

Slovenská technická univerzita v Bratislave

Slovenská technická univerzita v Bratislave

Ilkovičova 2, 842 16 Bratislava 4

Proxy server

Študijný program: Informatika

Ročník: 3, Krúžok: St 13:00, PU1

Autor: Martin Černák

Predmet: Konvergencia mobilných a pevných sietí

Ak. rok: 2014/15

Obsah

Zadanie.....	3
Použité vývojové prostredie.....	4
Moduly.....	5
SIP.....	5
Register.....	6
Invite.....	7
Autentizácia.....	8
HTTP Digest.....	9
Decorators.....	10
Gui.....	11
Run.....	13
Konfiguračný súbor.....	13
Zhodnotenie.....	14
Zdroje.....	15

Zadanie

Naprogramujte SIP proxy server, ktorý bude schopný obslúžiť základné SIP správy a umožní používateľom zaregistrovať sa cez SIP telefón ako aj uskutočniť hovor medzi dvoma účastníkmi. SIP správy budú obsluhované Vaším proxy serverom, t.j. žiadne správy nepôjdu priamo medzi koncovými účastníkmi.

Odporúčanie: používajte sieť bez NAT / NAPT (Network address port translation)

Aplikácia môže byť implementovaná v ľubovolnom prostredí a nesmú byť žiadne údaje vkompilované do zdrojového kódu (IP adresa, heslá, doménové mená, porty, databáza). Program musí mať konfiguračný súbor.

Hodnotiaci hárok bude v samostatnom súbore v dokumentovom serveri

BONUS: Instant messaging, SIP REGISTER správy s výpočtom bezpečného hesla, ukončenie hovoru priamo z proxy (korektne, tak aby oba telefóny ukončili hovor hneď po jeho ukončení na proxy), použitie doplnkových systémov ako: databáza (zoznam aktuálne prihlásených, lokalizácia používateľov, správa používateľov, ukladanie správ do db, bezpečné prihlásenie sa administrátora do databázy...), Mediaproxy, nahrávanie hovorov (iba RTP pakety alebo priamo do zvukového súboru), dobré GUI, zobrazenie prebiehajúcich hovorov...

Použité vývojové prostredie

Na vytvorenie tohto zadania som sa rozhodol použiť programovací jazyk python z dôvodu zo študijných dôvodov, v rámci SIP som nepoužil žiadnu knižnicu, iba štandardné knižnice programovacieho jazyka.

Na frontend som sa rozhodol použiť html5 a css3 za použitia bootstrapu a voľne dostupnej šablóny <http://startbootstrap.com/template-overviews/scrolling-nav/>. Na prepojenie frontendu a backendu som použil web.py ktorý som nainštaloval pomocou správcu balíkov python-pip.

Na kontrolu napísaného kódu som použil nástroj pylint. Riešenie sa drží štandardov pythonu až na C0326(bad-whitespace) a C0111(function-class doc) najmä z dôvodu zarovnávanie multiargumentových funkcií.

Na odsadzovanie som používal štyri medzery.

