Katedra Informatiky
Fakulta matematiky, fyziky a informatiky
Univerzita Komenského, Bratislava

Autentifikácia jednorázovými heslami generovanými pomocou mobilného telefónu

(Bakalárska práca)

Kamil Kuboň

Študijný odbor 9.2.1 Informatika

Vedúci: RNDr. Jaroslav Janáček

Bratislava, 2008

Čestne prehlasujem, že som túto diplomovú prácu vypracoval samostatne s použitím citovaných zdrojov.

.....

Poďakovanie

Za inšpiráciu a podnety k tejto bakalárskej práci chcem poďakovať hlavne svojmu školiteľovi RNDr. Jaroslavovi Janáčkovi. Poďakovanie tiež patrí mojej rodine, priateľom a priateľkám.

Abstrakt

V tejto práci sa popisuje realizácia systému na autentifikáciu jednorázovými heslami generovaných pomocou kryptografickej funkcie HMAC s hashovacou funkciou SHA1. Heslá užívateľ generuje pomocou mobilného telefónu. Serverová časť systému je postavená na platforme Linux s použitím webmailu SquirrelMail.

Kľúčové slová: *jednorázové heslo*, *autentifikácia*, *mobilný telefón*, *Linux*

Predhovor

Jednou z najčastejšie používaných služieb, ktoré poskytuje internet je internetová pošta. Prirodzenou požiadavkou je mať svoju emailovú schránku kedykoľvek k dispozícii - na cestách, v zamestnaní, v škole, doma. Často treba riešiť otázku pripojenia sa k emailu pomocou počítača, o ktorého bezpečnosti nemáme istotu. Dôsledkom takéhoto kroku môže byť odcudzenie prihlasovacích údajov, ktoré môžu byť neskôr zneužité. Tomu sa snaží táto práca zabrániť. V prípade internetového bankovníctva sa zvyknú na tento účel používať grid karty, prípadne sms správy s autentifikačnými reťazcami. Systém vytvorený a popísaný v tejto práci rieši tento problém pomocou autentifikácie jednorázovými heslami za pomoci mobilného telefónu.

Obsah

Kapitola 1

Úvod

V súčasnej ére rozmachu informačných a komunikačných technológií (IKT) sa čoraz viac dostáva do popredia otázka bezpečnosti informačných systémov. Veľký podiel výsledkov každodennej ľudskej práce, osobných informácií, administratívnych údajov sa uchováva v elektronickej podobe, či už na osobných počítačoch alebo serveroch. Rôzna miera dôležitosti týchto údajov vyžaduje rozličné prístupy na ochranu ich dôvernosti, integrity a autentickosti, prípadne iných ďalších bezpečnostných aspektov.

Spôsob ochrany závisí na prostredí, kde je informácia v rámci IKT zadávaná, ktorým sa prenáša a kde sa ukladá. Bezpečné putovanie dát internetom je často riešené HTTPS protokolom, obzvlášť citlivé dáta bývajú na disku šifrované, ale samotný zdroj, odkiaľ informácie putujú ďalej, môže byť často kompromitovaný. Príkladom môže byť počítač, ktorý slúži počas dňa mnohým ľuďom(v internetovej kaviarni, v škole), ale ohrozená môže byť aj pracovná stanica využívaná jednotlivcom. Čítanie a odosielanie elektronickej pošty, využívanie internet bankingu a internetových obchodov býva ohrozované rôznymi aplikáciami schopnými napríklad odchytiť heslá a následne ich odoslať útočníkovi, meniť odosielanú a prijímanú informáciu, prípadne zneužiť počítač bez vedomia užívateľa na nekalé ciele. Cieľom tejto práce

je zvýšiť bezpečnosť používania internetovej pošty na počítačoch, o ktorých bezpečnosti máme pochybnosti, prostredníctvom jednorázových hesiel. Aj v prípade odchytenia hesla zostane z dlhodobého hľadiska poštová schránka pred útočníkom chránená, pretože prístup k nej bude vyžadovať nové heslo, takže potenciál útočníka zostáva obmedzený iba na dobu platnosti hesla, ktoré momentálne odchytil a tým sa jeho možnosti značne zúžia.

Kapitola 2

Návrh systému

2.1 Špecifikácia systému s požiadavkami

Pre systém na zabezpečenie mailu spomínaný v úvode počas analýzy vyplynuli nasledovné požiadavky:

1. generovať jednorázové heslá mobilným telefónom bez fyzického spojenia s kompromitovaným terminálom
2. aplikácia na mobilnom telefóne musí byť spustiteľná na čo najväčšej škále zariadení
3. serverová časť musí bežať na OS Linux
4. oddelenie webmailového rozhrania od poštového servera
5. vytvorenie samostatnej aplikácie obsluhujúcej heslá komunikujúcej s webmailom cez TCP/IP protokol, ktorá bude bežať s právami špeciálne vytvoreného užívateľa.

