Robert Meolic
meolic@uni-mb.si
zadnja sprememba: 25. 02. 2009

Zapiski predavanj 3

interno gradivo za predmet VSO, 2008/09

16. Oddaljena prijava preko interneta

V preteklosti je bil najbolj pogost način oddaljene prijave preko interneta storitev telnet. Ker pri telnetu ni poskrbljeno za ustrezno zaščito podatkov, je uporaba telneta že nekaj časa odsvetovana.

Na sistemih Unix se je v preteklosti uporabljala (in se še uporablja) storitev rsh. Tudi pri rsh ni ustrezno poskrbljeno za varnost, zato se je moramo izogibati.

Varen način oddaljene prijave je storitev **ssh** (Secure Shell). V začetku je bil na voljo le kot komercialni izdelek podjetja SSH Secure Communications (avtor je pravzaprav najprej napisal freeware, nato pa je ustanovil podjetje in nove verzija izdelka izdal kot komercialni produkt). Danes sta za Linux na voljo prosta paketa **OpenSSH** in **lsh**. Za Microsoft Windows so na voljo komercialne rešitve, npr. **SSH Tectia Server**, **WinSSHD**, **Vshell Server**, pa tudi prosti programi, npr.: **freeSSHd**, **MobaSSH** in **KpyM SSH Server**. Če potrebujemo samo odjemalca za Microsoft Windows, sta zelo dobra izbira prosta programa **PuTTY** in **Tera Term**.

Oglejmo si, kako ssh uporabljamo iz terminala na Linuxu. Na oddaljen računalnik `vso.ms.si` se prijavimo na naslednji način:

```
ssh vso.ms.si
```

Če se želimo prijaviti z drugačnim uporabniškim imenom, kot ga imamo na lokalnem računalniku, uporabimo kretnico `-l`, ki ji (običajno brez presledka) sledi uporabniško ime (npr. `Uporabnik`):

```
ssh -lUporabnik vso.ms.si
```

Ko se prijavimo preko ssh lahko na daljavo prenašamo tudi ukaze XWindows, kar pomeni, da lahko poganjamo tudi grafične programe. V praksi je sicer pogosto nastavljeno tako, da moramo grafiko eksplicitno omogočiti kar storimo s kretnico `-X` (starejši način) ali `-Y` (bolj varen način).

Če želimo izvesti le en ukaz (npr. zagnati nek določen program) na oddaljenem računalniku, lahko ta ukaz navedemo kot argument.

```
ssh -Y vso.ms.si firefox
```

Strežnik za ssh pri prijavi seveda zahteva geslo na ciljnem sistemu, čemur pa se lahko izognemo. Če generiramo asimetrična ključa (RSA ali DSA) in javnega namestimo na strežniku v datoteki `~/.ssh/authorized_keys` , privatnega pa shranimo pri sebi v mapi `~/.ssh/`, potem nas strežnik ne bo vprašal za geslo, saj nam bo zaradi privatnega ključa zaupal.

Za kreiranje ključa in prenos javnega dela na strežnik na Linuxu uporabimo naslednja dva ukaza (več o ključih in šifriranju v enem od naslednjih poglavij):

```
ssh-keygen -t rsa
ssh-copy-id -i ~/.ssh/id_rsa.pub Uporabnik@vso.ms.si
```

17. Prenos datotek preko interneta

V preteklosti je bil najbolj pogost način prenašanja datotek preko interneta storitev ftp. Ker pri ftp-ju ni poskrbljeno za ustrezno zaščito podatkov, se danes ftp uporablja le za tako imenovani anonimni prenos datotek, ki ne vsebujejo zaupnih podatkov.

Na sistemih Unix se je v preteklosti uporabljala (in se še uporablja) storitev rcp. Tudi pri rcp ni ustrezno poskrbljeno za varnost, zato se je moramo izogibati.

Varen način prenosa datotek preko interneta so storitve **sftp** (Secure FTP), **scp** (Secure Copy) in **sshfs** *(Secure Shell Filesystem)*. Vse tri storitve na strežnikovi strani uporabljajo ssh. Na odjemalčevi strani, če je tam Linux, je za sftp in scp dovolj, da namestimo odjemalca ssh, za sshfs pa moramo odjemalca namestiti posebej. Na operacijskem sistemu Microsoft Windows je na voljo program **WinSCP**, ki vsebuje odjemalca za sftp in scp. sshfs je tam zaenkrat še bolj slabo podprt, na voljo sta komercialna **SftpDrive** in **WebDrive** ter zaenkrat še nekoliko nezanesljiv prost program **Dokan**.

Uporaba **sftp** je podobna uporabi ftp. Uporabljamo ukaze dir, cd, put, get, mput, mget itd.

