Administrace Unixu a sítí

```
inet6-adr: fe80::210:a4ff:fee1:9e5d/64 Rozsah:Linka
AKTIVOVÁNO VŠESMĚROVÉ_VYSÍLÁNÍ BĚŽÍ MULTICAST  MTU:1500  Metrika:1
RX packets:66690 errors:0 dropped:0 overruns:0 frame:0
TX packets:100149 errors:0 dropped:0 overruns:0 carrier:0
kolizí:0 délka odchozí fronty:0
RX bytes:21490419 (20.4 MiB)  TX bytes:10545763 (10.0 MiB)
```

8. WWW, HTTP

Miroslav Spousta

2006

```
bug:/home/qiq# getent passwd|grep 10
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
Debian-exim:x:102:102::/var/spool/exim4:/bin/false
qiq:x:1000:1000:Miroslav Spousta,,,:/home/qiq:/bin/bash
sshd:x:100:65534::/var/run/sshd:/bin/false
identd:x:101:65534::/var/run/identd:/bin/false
messagebus:x:103:104::/var/run/dbus:/bin/false
gdm:x:104:105:Gnome Display Manager:/var/lib/gdm:/bin/false
hal:x:106:106:Hardware abstraction layer,,,:/var/run/hal:/bin/false
saned:x:109:109::/home/saned:/bin/false
bind:x:105:110::/var/cache/bind:/bin/false
smmta:x:107:111:Mail Transfer Agent,,,:/var/lib/sendmail:/bin/false
smmsp:x:108:112:Mail Submission Program,,,:/var/lib/sendmail:/bin/false
test:x:1001:1001:Test User,,,:/home/test:/bin/bash
postfix:x:110:115::/var/spool/postfix:/bin/false
```

http://www.ucw.cz/~qiq/vsfs/

Xen

- virtuální servery (http://www.cl.cam.ac.uk/Research/SRG/netos/xen/)
- budete mít rootovská oprávnění ve virtuálním serveru
- přístup je po dobu výuky
- OS: Debian 3.1
- RAM: 32 MB, swap: 128 MB (/dev/sda2), root: 512 MB (/dev/sda1)
- několik síťových karet (eth0, eth1, ...)
- Síťové karty jsou propojeny virtuálními přepínači

Přístup k virtuálním serveru:

- **`ssh login@kozel.vsfs.cz`**
- **`xencons localhost 90xx`**
- xx je číslo serveru (login: **`root`**, heslo: žádné)

HTTP

- HyperText Transport Protocol, RFC 2616
- v současné době se používá verze 1.1 a 1.0
- klient-server, server odpovídá na požadavky klienta
- stejně jako u SMTP se jedná o jednoduchý textový protokol
  - chybové kódy podobně jako u SMTP (tři číslice)
  - 2xx: bez chyby: 200 OK, 201 Created, 202 Accepted, ...
  - 3xx: přesměrování klienta: 301Moved Permanently, 302 Found, 304 Not modified
  - 4xx: chyba u klienta: 400 Bad request, 403 Forbidden, 404 Not found
  - 5xx: chyba na serveru
- používá protokol TCP, port 80
- funguje bezestavově
- podporuje cachování odpovědí (podobně jako DNS)

HTTP metody

- klient posílá požadavky pomocí *metody*
  - GET, POST, PUT, DELETE, OPTIONS, CONNECT, ...
  - formát: metoda cesta protokol
  - následují hlavičky (podobné jako u SMTP)
  - prázdný řádek (CRLF)
  - případná data
- GET: stáhnout obsah URL
- HEAD: pouze hlavičky
- POST, PUT: nahrát data na server
- CONNECT: SSL a proxy server
- OPTIONS: povolené metody

```
C: GET / HTTP/1.0
C:
S: HTTP/1.1 200 OK
S: Date: Wed, 11 May 2005 ...
S: Connection: close
S: Content-Type: text/html; ...
S:
<!DOCTYPE...
```