2.2 Popis riešenia

Realizáciou uvedených požiadaviek sú časti:

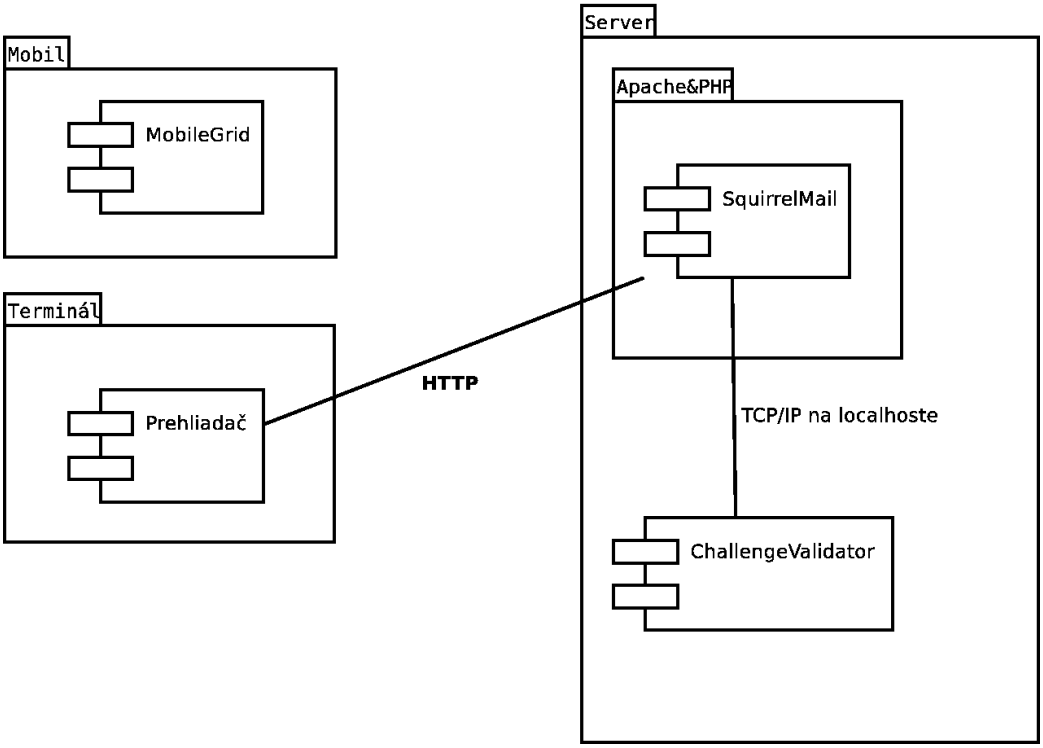
Mobilná aplikácia *MobileGrid*, upravený webmail *SquirrelMail* a C++ aplikácia *ChallengeValidator*.

Aplikácia MobileGrid je distribuovaná medzi užívateľov, pričom každý má 2 heslá. Jedno slúži na generovanie jednorázových hesiel a zadáva sa do mobilu. (V ďalšom texte ho budeme nazývať *heslo pre mobil*). Toto je tiež uložené na serveri a má k nemu prístup aplikácia ChallengeValidator. Druhé heslo slúži na autentifikáciu s poštovým serverom a je uložené na serveri (nazvime ho *heslo pre server*) spoločne s prvým heslom. SquirrelMail a ChallengeValidator bežia na rovnakom serveri, ale s odlišnými užívateľskými právami. ChallengeValidator bude bežať pod právami užívateľa, ktorý ako jediný(s výnimkou roota) bude môcť čítať svoj konfiguračný súbor. Distribúcia komponent je zachytená na obrázku 2.1

Na realizáciu generovania jednorázových hesiel sa zvolila kryptografická primitíva HMAC-SHA1 [rfc97a],[NIS02], ktorá je implementovaná v aplikácii MobileGrid a na serveri v ChallengeValidatore. Užívateľ sa prihlási na upravený SquirrelMail, ktorý ho vyzve reťazcom nazvaným *Challenge* na zadanie jednorázového hesla. Tento reťazec je vytvorený aplikáciu ChallengeValidator po dopyte zo SquirrelMailu cez TCP/IP protokol na localhost, kde má ChallengeValidator otvorený TCP/IP server.

Užívateľ vloží tento reťazec do MobileGridu, ktorý beží na jeho mobilnom telefóne spoločne so svojím heslom určeným špeciálne pre účel generovania jednorázových hesiel - viď obrázok 2.2 v podkapitole 2.2.1.

MobileGrid následne spočítava HMAC-SHA1 s týmito parametrami a zobrazí 2 formy výsledku *Stronger* a *Normal*, oba slúžia ako jednorázové heslo. Užívateľ si vyberie tú formu, ku ktorej ho vyzýva server a vloží ju do webového prehliadača spoločne so svojím prihlasovacím menom do SquirrelMailu



Obr. 2.1: Schéma systému pre jednorázové heslá

- viď obrázok 2.3 v podkapitole 2.2.2.

