druhe);
$vloz = mysql_query("INSERT INTO users (mail, pass_hash) VALUES ('$username',
'$md5heslo')");
    require_once('/multiotp/multiotp.class.php');
    $multiotp = new Multiotp('MyPersonalEncryptionKey');
    $multiotp->SetUser($username);
    $multiotp->SetUserAlgorithm('TOTP');
    $multiotp->SetUserTokenSeed($seed);
    $multiotp->SetUserTokenNumberOfDigits($znaky);
    $multiotp->SetUserTokenTimeInterval($sec);
    $multiotp->WriteUserData();
    if ($vloz)
    {
        echo('true');
    }
    else
    {
        echo('neco se stalo');
    }
}
}
```

4.3 Generátor OTP (část klient)

Aplikací na chytré telefony, sloužících pro generování jednorázových hesel, je ke stažení poměrně velké množství. Nejvhodnější aplikací na platformě iOS podporující TOTP, doporučovanou tvůrci MultiOTP, je aplikace OATH Token, ke stažení zdarma na App Store. Tato open-source aplikace umožňuje dvoufaktorovou autentizaci a je kompatibilní se standardem RFC 4226. Výhodami této aplikace jsou možnost nastavení více tokenů, generace náhodných seedů a možnosti nastavení časového limitu a počtu číslic jednorázového hesla.

Nevýhodami jsou absence zapezpěčeného přístupu k aplikaci, neaktuální GUI a poměrně časté pády při chybně zadaných nebo nevyplněných hodnotách při přidávání tokenu. Pro ověřování běžných uživatelů, nebo v menších firmách nepracujících s rizikovými daty, je tato aplikace dostačující. Pro enterprise řešení, nebo generování hesel pro větší množství uživatelů by již bylo vhodné zauvažovat o zásahu do kódu této, případně vývoji nové, vlastní aplikace.



Obr. 10 – aplikace OATH Token – zdroj: autor

Vhodné aplikace lze nalézt i pro ostatní využívané mobilní platformy, stačí, aby podporovaly standard TOTP a umožňovaly volbu vlastního seedu, časového intervalu a délky jednorázového hesla. Pro Android je to například androidtoken.

5 Zhodnocení výsledků

5.1 Další způsoby zabezpečení

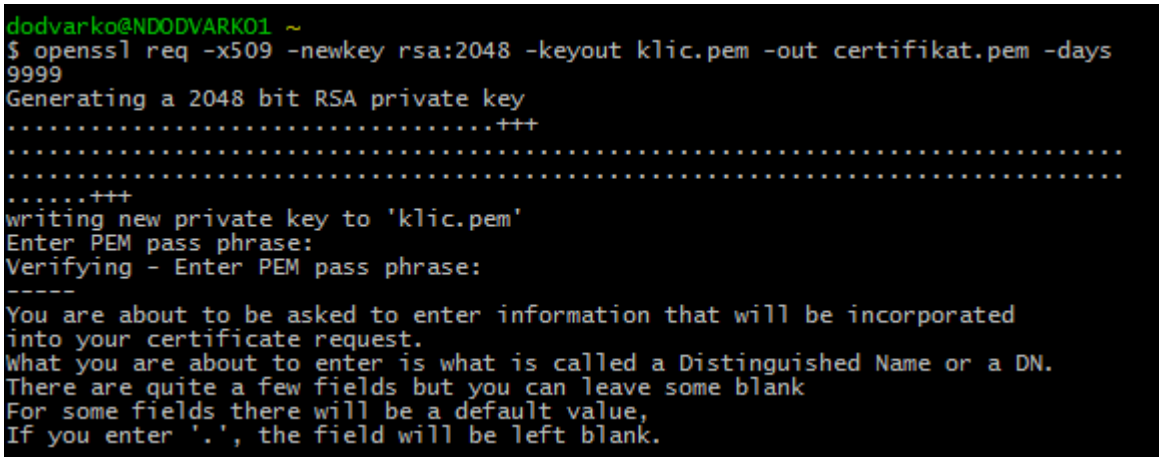
Mezi další způsoby, jak zvýšit úroveň zabezpečení jakékoli (nejen webové) aplikace patří zavedení politiky hesel, zahrnující požadavky na malá a velká písmena, číslice, nebo speciální znaky v heslech, frekvenci jejich změn a jejich neopakovatelnost.