stáhněte si stránku z některého serveru

- *pomocí příkazu netcat*

wget

- nástroj pro stahování stránek z webu
- umí autentizaci, cookies, rekurzi, ssl, ...
- **wget *url***
  - **-S** zobrazí hlavičku odpovědi
  - **-d** zobrazí debugovací informace, mj. také hlavičku požadavku :-)
  - **-r** stahuje data rekurzivně, až do úrovně **-l *level***
  - **--save-cookies *file*, --load-cookies *file***
  - **--http-user *user*, --http-passwd *pass***

```
wget -S http://atrey.karlin.mff.cuni.cz/
wget -d http://idnes.cz/
```

vyzkoušejte si wget (opět stáhněte stránku z některého serveru včetně hlaviček)

HTTP hlavičky klienta

- Accept, Accept-charset, Accept-encoding, Accept-language
  - preference klienta ohledně obsahu, kódování, přenosu, jazyka...
  - např. Accept-language: cz; q=0.8, en-gb; q=0.7, en; q=0.5
- Referer, User-Agent
  - odkaz na URL, ze kterého požadavek pochází
  - verze klienta (např. User-Agent: Mozilla/5.0), pro statistické účely
- If-Modified-Since
  - následuje datum (v GMT)
  - podmíněný požadavek, vrací stránku pouze pokud byla změněna od uvedeného data
  - jinak vrátí 304 Not modified
- Content-Type, Content-Length, Transfer-encoding
  - jako v SMTP poště, velikost dat
- Pragma: no-cache

HTTP cookies

- HTTP je navržený jako bezestavový protokol
- je potřeba na serveru rozpoznat požadavky jednoho klienta
  - obchod, osobní nastavení, ...
- mechanismus cookies, RFC 2965
  - server pošle hlavičku Set-cookie (Set-cookie2):
    - může jich být i víc
  - klient při příštím přístupu na stejné (nebo specifičtější) URL pošle hlavičku Cookie (Cookie2):
    - mohou se řetězit
- pozor, klient může cookie měnit!

vyzkoušejte si cookies pomocí wget:

- **`wget -d --save-cookies .cook http://www.seminar/cookies.php`**
- **`wget -d --load-cookies .cook http://www.seminar/cookies.php`**
- skript cookies.php nastaví cookie a při příští návštěvě ji zobrazí

Web server Apache

- nejrozšířenější web server (více než 60% serverů v Internetu)
  - aktuální verze: 2.0 (2.2)
- jednoduchá konfigurace
  - nastavení konfigurace: /etc/apache
  - strom web serveru: /var/www
  - logy: /var/log/apache
- spuštění: /etc/init.d/apache2 start

vytvořte si html soubor v adresáři /var/www a ověřte, že lokální web server funguje

- *wget http://localhost/test.html*
- *wget http://www.vmXX.aus/test.html*
- *případně použijte links*

HTTP autentizace

- omezení přístupu k URL na základě jména a hesla
- standardní mechanismus HTTP, RFC 2617
- při přístupu k zabezpečenému objektu vygeneruje server odpověď
  - 401 Unauthorized
  - do hlavičky odpovědi přidá položku WWW-Authenticate: Basic realm="xxx"
- klient se zeptá uživatele na jméno a heslo
  - vytvoří řetězec „jmeno:heslo"
  - ten zakóduje pomocí uuencode algoritmu a odešle v hlavičce požadavku
  - Authorization: Basic aWFuOmJvb2J5
- realm slouží k tomu, aby klient rozlišil, které heslo má poslat
- kromě Basic autentizace existují i vylepšená varianta, kde se neposílá plain-textové (resp. uuencodované) heslo, ale jeho MD5
  - Digest autentizace, může používat různé seedy

HTTP autentizace: nastavení

- někde v adresáři web serveru (/var/www) si vytvořte podadresář, do kterého budete omezovat přístup
- v něm vytvořte soubor, který bude sloužit pro ověřování uživatelů
  - **`htpasswd -c /path/to/file username`**
- v tomtéž adresáři vytvořte soubor .htaccess (pozor na tečku na začátku!), který bude řídit přístup do daného adresáře

```
AuthType Basic
AuthName "Test (realm)"
AuthUserFile /path/to/file
Require valid-user
```