SquirrelMail sa následne pripojí na aplikáciu ChallengeValidator. ChallengeValidator znova spočíta jednorázové heslo, pričom parametrami sú na serveri uložené heslo pre mobil a zapamätaný Challenge. Oba výsledky sa porovnajú a na základe úspešného porovnania upravený SquirrelMail užívateľa autentifikuje v rámci poštového servera pomocou hesla pre poštový server.

Samotné *jednorázové heslo* je upravený MAC(*Message authentication code* [NIS02]) po aplikácii *hesla pre mobil* ako kľúča a reťazca *Challenge* ako vstupného textu. Výsledkom je 160 bitový reťazec, ktorý by mal v hexa zápise 40 znakov. Vzhľadom na ergonómiu používania systému je vhodné tento reťazec skrátiť, čo sa realizuje kódovaním Base64 [BAS] s upravenou 64 znakovou

abecedou:

```
"ABCDEFGH*JKLMNOPQRSTUVWXYZabcdefghijk#mnopqrstuvwxyz0123456789+."
```

Vstup sa spracováva po 6-tich bitoch, čo sa realizuje postupným prekladom každých 3 bajtov vstupu na 4 znaky z vyššie definovanej abecedy. V prípade nedeliteľnosti bajtovej dĺžky vstupu 3-mi sa vstup doplní nulami. Abeceda je upravená kvôli optickej podobnosti veľkého i a malého L, ktoré su nahradené znakmi '*' a '#'. Štandartný znak '/' bol nahradený bodkou. Nazvime toto kódovanie *Upravené Base64 kódovanie*.

Upravené Base64 kódovanie je aplikované na MAC dĺžky 20 bajtov - forma *Stronger* a na 12 bajtový MAC (prvých 96 bitov z pôvodných 160) - forma *Normal*.

V ďalšom texte je presnejší popis jednotlivých komponent, ich interakcie, popis jednoduchého protokolu medzi Squirrelmailom a ChallengeValidatorom. V ďalšej kapitole sa nachádzajú implementačné detaily s časťami zdrojového kódu a jeho vysvetlením.

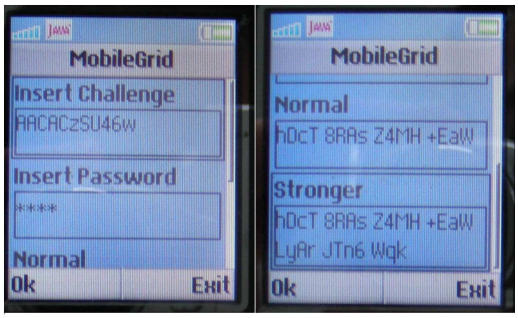
2.2.1 MobileGrid

Komponent MobileGrid je mnou napísaná aplikácia mobilná aplikácia napísaná v jazyku Java so špecifikáciami CLDC 1.0 a MIDP 1.0. Využíva kryptografickú primitívu HMAC-SHA1, pomocou ktorej sa generujú jednorázové heslá. Špecifikácie CLDC 1.0 [CLD]a MIDP 1.0 [MID] zaručujú nasadanie na čo najväčšom množstve mobilných telefónov. Vyvíjaná bola v prostredí Netbeans 6 [NET] pomocou balíčka na vývoj mobilných aplikácií.

Na obrázku 2.2 je screenshot aplikácie MobileGrid spustenej na mobilnom telefóne Ericsson W200.

Ovládanie aplikácie je intuitívne, užívateľ musí prepísať jednu z foriem jednorázového hesla bez medzier do prehliadača.