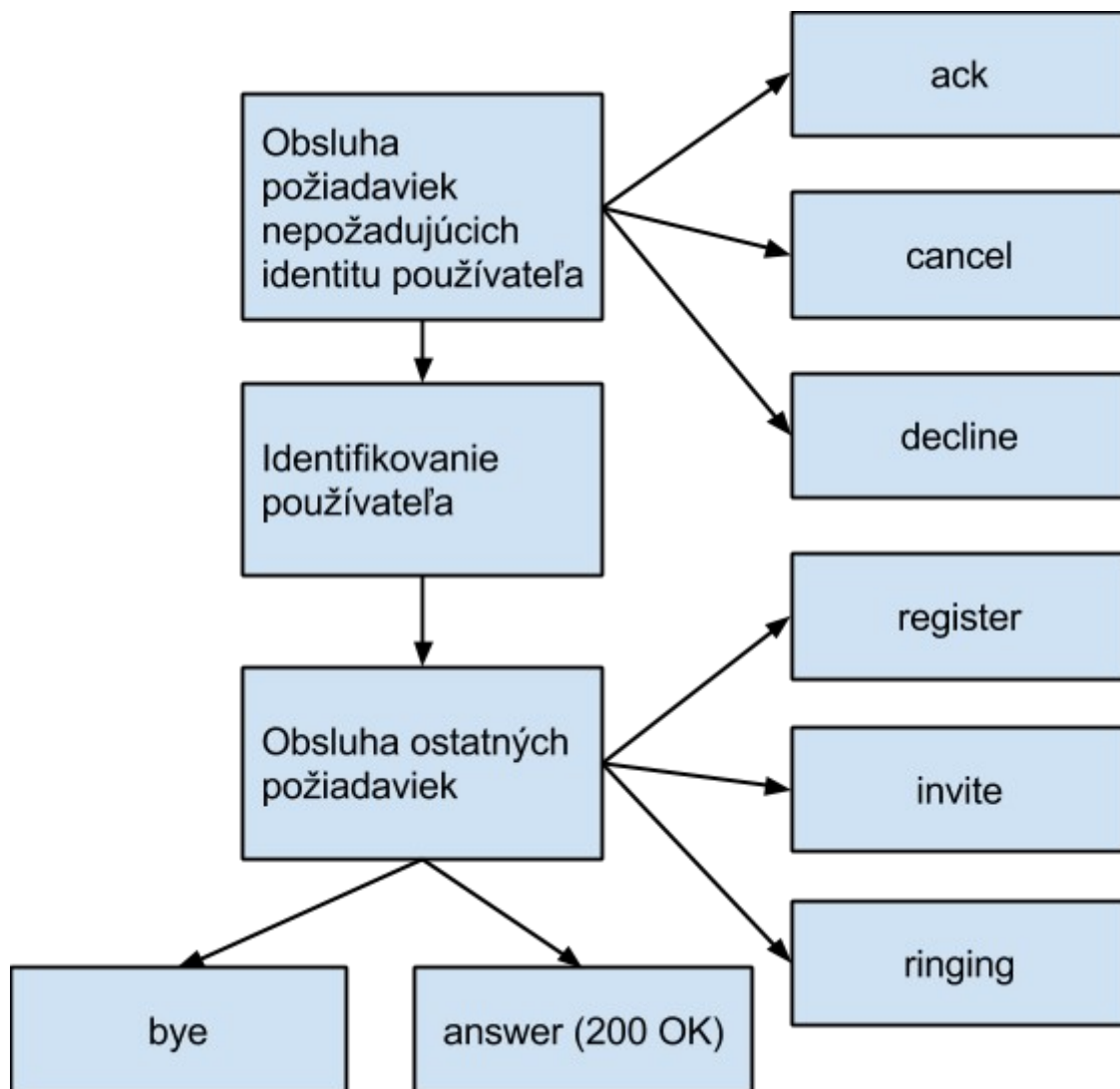
Na správu virtuálnych strojov som použil vagrant spolu z virtualbox-om, pričom vo vagrante bol nastavený server a klientov som konfiguroval osobitne.

Moduly

Aplikácia je rozdelená do niekoľkých modulov.

SIP

Modul obsahujúci implementácia Proxy servera a parsera SIP packetov, server je minimálna funkčná implementácia,



Server si neukladá informácie o transakciách ani hovoroch, každý request obsluhuje úplne samostatne. Aplikácia v podstate nevytvára vlastné packety, vždycky modifikuje packety ktoré jej prišli a preposiela ich ďalej.

Táto aplikácia je tak isto multivláknová, prijímanie packetov beží v osobitnom vlákne od ovládacieho vlákna.

Register

Registrácia v rámci svojej činnosti odstraňuje z návratových packetov hlavičky Authorization a WWW-Authenticate. Následne na základe hodnoty expires určí či sa užívateľ chce registrovať a odregistrovať.

Ak je Expires 0 tak užívateľa odregistrovuje inak zaregistruje.