Nejslabším článkem je vždy uživatel a tak je potřeba ho přimět používat co nejbezpečnější stálé heslo. Přestože je používána dvoufaktorová autentizace, je také vhodné zauvažovat nad logováním přístupů a blokací stálých i jednorázových hesel při opakovaném neúspěšném přihlášení. Přístup do administrace je samozřejmě možno omezit také na konkrétní IP adresy, což ovšem znovu zvyšuje technologickou náročnost a nepraktičnost pro uživatele.
Při administraci portálu více uživateli je praktické vytvořit role a jednotlivým uživatelům je přidělit, toto ovšem u malého projektu „from scratch“ vyžaduje spoustu nepříliš levného času.

5.2 SSL certifikát

Je třeba také zauvažovat o zakoupení SSL certifikátu, který je dnes již standardem na větších webech typu Seznam, Google atd. Běžný http protokol totiž přenáší data v textové podobě a potenciálnímu útočníkovi tak snadno odhalí komunikaci mezi klientem a serverem. Instalací SSL (Secure Socket Layer) certifikátu na web se komunikace stává šifrovanou a pro útočníka neodhalitelnou. Ceny důvěryhodného certifikátu vydaného ověřenou certifikační autoritou se momentálně pohybují od 200 do 900 korun za rok, certifikát si může také vygenerovat každý sám a to zcela zdarma, nicméně prohlížeče a antivirové programy budou upozorňovat na jeho nedůvěryhodnost. Vlastní certifikát je možné vytvořit pomocí linuxového terminálu, popřípadě jeho portu pro Windows s názvem Cygwin, zadáním následujícího příkazu:

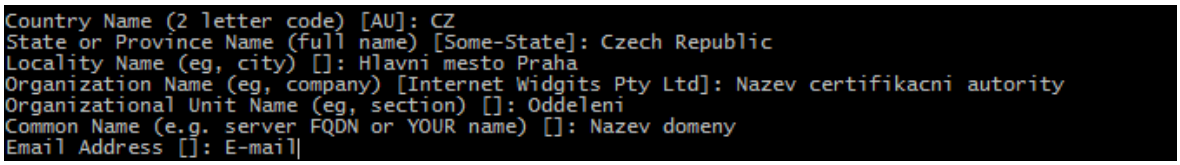
openssl req -x509 -newkey rsa:2048 -keyout klic.pem -out certifikat.pem -days 9999

Kde jsou libovolně voleny názvy souborů klíče a certifikátu a také doba platnosti certifikátu ve dnech. Po odeslání příkazu je třeba zadat a potvrdit heslo, to je shodné s názvem souboru certifikátu, pokud se certifikát jmenuje *certifikat*, heslo bude také *certifikat*.



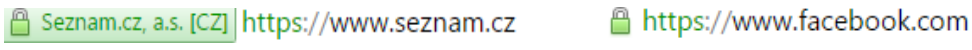
Obr. 11 – generování vlastního certifikátu - zdroj: autor

Poté je potřeba vyplnit údaje o certifikační autoritě. Důležité je kód země uvést dle standardu ISO a do Common name vyplnit název domény, pro kterou bude certifikát platit, je důležité rozlišit tvar bez prefixu WWW i s ním.



Obr. 12 – generování vlastního certifikátu - zdroj: autor

Takto vygenerovaný certifikát je poté třeba nahrát na hosting. Komunikace se serverem je tímto zabezpečena a uživatel je o tom informován v adresním řádku, buď zeleným nápisem s názvem a kódem země, nebo alespoň zámečkem oznamujícím šifrované spojení HTTPS.