Na distribúciu aplikácie sa najlepšie osvedčil prenos cez bluetooth/infra-

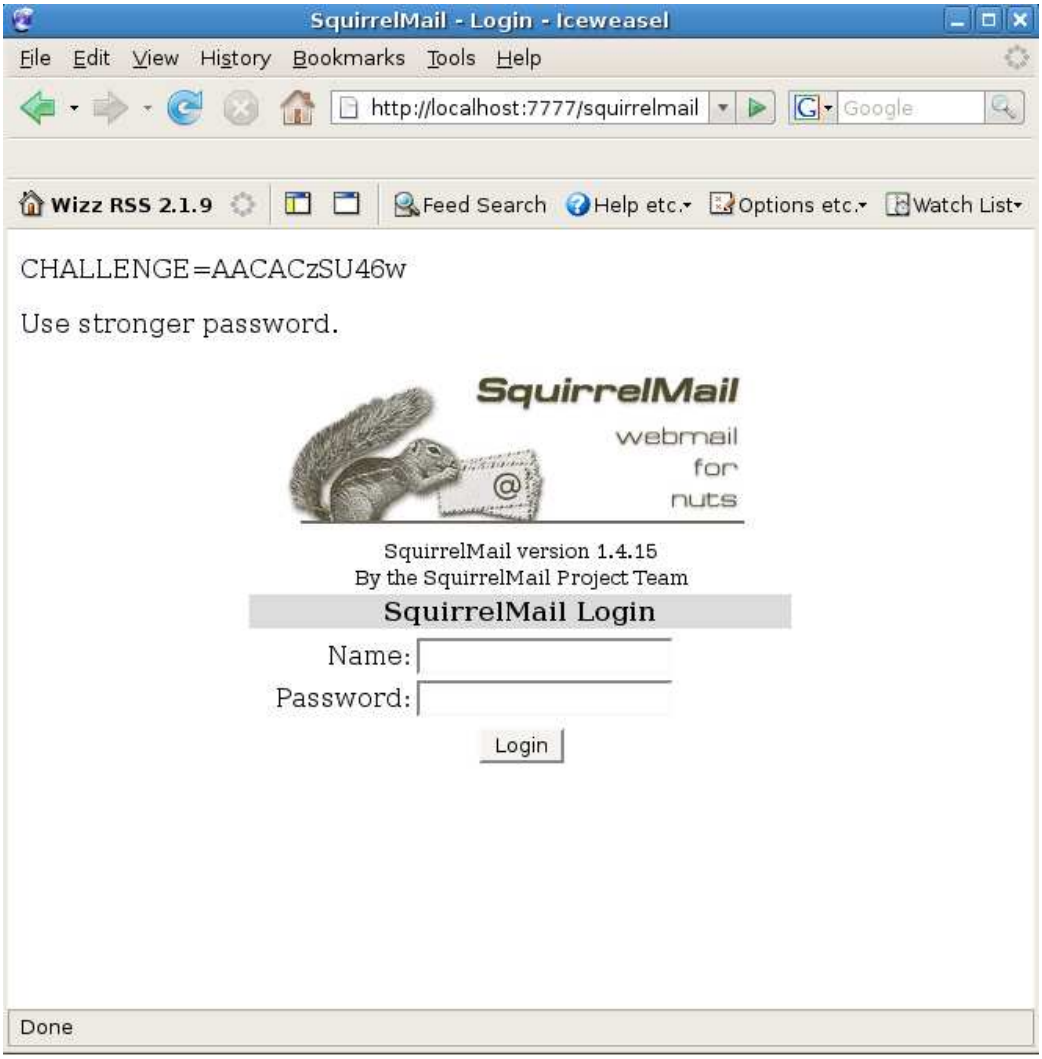


Obr. 2.2: Screenshot aplikácie MobileGrid spustenej na mobilnom telefóne Ericsson W200

red z mobilu kde je už nainštalovaná, ale samotný súbor MobileGrid.jar sa dá inštalovať aj inými štandartnými cestami ktoré dané zariadenie umožňuje (dátový kábel, SD karta..).

2.2.2 SquirrelMail

Za webmailový systém bol zvolený opensource produkt SquirrelMail, [SM] s možnosťami konfigurácie pre ľubovoľný IMAP/POP3/SMTP server, veľkou škálou rozšírení(pluginov) a s pravidelnými bezpečnostnými aktualizáciami. Napísaný je v skriptovacom jazyku PHP [PHP] s voľným zdrojovým kódom, čo z neho robí ideálneho kandidáta na nami požadované úpravy. V produkčnom prostredí sa SquirrelMail spojazdní na webserveri Apache [Apa] s právami samostatného užívateľa(aby nemal root práva a aby nemohol čítať konfiguračný súbor ChallengeValidatora). Počas vývoja som používal verziu 1.4.15. Na obrázku 2.3 je zachytený SquirrelMail po úprave komponentou PhpMobileGrid.



Obr. 2.3: Upravené prihlasovacie okno do squirrel mailu

2.2.3 ChallengeValidator

ChallengeValidator je konzolová aplikácia pre OS Linux mnou napísaná v jazyku C++. V produkčnom prostredí bude bežať pod právami špeciálne na to určeného užívateľa, ktorý ako jediný(s výnimkou roota) bude mať práva

na čítanie a zápis konfiguračného súboru *config*. Tento musí byť umiestnený v rovnakom adresári ChallengeValidator. Má nasledovnú štruktúru:

```
<forma_hesla><koniec riadku>
<port><koniec riadku>
<timeout><koniec riadku>
[<meno> <heslo pre mobil> <heslo pre pošt.server> <koniec riadku>]*
```

`<forma_hesla>` je celé číslo, ak je nenulové, použije sa autentifikácia pomocou *Stronger* hesla, v opačnom pomocou *Normal*. `<port>` je port, na ktorom bude ChallengeValidator očakávať spojenia. `<timeout>` je počet sekúnd od vygenerovania a zobrazenia hesla v SquirrelMail prihlasovacom okne, počas ktorého sa musí užívateľ autentifikovať. `<port>` je port na ktorom bude ChallengeValidator očakávať spojenia. Zvyšok súboru tvoria po riadkoch údaje o užívateľovi.

