Operációs rendszerek alapjai

7. gyakorlat

Mai gyakorlat

- Terminál-internet interakció

Security jelentése

- Fontos elkülöníteni a biztonság fogalmakat
- Ahogy az ssh (secure shell) esetében is láttuk, a korábbi rsh (remote shell) használatakor is lehet szabályozni a hozzáférést például jelszavakkal
    - tehát alapvetően aki nem ismer egy létező user-password párost, az nem tud belépni
    - azonban a kommunikáció nyíltan történik, tehát lehallgatható
- Secure változat esetén a bejelentkezést követően a kommunikáció is titkosított, tehát nem lehallgatható (pontosabban lehallgatható, csak értelmezhetetlen)
- Eredeti változatok gyakran nem is használhatóak, például rsh és rcp igazából egy szimbolikus link ssh-ra és scp-re

scp

- Secure copy protocol
  - rcp (remote copy) secure változata
  - ssh-t használ
  - `scp files target`
    - kapcsolók a hagyományos `cp`-hez hasonlóan, például `-r` kapcsolóval rekurzívan másol
- A protokollhoz nem tartozik RFC
- Manapság elavultnak és rugalmatlannak tartják, komoly használathoz sftp-t ajánlanak

sftp

- ftp (File Transfer Protocol) secure változata
  - másik biztonságos változata az ftps, ami a http-https pároshoz hasonlóal SSL/TLS-t is használ
- Létrehoz egy kapcsolatot, így tudunk navigálni a szerver fájlrendszerében is a szokásos parancsokkal (`ls`, `cd`) és a lokális fájlrendszerben is (`lls`, `lcd`)
- `put` és `get` parancsokkal fel- és letölthetünk fájlokat

G07F01

- Próbáld ki az sftp parancsot valamelyik szerveren!

wget

- Fájlok letöltésére alkalmas HTTP(S) és FTP(S) protokollok használatával
- Nem interaktív, kifejezetten alkalmas script-eken belüli használatra
- Rekurzív letöltésre, linkek követésére és időbélyegek figyelésére is alkalmas
- `wget [options] [URLs]`
  - URL-ben megadhatjuk a portot, illetve felhasználói adatokat is

    `http://host[:port]/directory/file` `http://user:password@host/path`

    `ftp://host[:port]/directory/file` `ftp://user:password@host/path`
- Logot standard outputra írja alapértelmezetten

wget kapcsolók

- `-b`/`--background`: háttérben indítja el, log a wget-log fájlba kerül (ha nem adunk meg mást)
- `-o`/`-a logfile`: megadott fájlba írja vagy fűzi a logot
- `-q`/`-v`/`-nv`: quite, verbose vagy a kettő közötti mód
- `-i file`: megadott fájlból olvassa a címeket
- `-t number`: próbálkozások számának megadása, alapból 20 (ha nincs súlyos hiba)
- `-c`/`--continue`: folytatja a letöltést
- `-P prefix`: mentés helyének megadása, alapértelmezetten `.`
- `-r -l5`: rekurzív letöltés bekapcsolása és mélység megadása (alapértelmezetten öt)
- `-O filename`: letöltött fájlokat konkatenálja és elmenti a megadott fájlba
- `--post-data 'key=value'`: kulcs-érték párokkal adat megadása
- `--post-file filename`

G07F02

- Töltsd le az alábbi közzétett Google táblázatot wget segítségével!
  - `wget -O filename.csv "link"`
  - (közzétenni a Fájl > Közzététel az interneten menüben lehet)

G07F03

- Töltsd ki az alábbi Google formot wget segítségével!
- A weboldal kódjában keresd meg a megfelelő mezők nevét, majd használd ezeket kulcsként a `--post-data` argumentumában!

G07F04

- DOI alapján tölts le egy cikket a sci-hub.do oldalról!
  - példa DOI: 10.1016/j.ifacol.2019.07.008
- Először próbáld ki az oldalt és döntsd el, hogyan lehetne kiszűrni a pdf fájlt!
- Az eddigi funkciókat használva igyekezz minél kevesebb sorból megoldani a problémát!
  - irányadó: one-liner