V oboch prípadoch na konci v packete zmení status na 200 OK a vráti ho klientovi.

```
def registration(self, packet):
    if 'Authorization' in packet.headers:
        del packet.headers["Authorization"]
    if 'WWW-Authenticate' in packet.headers:
        del packet.headers["WWW-Authenticate"]
    if exp_pattern.match(packet.headers['Expires']):
        self.unregister(packet)
    else:
        self.register(packet)

@decorators.register
def register(self, packet):
    packet.status = "SIP/2.0/200 OK"
    self.route[packet.get_sending_client()] = packet.get_return_address()
    self.send(packet, packet.get_return_address())

@decorators.unregister
def unregister(self, packet):
    packet.status = "SIP/2.0/200 OK"
    if packet.get_sending_client() in self.route:
        del self.route[packet.get_sending_client()]
    self.send(packet, packet.get_return_address())
```

Invite

Keďže invite je preposlanie packetu ďalej, tak prvé čo kontroluje je hlavička Max_Forwards a dekrementuje ju ak je viac ako 0.

Následne kontroluje či požadovaný klient je zaregistrovaný, ak nie tak vráti 404.

Ak požadovaný klient je zaregistrovaný klientovi ktorý požaduje spojenie pošle 100 Trying, modifikuje iba status.

Ďalej prepošle invite z upraveným statusom(ak by bol chybný, pre istotu ho server znova generuje), pridaným svojím via a pridanou hlavičkou record-route.

```
@decorators.invite
def invite(self, packet):
    #---if not self.max_forwards(packet): return False
    #---target_user=packet.get_requested_client()

    #---if target_user in self.route:
    #---    packet.status = "SIP/2.0 100 Trying"
    #---    self.send(packet, packet.get_return_address())
    #---    packet.status = "INVITE sip:[0]@[1] SIP/2.0".format(
    #---        target_user, self.route[target_user][0])
    #---    packet.via.insert(0, self.get_via(packet))
    #---    packet.headers['Record-Route'] = "<sip:{0};l>".format(
    #---        self.get_address())
    #---    self.send(packet, self.route[target_user])
    #---else:
    #---    self.send_not_found(packet)
```

Autentifikácia

Autentifikácia sa vykonáva iba pre niektoré typy správ a funguje ako samostatná časť programu. V prípade že príde požiadavka na vykonanie akcie ktorá vyžaduje autentifikáciu, automaticky sa kontroluje a ak odosielateľ nieje au posiela sa mu unauthorized správa. Táto funkcia využíva funkciu is_valid z modulu http_digest.

```
@decorators.auth
def auth(self, packet):
    """if not packet.get_sending_client() in self.route:
    """if "Authorization" in packet.headers:
    """digest = http_digest.parse(packet.headers['Authorization'])
    """if not digest['username'] in self.clients:
    """self.send_unauthorized(packet)
    """return False
    """password = self.clients[digest['username']]
    """if not http_digest.is_valid(password, digest, packet.get_tr_method()):
    """self.send_unauthorized(packet)
    """return False

    """return True
    """else:
    """self.send_unauthorized(packet)
    """return False
    """else:
    """return True
```


HTTP Digest

Tento modul poskytuje funkciu `is_valid` ktorá kontroluje či klientom vrátený result je správny a funkciu `generate` ktorá vygeneruje nonce. Program používa na tvrdo priradenú jednoduchú metódu autentizácie, teda qop nieje špecifikované. Výsledný kľúč sa generuje následovne:

HA1=`MD5`(username:realm:password)

HA2=`MD5`(method:digestURI)

response=`MD5`(HA1:nonce:HA2)

Čo je v kóde:

```
@decorators.is_valid
def is_valid(password, digest, method):
    """is_valid"""
    # HA1 = MD5(username:realm:password)
    sh1 = md5.new(
        '{}:{}'.format(digest['username'], digest['realm'], password)
    ).hexdigest()
    # HA2 = MD5(method:digestURI)
    sh2 = md5.new('{}:{}'.format(method, digest['uri'])).hexdigest()
    # response = MD5(HA1:nonce:HA2)
    response = md5.new('{}:{}'.format(sh1, digest['nonce'], sh2)).hexdigest()
    return digest['response'] == response
```

Nonce je v tomto programe generovaná pomocou hashovania aktuálneho timestampu.

```
@decorators.control_message
def generate(server_name):
    """generate"""
    # nonce = MD5(timestamp)
    nonce = md5.new(str(time.time())).hexdigest()
    return {'nonce': nonce}
```