Komunikácia s PHP kódom PhpMobileGrid je realizovaná nasledovným protokolom. Kódy správ s prefixom CHALPROT sú v súbore ChallengeProtocol.h

Na získanie *Challengu* sa pošle správa

`<CHALPROT_REQ_CHAL_ID>`

Server odpovie správou

`<CHALPROT_REQ_CHAL_ID> <ID> <Challenge>`

`<ID>` je číslo slúžiace na na identifikáciu dopytu, ktorý klient práve spravil, v rámci neho si ChallengeValidator zapamätá `<Challenge>`. Od momentu vygenerovania `<ID>` má klient časový interval určený konfiguračným súborom, počas ktorého je `<ID>` platné. Samotná autentifikácia prebieha správou

`<CHALPROT_REQ_VALID_ID> <ID> <prihlasovacie meno> <jednorazove heslo>`

V prípade úspešnej autenfikácie dostane klient správu

`<CHALPROT_REQ_VALID_ID_OK> <heslo pre server>`

V prípade, že platnosť `<ID>` vyprší (čas je definovaný timoutom viď v súbore config), dostane klient správu

`<CHALPROT_REQ_VALID_ID_TIMEOUT>`

V prípade akejkoľvek inej chybu sa zašle správa

`<CHALPROT_REQ_FAILED>`

Všetky správy sa posielajú ako reťazce s fixnou dĺžkou `CHALPROT_BASIC_BUF_SIZE` a ich jednotlivé zložky sú oddelené medzerami.

2.2.4 PhpMobileGrid

PhpMobileGrid je mnou implementované rozhranie medzi SquirrelMailom a aplikáciu ChallengeValidator. Pozostáva zo sady funkcií napísaných v jazyku PHP, ktoré sa dajú jednoducho prispôsobiť a upraviť aj pre úplne iný systém ako je SquirrelMail.

2.2.5 Produkčné prostredie

Systém bol vyvíjaný na HTTP serveri Apache verzie 2.2.8, skriptovacom jazyku PHP5 verzie 5.2.6 a linuxovom OS Debiane Etch [Deb]. Ako testovací IMAP server bol zvolený Courier Mail Server [COU] inštalovaný priamo cez balíčkový systém Debianu. Podobné prostredie sa očakáva aj pri prípadnom využívaní systému, pričom kompaktibilita riešenia závisí do od aktuálnej verzie systému SquirrelMail.

2.3 Analýza bezpečnosti

Bezpečnosť systému pre odhalením *hesla pre poštový server* spočíva v sile kryptografickej hashovacej SHA1 používanej v rámci HMAC viď [NIS02]. Skrátená forma 96 bitov zjednodušuje zadávanie hesla z mobilu do prehliadača, pri zachovaní rozumnej miery bezpečnosti - viď [NIS02] kapitola *Trun-*

cated Output strana 4. V prípade požiadavky na vyššiu úroveň je v aplikácii ChallengeValidor konfigurovateľná silnejšia forma. Samotné oddelenie kódu pre generovanie Challengov a overovanie jednorázového hesla s prístupom do súboru s heslami do samostatnej aplikácie má za následok, že aj v prípade bezpečnostnej chyby Squirrel mailu, Apachu alebo PHP bežiacich bez rootovských práv bude súbor s heslami chránený, pretože výhradný prístup k nemu má root a aplikácia ChallengeValidator.

Kapitola 3

Implementácia

3.1 MobileGrid

Kľúčový zdrojový kód aplikácie Mobile Grid je obsiahnutý v súboroch:

`HmacSha1.java`
`Sha1.java`
`MobileGrid.java`

V súbore MobileGrid.java je definovaný midlet MobileGrid, ktorý vytvára GUI a sú v ňom definované odozvy od užívateľa implementáciou interfacu *CommandListener*. Po stlačení tlačidla *Ok* sa zavolá funkcia `setDigest()`:

```
1     private void setDigest() {
2         HmacSha1 hmac = new HmacSha1();
3         hmac.makeHMAC(HmacSha1.stringToByteArray(tfPass.getString()),
4                       HmacSha1.stringToByteArray(tfChallenge.getString()));
5         tfHMAC.setString(hmac.getEncoded64(12));
6         tfHMACStronger.setString(hmac.getEncoded64(20));
7     }
```

Premenné tfHMAC a tfHMACStronger označujú textfield polia do ktorých sa vložia jednorázové heslá,tfPass a tfChallenge polia, do ktorých sa zadáva *heslo pre mobil* a *challenge*.