API hívások

- Egyszerűbb API hívásokat is végezhetünk wget segítségével
  - HTTP status code-oknak megfelelő cicás képek: http.cat
  - időjárás: wttr.in
    - többféle formátum van, például ez
- Akár lehet sütiket is elmenteni és betölteni
- A probléma, hogy bonyolultabb esetben JSON-ben szokás kommunikálni API-val, amit nehéz bash-ben kezelni
  - feltöltést még csak-csak össze lehet hozni, de bash-ben nem léteznek például beágyazott dict-ek
  - egyéb scriptnyelvekben (például Python, Perl, Ruby) ez egy kényelmes feladat

cURL

- wget-hez hasonló nem interaktív program
- Van mögötte egy könyvtár (libcurl), így többféle platformon és nyelven használható
- Többféle protokollt ismert
- Rugalmasabb feltöltés (nem csak HTTP POST kérések)
- Tömöríteni és kitömöríteni is tud

rsync

- Lokális és remote fájlmásolásra alkalmas, nagyon rugalmas, hatékony és robosztus
- Alapértelmezetten is csak a változott fájlokat másolja, a működésnek szinte minden aspektusa állítható
- Képes
  - megőrizni a tulajdonost, időbélyegeket stb.
  - megtartani a fogadó frissebb fájljait
  - backup-t készíteni
  - tömöríteni másolás közben
- Példa: Linux-on az egyik legjobban használható cloud szolgáltatás a Dropbox
  - telepítés után megadott mappába (alapértelmezetten ~/Dropbox) tölti le a fájlokat
  - daemon folyamatosan figyeli a változásokat, le- és feltölti az új fájlokat
  - rsync segítségével kifejezetten kényelmes

Egyéb parancsok

- `who` / `w` - bejelentkezett felhasználók listázása
  - többszöri bejelentkezést is jelez
- `finger` - alapvető információk az adott felhasználról
  - tempus-on nincs
- `write` - üzenet írása másik felhasználónak
  - `write vagmi` után megnyílik a csatorna és küldheted az üzeneteket
  - `write vagmi ttyname` paranccsal többször bejelentkezett hallgatónak adott terminájára írsz
  - lehet pipe-olni is fájlt
- `wall` - üzenet írása minden felhasználónak
  - tempus-on nincs
- `mesg` - adott terminálra való írás engedélyezése vagy blokkolása

Egyéb parancsok

- `find` - search for files in a directory hierarchy
  - rekurzívan keres adott tulajdonságnak megfelelő objektumot
  - gyakorlatilag az inode bármilyen tulajdonságra lehet szűrni
  - `-type b/c/d/p/f/l/s` kapcsolóval block / karakter / mappa / pipe / fájl / symlink / socket keresése
  - `-name regex` kapcsolóval pattern-nek megfelelő fájlokat keres
  - `-exec command {} ;`
    - a megtalált fájlokra végrehajtja az adott parancsot, ahol a fájlt a `{}` helyére helyettesíti be
    - ha parancssorból hívjuk meg, akkor `\;`-el kell lezárni
  - `-exec command {} +`
    - a megtalált fájlokat összefűzi és végrehajtja az adott parancsot, ahol a fájlt a `{}` helyére helyettesíti be
    - ha parancssorból hívjuk meg, akkor `\+`-el kell lezárni

Folyamatok

- Adott session során rengeteg folyamat fut
- Eddig főleg előtérben futtattunk, de láttuk már, hogy az & jellel futtathatunk háttérben is
- Minden folyamat rendelkezik egy egyedi azonosítóval, ez a PID
  - amikor háttérben futtatsz egy parancsok, akkor kiírja a PID-jét
- `top` / `htop` parancsokkal interaktívan listázhatjuk az összes futó parancsot
- `ps` paranccsal az aktuális folyamatokról kapunk egy snapshot-t
  - `ps aux` - kiírja az összes felhasználó tty nélküli folyamatait is, felhasználók vannak az első oszlopban
- `pstree -t` - részletesen listázza a folyamatfát

Folyamatok lezárása

- Szükségünk van a folyamat PID-jére
  - `ps aux | grep process`
  - `pgrep process`
  - `pidof process`
- `kill PID`
- Ha nincs szükség ennyire precíz lezárásra, akkor
  - `pkill process`
  - `killall process`
- `xkill` paranccsal grafikusan választhatunk a lezárni kívánt ablakok közül

Szálkezelés

- `jobs` paranccsal listázhatjuk a leszármazott folyamatokat
- `wait` paranccsal várhatunk az összes leszármazott folyamatra, vagy a felsorolt PID-ekre

Következő gyakorlat

- Feladatok ütemezése
- Egyéb fontos parancsok

Érdekességek, olvasnivalók

- Selenium
  - Rövid script tárgyfelvételhez
- BeautifulSoup