Uporaba scp je podobna ukazu cp. Datoteko `Vaje.pdf` prenesemo iz računalnika `vss.ms.si` k sebi z naslednjim ukazom:

```
scp vso.ms.si:imenik/Vaje.pdf .
```

Obratno, torej prenos na oddaljeni računalnik, pa izvedemo na naslednji način:

```
scp Vaje.pdf vso.ms.si:imenik/podimenik
```

Če se želimo datoteko prenesti iz uporabniškega računa z drugačnim uporabniškim imenom, kot ga imamo na lokalnem računalniku, potem uporabimo ukaz:

```
scp Uporabnik@vso.ms.si:imenik/Vaje.pdf .
```

Podobno, kot pri ssh, lahko tudi tukaj kreiramo ključe, drugače nas strežnik pri vsakem prenosu vpraša za geslo.

Sshfs nam omogoča, da oddaljeni računalnik priklopimo neposredno v lokalni datotečni sistem (v obstoječo prazno mapo). Na primer, priklopimo svojo domačo mapo na računalniku `vso.ms.si` v mapo VSS:

```
mkdir ./VSS
sshfs vso.ms.si: ./VSS
```

Sedaj lahko datoteke premikamo in kopiramo z navadnima mv in cp, uporabimo lahko katerikoli grafični upravljalnik ali pa datoteke z oddaljenega računalnika neposredno urejamo kot da bi bile lokalne. Ko ne potrebujemo več dostopa do oddaljenega računalnika, izvedemo ukaz:

```
fusermount -u ./VSS
```

18. Deljenje datotek v omrežju

Pri deljenju datotek omogočimo drugim uporabnikom v omrežju (lokalno ali pa kar preko interneta), da dostopajo do datotek shranjenih na lokalnem disku. Oddaljeni uporabniki lahko datoteke tudi spreminjajo.

Pogoj za deljenje datotek je vzpostavljena **omrežna povezava**. Omrežno povezavo lahko vzpostavimo na več načinov, a se danes skoraj izključno uporabljajo le **internetne povezave,** ki jih vzpostavimo s pomočjo protokolov **TCP/IP** ali **UDP/IP** (IP je omrežni protokol, TCP in UDP pa sta transportna protokola, za vzpostavitev omrežne povezave je potreben IP in eden od transportnih protokolov). Naj omenimo še dve drugi možnosti. V omrežjih **Netware** (ta omrežja so danes skoraj izumrla, bila so podobna današnjim intranetom, razvijalo jih je podjetje Novell) se je omrežna povezava tvorila z uporabo protokolov **IPX/SPX** (IPX je omrežni protokol podoben IP-ju, SPX pa transportni protokol podoben TCP-ju). V omrežjih **Microsoft Windows Network** se povezava tvori s pomočjo protokola **NetBIOS**. Za razliko od prej omenjenih pa to ni omrežni ali transportni protokol, ampak sejni protokol (peti sloj OSI modela). NetBIOS je na nižjih slojih originalno uporabljal protokol NetBEUI, ki je hkrati omrežni in transportni protokol, a precej manj zmogljiv od npr. kombinacije TCP/IP. Microsoft je pozneje NetBEUI kombiniral skupaj s TCP/IP in to kombinacijo imenoval NBT.

Ko imamo omrežno povezavo, potrebujemo še aplikacijski protokol, s pomočjo katerega se bodo računalniki dogovorili o tem, katere datoteke so deljene in kakšna je njihova zaščita. Poznamo več takih protokolov:

- **NFS** (Network File System) so naredili v podjetju Sun, danes pa skrbi za njega organizacija IETF. Deluje preko internetne povezave, uporabljamo pa ga lahko na sistemih MS Windows, Linux in drugih.

- **SMB** (Server Message Block) je Microsoftov protocol, ki se primarno uporablja na sistemih MS Windows. Deluje preko protokola NetBIOS. Ker pri deljenju datotek kompleksni sejni sloj, kot je NetBIOS, pravzaprav ni nujno potreben, so pri Microsoftu v sistem Windows 2000 in novejše vgradili različico protokola SMB, ki deluje neposredno nad internetno povezavo. Danes je protokol SMB dobro podprt tudi na sistemih Linux, orodje za delo z njim se imenuje

Samba. Microsoft je konec 90-ih let želel protokol SMB nadgraditi v zmogljivejši protokol z imenom CIFS (Common Internet File System), vendar pa tega projekta niso dokončali, tako da je bil rezultat le novo ime in nekaj dodatne funkcionalnosti. Samba podpira tudi vso novo funkcionalnost iz projekta CIFS.

- **NCP** (NetWare Core Protocol) so naredili v podjetju Novell. Deluje s povezavami TCP/IP in tudi IPX/SPX. Uporabljamo ga lahko na sistemih MS Windows, Linux in drugih.

- **OpenAFS** je različica protokola AFS (Andrew file system), ki so si ga izmislili na univerzi CMU v ZDA, nato pa ga je razvijalo podjetje IBM. Od leta 2000 je to odprto-kodni projekt. Deluje preko internetne povezave in je podoben NFS. Tudi OpenAFS lahko uporabljamo na različnih sistemih, vključno z MS Windows in Linuxom.