Trieda HmacSha1 má tieto metódy:

- `public HmacSha1()` konštruktor
- `public void makeHMAC(byte[] K,byte [] text)` základná metóda, `K` je kľúč, `text` je vstup do hashovacej funkcie.
- `public String getString()` vracia MAC v hexadecimálnej forme, využíva sa pri testoch, viď kapitola 4.
- `public static int getUnsigned(byte b)` pomocná funkcia na získanie neznamienkovej hodnoty bajtu b, pretože jazyk Java má iba znamienkové bajty
- `public static String encode64(byte input[],`
  `int length, boolean useWhiteSpaceBy4)`

  táto funkcia vykoná *Upravené base64 kódovanie*, pričom využije prvých `length` bajtov z poľa `input`. premenná `useWhiteSpaceBy` určuje, či majú byť po každých 4 znakoch na výstupe medzery
- `public String getEncoded64(int MACbytesFromRight)`
- `public SHA1 getOHash()`
- `public static boolean testCase(byte[] key,`
  `byte[] text,String result,int testIndex)`

  vytvorí MAC s parametrami `key` a `text` a výsledok porovná s reťazcom `result`(ktorý by mal byť v hexadecimálnej forme bez prefixu 0x) - viď 4

- `public static byte[] stringToByteArray(String str)` pretransformuje vstup vo forme UTF-16 Java String na súbor bajtov, pričom z každého znaku vstupu sa vezme prvý bajt.
- `public static void main(String args[])` implementuje testovanie na vektoroch z [RFC97b], viď kapitolu 4.

Voľne šíriteľná implementácia hashovacej funkcie Sha1 v súbore SHA1.java je prevzatá z projektu Jabber.org [JAB] (presnejšie informácie sú v komentároch k tejto triede.)

Samotná implementácia metódy `public void makeHMAC(byte[] K,byte [] text)`:

```
1       /*
2        * K - tajny kluc medzi odosielatelom a prijimatelom
3        * text - vstupny retazec
4        */
5       public void makeHMAC(byte[] K,byte [] text){
6           /*
7            *jednotlive kroky aj s popisom zodpovedaju
    FIPS PUB 198 strana 4
8            */
9
10          int B=64;//vstupna velkost bloku do sha1
11          int L=20;// velkost vystupu z sha1
12          int ipadByte=0x36;
13          int opadByte=0x5c;
14
15          byte[] K0=null; //v krokoch 1-3 naplnime K0
16          if(K.length==B){
17              //krok 1
18              K0=K;
19          }else if(K.length>B){
```

```
20              //krok 2
21              K0=new byte[B];
22
23              //zahashujeme K a ziskame prvych L bytov
24              SHA1 sha1=new SHA1();
25              sha1.init();
26              sha1.update(K);
27              sha1.finish();
28              //do K0 doplnime prvych L bytov z
29              //vystupu sha1 a (B-L) bytov 0x00
30              for(int i=0;i<B;i++){
31                  if(i<L){
32                      K0[i]=sha1.digestBits[i];
33                  }else{
34                      K0[i]=0x00;
35                  }
36              }
37          }else if(K.length<B){
38              //krok 3
39              K0=new byte[B];
40              //pripojime  (B-K.length) bytov 0x00 k vstupu K
41
42              for(int i=0;i<B;i++){
43                  if(i<K.length){
44                      K0[i]=K[i];
45                  }else {
46                      K0[i]=0x00;
47                  }
48              }
49          }
50          //krok 4
```

```
51          //ziskame K0 xor ipad
52          byte[] K0_xor_ipad= new byte[B];
53          for(int i=0;i<B;i++){
54              K0_xor_ipad[i]=(byte)(K0[i] ^ ipadByte);
55          }
56          //krok 5
57          SHA1 iHash = new SHA1();
58          iHash.init();
59          iHash.update(K0_xor_ipad);
60          //pripojime vstup text k K0_xor_ipad
61          iHash.update(text);
62
63          //krok 6 spocitame hash
64          iHash.finish();
65
66          //krok 7 spocitame K0 xor opad
67          byte[] K0_xor_opad= new byte[B];
68          for(int i=0;i<B;i++){
69              K0_xor_opad[i]=(byte)(K0[i] ^ opadByte);
70          }
71
72          //krok 8
73          oHash = new SHA1();
74          oHash.init();
75          //pripojenie vysledku z kroku 7
76          oHash.update(K0_xor_opad);
77          //pripojenie vysledku z kroku 6
78          oHash.update(iHash.digestBits);
79          //krok 9:
80
81          oHash.finish();
```

```
82          //krok 10 - ako velkost MACU ponechame L
83      }
```

3.2 PhpMobileGrid

Úprava webmailu SquirrelMail v súlade s definovanou špecifikáciou z kapitoly 2 spočíva v prekopírovaní súborov:

`mobilegrid/mobilegrid.php`
`src/login.php`
`src/redirect.php`

Súbor `mobilegrid.php` obsahuje tieto funkcie:

- `function getCharEndedString(&$str,&$beginIndex,$chr)`

  vráti podreťazec reťazca `$str` začínajúci sa `$beginIndex` a končiaci znakom `$chr`

- `function generateChallenge()` pripojí sa na ChallengeValidator, vypýta si *Challenge* a uloží ho aj s identifikátorom `<ID>` do session - viď kapitolu 2

- `function isStrongerChallenge()` funkcia slúžiaca ako indikátor toho, aká forma autentifikácie sa používa - či *Stronger* alebo *Normal*. V prípade prekonfigurovania súboru config pre ChallengeValidator treba zmeniť aj telo tejto metódy.