  - říkáme že chceme autentizovat uživatele, basic authentication, cestu k souboru a že vyžadujeme, aby uživatel byl uveden v daném souboru

nastavte a vyzkoušejte si autentizaci:

  - wget -d –-http-user *username* -–http-passwd *passwd* http://*server/path*

Virtuální web servery

- dva základní přístupy: IP based a name-based
  - IP based vyžaduje pro každý virtuální server samostatnou IP adresu
  - name based – rozlišují požadavky podle obsahu hlavičky Host:
- vyzkoušíme si natavení name-based web serveru
- v DNS jsou dva záznamy:

```
www     IN  A       10.0.0.1xx
www2    IN  CNAME   www
```

- nastavení virtuálního serveru
  - /etc/apache2/sites-enabled/test-config:
- start Apache:
  - /etc/init.d/apache2 start

```
Listen 10.0.0.1XX:80
NameVirtualHost 10.0.0.1XX:80
<VirtualHost 10.0.0.1XX:80>
ServerName www.vmXX.aus
DocumentRoot /var/www/www1
</VirtualHost>
# to samé pro www2
```

nastavte a vyzkoušejte si virtuální servery
- *links http://www.vmXX.aus*
- *links http://www2.vmxx.aus*

HTTPS

- používá protokol SSL, běží na portu 443
  - podle RFC 2818 by mohlo běžet i HTTP nad TLS
- podobné jako POP3, IMAP; Apache podporuje SSL pomocí modulu

do souboru /etc/apache2/sites-enabled/test-config přidejte definici dalšího virtual hostu:

```
NameVirtualHost 10.0.0.1XX:443
Listen 10.0.0.1XX:443
<VirtualHost 10.0.0.1XX:443>
ServerName www.vmXX.aus
DocumentRoot /var/www/www1
SSLEngine On
SSLCertificateFile /etc/apache2/ssl/apache.pem
</VirtualHost>
```

- před spuštěním ještě potřebujeme získat certifikát

Certifikáty

- certifikační autorita (bezpečné) / self-signed certifikáty (snazší)
- vygenerujeme žádost o certifikát (commonName je FQDN)
  - `openssl req -new -nodes -out cert.csr -keyout cert.key`
  - vygenerovali jsme žádost a klíč, můžeme si je zobrazit:
  - `openssl req -text -in cert.csr | less`
  - `openssl rsa -test -in cert.key | less`
- sami si certifikát podepíšeme (self-signed):
  - `openssl x509 -req -in cert.csr -signkey cert.key -out cert.crt`
  - `openssl x509 -text -in cert.crt`
- zkopírujeme certifikát a jeho klíč
  - `cat cert.crt cert.key >/etc/apache2/ssl/apache.pem`
- restartujeme Apache: /etc/init.d/apache2 restart

- můžete vyzkoušet pomocí openssl:
  - `openssl s_client -connect 10.0.0.1XX:443`

Certifikační autorita

- pokud nechceme používat self-signed certifikát, můžeme:
  - nechat si žádost podepsat od veřejné CA (Verisign, 1. CA, …)
  - vytvořit si vlastní CA (ale ta bude opět self-signed) a pomocí ní podepisovat žádosti
- vlastní CA může být vhodná, pokud máme více certifikátů pro různé služby
- lze např. pomocí skriptů v openssl:
  - vytvoříme demo-CA: `cd /usr/lib/ssl; misc/CA.sh -newca`
  - zkopírujeme žádost: `cp cert.csr newreq.pem`
  - podepíšeme žádost: `misc/CA.sh -sign`
  - výsledek je v newcert.pem, zkopírujeme do adresáře /etc/apache2/ssl/:
  - `cat cert.key newcert.pem >/etc/apache2/ssl/apache.pem`

- můžeme vyzkoušet opět pomocí openssl:
  - `openssl s_client -connect 10.0.0.1XX:443`