Obr. 13 – informace o zabezpečeném spojení – zdroj: autor

6 Závěr

Tato bakalářská práce objasnila problematiku jednorázových hesel a podařilo se úspěšně implementovat třídu MultiOTP do malého projektu. Bylo by ještě možné do budoucna tuto administraci rozšířit o možnost správy uživatelů, tisk jejich údajů a jejich mazání. Implementace třídy proběhla bez větších problémů a bylo by možné ji provést u téměř jakýchkoli přihlašovacích skriptů. Jak je zmíněno v předchozí kapitole, bylo by jistě dobré pro zvýšení bezpečnosti blokovat přihlašovací údaje při více neúspěšných pokusech o přihlášení, z důvodu zabránění útokům hrubou silou, a nainstalovat SSL certifikát na server. Jednorázová hesla jsou velkým přínosem pro bezpečnost jednotlivců i firem všech velikostí. Vzhledem k tomu, že žijeme v době chytrých telefonů a hodinek a stále existují hesla typu Banka123, se jejich uplatnění bude stále více rozšiřovat. Největší rozmach samozřejmě tento druh zabezpečení zažívá v bankovním sektoru, kde se s ním setkáme při řadě běžných uživatelských činností ve většině bank, určitě by se o něj měly zajímat ale i další instituce, které jsou ohrožené úniky dat. Nedávná kauza hacknutí e-mailu předsedy vlády jen potvrzuje, že ani vysocí vládní představitelé si správně nechrání svá data před jejích odcizením a možným zneužitím. Ve všech případech je nezbytné, aby uživatelé přihlašující se ať už do tohoto portálu, nebo kamkoli jinam, měli aktualální operační systém, bezpečný prohlížeč, a spolehlivý antivirový program, schopný odhalit moderní hrozby pro bezpečnost uživatele.

MultiOTP je praktickým a užitečným nástrojem, který v takovém rozsahu není v žádném jiném programovacím jazyce zpracován. Zajímavá by jistě byla také jeho implementace do nějakého většího projektu s infrastrukturním zázemím na realizaci odesílání přes SMS bránu a využití dalších praktických funkcí, které tato obsáhlá třída umožňuje, jako například generování QR kódu s informacemi pro generátor OTP.

7 Seznam použitých zdrojů

A One-Time Password System. 1. Bloomfield: Kaman Sciences Corporation, 1998.

HOTP: An HMAC-Based One-Time Password Algorithm. 1. Virginia: Verisign, 1998.

HMAC: Keyed-Hashing for Message Authentication. 1. New York: IBM, 1997.

TOTP: Time-Based One-Time Password Algorithm. 1. Virginia: Verisign, Inc., 2011.

One-Time Passwords – HOTP and TOTP. Aldaris' blog [online]. [cit. 2015-12-11]. Dostupné z: http://blogs.forgerock.org/petermajor/2014/02/one-time-passwords-hotp-and-totp/

CHESWICK, William R a Steven M BELLOVIN. *Firewalls and internet security: repelling the wily hacker*. Reading, Mass.: Addison-Wesley, c1994. Addison-Wesley professional computing series. ISBN 0-201-63357-4.

DOSEDĚL, Tomáš. *Počítačová bezpečnost a ochrana dat*. Vyd. 1. Brno: Computer Press, 2004. ISBN 80-251-0106-1.

CONDLIFFE, Jamie. *The 25 Most Popular Passwords of 2014: We're All Doomed* [online]. [cit. 2015-10-23]. Dostupné z: http://gizmodo.com/the-25-most-popular-passwords-of-2014-were-all-doomed-1680596951

2006. *Velký průvodce infrastrukturou PKI a technologií elektronického podpisu*. Brno: Computer Press, s. 31-36. ISBN 80-251-0828-7.

2011. CIAMPA, Mark. *Security+ Guide to Network Security Fundamentals*. Western Kentucky University: Course Technology, s. 268. ISBN 978-1111640125.

One-time password (OTP) definition [online]. [cit. 2015-10-23]. Dostupné z: http://searchsecurity.techtarget.com/definition/one-time-password-OTP

WOODFORD, Chris. *Two-factor authentication* [online]. [cit. 2015-10-23]. Dostupné z: http://www.explainthatstuff.com/how-security-tokens-work.html

IACONA, Louis J. Lamport's one-time password algorithm (or, don't talk to complete strangers!): A design pattern for securing client/service interactions with

OTP. *Javaworld* [online]. 2009 [cit. 2015-11-03]. Dostupné z: http://www.javaworld.com/article/2078022/open-source-tools/lamport-s-one-time-password-algorithm--or--don-t-talk-to-complete-strangers--.html

Creating a Self-Signed SSL Certificate [online]. [cit. 2016-02-08]. Dostupné z: https://devcenter.heroku.com/articles/ssl-certificate-self

MultiOTP - the free strong authentication PHP library and command line tool to check a TOTP/HOTP/mOTP token [online]. [cit. 2016-02-07]. Dostupné z: https://www.multiotp.net/

OATH Token [online]. [cit. 2016-01-31]. Dostupné z: https://github.com/archiecobbs/oathtoken