- `function authenticateUser($username,$user_answer,&$passwordForMail,&$timeout)` vykoná autentifikáciu jednorázovým heslom pripojením sa na challengeValidator. Použije k tomu uložené `<ID>` zo session.

- `function testChallengeValidator()` pre testovacie účely je možné použiť túto funkciu, ktorá sa pripojí na ChallengeValidator a zistí odozvu.

- `function validatorSocketRequestResponse($code,$str,&$response)` Základná funkcia realizujúca dotaz cez TCP/IP zložený z premennej `$code` a reťazca `str`, odpoveď vráti v premennej `$response`. V prípade chyby vráti návratovú hodnotu `false`.

Súbory index.php a redirect.php obsahujú volania funkcií z mobilegrid.php a sú prevzaté z distrúcie SquirrelMailu 1.4.15. Úpravy sú minimálne, a v komentároch označené ako `//////////MOBILE GRID//////////` . V prípade zmeny pôvodných súborov login.php, redirect php by malo stačiť prekopírovanie kódu označeného týmto komentárom.

V súbore login.php je nasledovná úprava:

```
//////////MOBILE GRID//////////////////////
$chal=generateChallenge();
echo "<p> CHALLENGE=".$chal."</p>";
if(isStrongerChallenge()==true){
echo "<p> Use stronger password.</p>";
}else{
echo "<p> Use normal password.</p>";
}
////////////////////////////////////////////
```

V súbore redirect.php je nasledovná úprava:

```
//////////MOBILE GRID////////////////////
// testChallengeValidator();
$temp=$secretkey;
```

```
if(authenticateUser(trim($login_username),$temp,$secretkey,$timeout)){
// echo "One time pass authentification succeded, trying IMAP:<br>";
}else{
if(!$timeout){
logout_error( _("One time password failed!") );
}else{
logout_error( _("One time password validation timeout!") );
}
exit;
}
//////////////////////////////////////////
```

3.3 ChallengeValidator

Zdrojový kód pozostáva z nasledovných súborov:

- `ChallengeProtocol.h` definuje konštanty pre komunikačný protokol.
- `ChallengeValidator.h`
- `ChallengeValidator.cpp` implementuje ChallengeValidator
- logger.h
- `logger.cpp` v tomto súbore sa dá zmeniť frekvencia logovania cez premennú `granularity`
- `mainServer.cpp` obsahuje funkciu `main`
- `ValServerSocket.h`
- `ValServerSocket.cpp` zaobaľuje prácu s TCP/IP socketmi

V súbore `mainServer.cpp` je main funkcia, ktorá vytvorí inštanciu triedy ChallengeValidator, ktorá najskôr načíta konfiguráciu zo súboru `config`. Na vytvorenie TCP/IP servera sa použije trieda ValServerSocket, ktorá zabezpečuje volania API na prácu so socketmi(vo výpise nižsie je premenná `server` inštanciu práve tejto triedy).

Kľúčová metóda je `ChallengeValidator.start()`, v ktorej sa obsluhujú TCP/IP dopyty od SquirrelMailu.

```
1
2  void ChallengeValidator::start(){
3    try{
4      server.startListening(port);
5      while(true){
6        int socket;
7        server.acceptClient(socket);
8        bool closedConnection;
9        try{
10         clearBuf();
11         log("waitData",D_GRAN_TESTING);
12         server.waitData(socket,basicBuf,CHALPROT_BASIC_BUF_SIZE,
13           closedConnection);
14         if(closedConnection)continue;
15         log(basicBuf,D_GRAN_TESTING);
16         switch(basicBuf[0]){
17           case CHALPROT_TEST:
18             processTest(socket);
19           break;
20           case CHALPROT_REQ_CHAL_ID:
21             processChallenge(socket);
22           break;
23           case CHALPROT_REQ_VALID_ID:
```

```
24                 processValidation(socket);
25               break;
26               default:
27               break;
28             }
29           }catch(ValServerException &exc){
30             server.closeSocket(socket);
31             continue;
32           }
33           server.closeSocket(socket);
34         }
35       }catch(ValServerException &exc){
36         log(exc.getMsg(),D_GRAN_PRODUCTION);
37       }
38   }
```

Na počítanie HMAC-SHA1 sa v metóde `getHmacSha1` používa knižnica `Mhash` [MHA].

Kapitola 4

Testovanie

4.1 Testovanie HMAC-SHA1 v aplikácii MobileGrid

Implementácia HMAC-SHA1 v aplikácii triedou `HmacSha1` bola otestovaná viacnásobne.