Decorators

Činnosťou tohto modulu je logovanie. Loguje celú činnosť servra a to na cli a do súboru log.txt. Na logovanie som použil štandardnú knižnicu pythonu menom logging. Logi sú napísané vo formáte:

[\$čas \$typ_správy(INFO,DEBUG,ERROR,WARNING)] : \$správa

To že mám logovanie v inom súbore ako je aktuálny kód som dosiahol postupom menom decorating functions. Pri tomto postupe v podstate vytvorím obalovaciu funkciu ktorej pošlem funkciu ktorú chcem zavolať ako parameter. Ona následne zaznamená vstup aj výstup funkcie a na základe neho pridá správy do logu.

```
def invite(function):  
    def wrapper(*args, **kwargs):  
        logger.info("User {} is inviting {} to call\n".format(  
            args[1].get_sending_client(),  
            args[1].get_requested_client()))  
        function(*args, **kwargs)  
    return wrapper
```

Gui

Tento modul zabezpečuje spoluprácu z html5 a css3 front endom, tak isto nastavuje SIP.Server a prepisuje konfiguračné súbory na základe požiadaviek používateľa. Obsahuje niekoľko obrazoviek:

1. Nastavenia server

SIP Proxy

Users

Settings

Log

Settings

Port:

5062

IP:

192.168.50.30

Server name:




test.sip.local


Save

2. Zobrazenie/Mazanie používateľov

[SIP Proxy](#) [Users](#) [Settings](#) [Log](#)

Users Section

Account	Address	Delete
111		
113		
112		



3. Pridanie používateľa


[SIP Proxy](#) [Users](#) [Settings](#) [Log](#)

Add Users

Account:

Password:

Password 2:



4. Zobrazenie logu

SIP Proxy Users Settings **Log**

Log Section

```
[2014-11-26 00:27:53,653 INFO] : Starting SIP server

[2014-11-26 00:27:53,654 INFO] : Server ip_address: 192.168.50.30
port:5062

[2014-11-26 00:27:59,415 INFO] : Receiving packet from ('192.168.50.11', 5060):
REGISTER sip:192.168.50.30:5062 SIP/2.0
Via: SIP/2.0/UDP 192.168.50.11:5060;rport;branch=z9hG4bKPj1ab52d30-6099-4397-99f2-bf02248d6089
Max-Forwards: 70
From: <sip:111@192.168.50.11>;tag=e531b133-63fd-4e22-be38-514be13d62b7
To: <sip:111@192.168.50.11>
Call-ID: 05adc886-8319-4922-96c3-1600e190e107
CSeq: 43434 REGISTER
User-Agent: SFLphone
Contact: <sip:111@192.168.50.11:5060>
Expires: 0
Route: <sip:192.168.50.11>
Content-Length: 0

[2014-11-26 00:27:59,420 INFO] :
[generate]:
args:
('test.sip.local',)
```

Refresh

Run

Tento modul je v podstate alternatívny front-end ktorý kompletne beží v cli, podporuje vypísanie klientov(príkaz clients), vypísanie registrovaných klientova(príkaz route), zastavenie a spustenie servera (príkazy start a stop) a vyčistenie obrazovky(príkaz cls). Na ukončenie programu slúži príkaz q.

Konfiguračný súbor

Na konfiguráciu sa používajú súbory vo formáte py, čiže python skript, avšak ak chce používateľ editovať konfiguračné súbory pri plnom využití jazyka python, nemôže už používať nástroje na správu konfigurácie obsiahnuté v používateľskom rozhraní.

Príklad štandardného konfiguračného súboru

```
port = 5062
ip = "192.168.50.30"
server_name = "test.sip.local"
packet_buff = 40

clients = {
    '111' : 'pass',
    '112' : 'pass',
    '113' : 'pass',
}
```

Zhodnotenie

Vytvorený Proxy server je funkčný a jeho funkcionalita spĺňa stanovené požiadavky. Systém tak isto ponúka navyše autentifikáciu pomocou http_digest.

Zdroje

http://en.wikipedia.org/wiki/Digest_access_authentication

<http://webpy.readthedocs.org/en/latest/>

<http://getbootstrap.com/>

<http://tools.ietf.org/html/rfc3261>

<http://startbootstrap.com/template-overviews/scrolling-nav/>