- Pri zbežnom testovaní autentifikovanie voči serveru, kde je implementácia z [MHA] prebiehalo korektne.
- Testovanie tvorby MAC-u pomocou HmacSha1 bez *Upraveného base64 kódovania* podľa testovacích prípadov 1 až 7 uvedených v [RFC97b] Testovanie je implementované v metóde `HmacSha1.main()` a dá sa spustiť príkazom

  ```
  java -cp .;MobileGrid.jar mobilegrid.HmacSha1
  ```

  v adresári so súborom MobileGrid.jar.

  Tu je ukážka z testov:

```
1     public static void main(String args[]){
2         //TESTOVANIE PODLA RFC 2202
```

```
3           boolean ok = true;
4
5           byte [] key=new byte[20];
6           for(int i=0;i<20;i++){key[i]=(byte)0x0b;}
7           ok= testCase(key, stringToByteArray("Hi There"),
8                   "b617318655057264e28bc0b6fb378c8ef146be00",1)&&ok;
9
10          ok= testCase(stringToByteArray("Jefe"),
11                  stringToByteArray("what do ya want for nothing?"),
12                  "effcdf6ae5eb2fa2d27416d5f184df9c259a7c79",2)&&ok;
13
14          key=new byte[20];
15          for(int i=0;i<20;i++){key[i]=(byte)0xaa;}
16          byte [] data=new byte[50];
17          for(int i=0;i<50;i++){data[i]=(byte)0xdd;}
18          ok=testCase(key, data,
19                  "125d7342b9ac11cd91a39af48aa17b4f63f175d3",3)&&ok;
20
21          key=new byte[]{0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,
22              0x09,0x0a,0x0b,0x0c,0x0d,0x0e,0x0f,0x10,0x11,
23              0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19};
24          data=new byte[50];
25          for(int i=0;i<50;i++){data[i]=(byte)0xcd;}
26          ok= testCase(key, data,
27                  "4c9007f4026250c6bc8414f9bf50c86c2d7235da",4)&&ok;
28
29          key=new byte[20];
30          for(int i=0;i<20;i++){key[i]=0x0c;}
31          ok= testCase(key, stringToByteArray("Test With Truncation"),
32                  "4c1a03424b55e07fe7f27be1d58bb9324a9a5a04",5)&&ok;
```

4.2 Testovanie aplikácie MobileGrid na mobilných telefónoch

Aplikácia MobileGrid bola úspešne otestovaná na nasledovných telefónoch:

`Nokia 9300`
`Ericsson W200`
`Ericsson T610`

Na Nokiu 9300 bola inštalovaná pomocou dátového kábla, na oba Ericssony bola prenesená pomocou bluetooth.

Kapitola 5

Záver

Touto prácou sa podarilo realizovať systém na autentifikáciu pre poštový server pomocou mobilného telefónu. Serverová časť je určená pre platformu Linux, klient na čo najširšiu množinu mobilných telefónov.

Vďaka oddeleniu kódu pre autentifikáciu do samostatnej aplikácie ChallengeValidator sa ponúka možnosť použiť tento systém autentifikácie aj do iných webových aplikácií.

Kapitola 6

Prílohy

zdrojový kód a binárky sa nachádzaju v nasledovných adresároch:

- ChallengeValidator

  buildovanie prebieha príkazom `./build.sh` , pričom treba mať nainštalovanú knižnicu mhash, ktorá je v tomto adresári v súbore `mhash-0.9.9.tar.gz`.

- PhpMobileGrid

  pre inštaláciu viď kapitolu 2

- MobileGrid

  súbor , ktorý sa inštaluje do mobilu je `dist/MobileGrid.jar`

Literatúra

[Apa] *Apache HTTP Server*.
`http://httpd.apache.org/`.

[BAS] *Base64*.
`http://en.wikipedia.org/wiki/Base64`.

[CLD] *CLDC*.
`http://java.sun.com/products/cldc/`.

[COU] *Courier Mail Server*.
`http://http://www.courier-mta.org/`.

[Deb] *Debian Linux*.
`http://www.debian.org/`.

[JAB] *Jabber*.
`http://www.jabber.org`.

[MHA] *MHASH*.
`http://mhash.sourceforge.net/`.

[MID] *MIDP*.
`http://java.sun.com/products/midp`.

[NET] *Netbeans*.
`http://www.netbeans.org`.

[NIS02] Information Technology Laboratory NIST. *FIPS PUB 198 The Keyed-Hash Message Authentication Code(HMAC)*. 2002.
`csrc.nist.gov/publications/fips/fips198/fips-198a.pdf`.

[PHP] *PHP*.
`http://www.php.net/`.

[rfc97a] *HMAC: Keyed-Hashing for Message Authentication*. 1997.
`http://tools.ietf.org/html/rfc2104`.

[RFC97b] *rfc2202*, 1997.
`http://www.faqs.org/rfcs/rfc2202.html`.

[SM] *SquirrelMail*.
`http://www.squirrelmail.org/`.