- **AFP** (Apple Filing Protocol) so naredili v podjetju Apple. Uporablja internetne povezave in deluje v sistemu Mac OS.

- **WebDAV** (Web-based Distributed Authoring and Versioning) je zanimiv protokol, ki omogoča deljenje datotek preko vzpostavljene HTTP povezave. Razvija ga združenje IETF.

Aplikacijski protokoli za deljenje datotek (pravzaprav spadajo v šesti sloj OSI modela, to je predstavitveni sloj) so tipa **strežnik – odjemalec**. Torej moramo na tistem računalniku, na katerem se nahajajo datoteke, namestiti ustrezni strežnik. Za dostop do datotek ne potrebujemo strežnika, dovolj je le odjemalec.

18. 1. Deljenje datotek preko interneta po protokolu NFS (Network File System)

Deljenje datotek po protokolu NFS pogosto najdemo na komercialnih različicah Unix-a, npr. na Solarisu. Na računalniku, na katerem želimo dati datoteke v skupno rabo preko interneta, moramo namestiti strežnik za NFS.

Strežniki in odjemalci za NFS so na sistemih MS Windows manj razširjeni. S strani Microsofta lahko dobimo paket **Windows Services for UNIX**, ki vključuje NFS strežnik. Ta paket ne podpira sistema Windows XP Home Edition, je pa vključen v sistem MS Windows Vista. Na internetu lahko izbrskamo tudi nekatere komercialne produkte, npr. Allegro NFS Server, Omni NFS Server itd. Vsi našteti paketi vsebujejo tudi odjemalce. Med odjemalci je zanimiv tudi komercialni izdelek Reflection NFS Client.

Na sistemih Linux je protokol NFS dobro podprt in dokaj razširjen. Ko namestimo strežnik, imenike, ki so deljeni, zapišemo v datoteko `/etc/exports`. Seveda lahko to stori le administrator sistema.

Tukaj je preprost primer:

```
$ cat /etc/exports
/home/meolic/test               meolic.uni-mb-si(rw,sync)
/home/meolic/test1              meolic.uni-mb-si(ro,sync)
/home/meolic/test2              meolic(rw,sync) altair.uni-mb.si(ro,sync)
```

V vsaki vrstici podamo podatke o enem imeniku. Najprej napišemo celotno ime imenika, nato pa naštejemo računalnike, ki jim dovolimo dostop. Če naštejemo več računalnikov, jih ločimo s presledki. Za vsakega v oklepaju podamo ustrezne parametre. Nastavimo lahko številne parametre, najlažje je, če si pomagamo s kakšnim grafičnim vmesnikom.

Če smo datoteko /etc/exports tvorili brez grafičnega orodja, moramo uporabiti tudi ukaz /usr/sbin/exportfs, ki prebere datoteko /etc/exports in omogoči dostop navedenim računalnikom do navedenih imenikov.

Na računalniku, kjer je strežnik NFS, lahko spremljamo podatke o tem, kdo vse je trenutno priklopljen na deljene imenike. Podatki so zapisani v datoteki /var/lib/nfs/rmtab. Na primer:

```
$ cat /var/lib/nfs/rmtab
164.8.22.109:altair.uni-mb.si:0x00000002
altair.uni-mb.si:/home/meolic/test2:0x00000003
```

Če želimo do deljenih datotek dostopati iz Linuxa, uporabimo ukaz mount. Priklop je torej podoben priklopu lokalnih particij, le da pri kretnici -t, ki določa tip datotečnega sistema, napišemo nfs. Oblika ukaza je naslednja:

```
mount -t nfs racunalnik:/imenik/podimenik oddaljen/imenik
```

Primer:

```
mount -t nfs altair.uni-mb.si:/home/meolic/test2 mojtest
```

Podobno, kot pri priklopu lokalnih particij, lahko v datoteko /etc/fstab napišemo ustrezno vrstico in potem se bo oddaljeni imenik samodejno priklopil.

18.2. Deljenje datotek preko interneta po protokolu SMB (Server Message Block)

SMB je protokol, ki ga za deljenje datotek uporablja operacijski sistem MS Windows. Strežnik in odjemalec za SMB sta vgrajena v vse sisteme MS Windows. Imenike damo v skupno rabo s pomočjo raziskovalca ali pa iz lupine Windows command z ukazom NET SHARE. Imenik priklopimo prav tako s pomočjo raziskovalca ali pa iz lupine Windows command z ukazom NET USE.
Na Linuxu je protokol SMB podprt s paketom Samba. Uporaba Sambe je podobna uporabi NFS, le da ima več funkcionalnosti.

Imenike, ki so deljeni, zapišemo v datoteko /etc/samba/smb.conf (na nekaterih sistemih se uporablja /etc/smb.conf). Seveda lahko to stori le administrator sistema. Vsakemu imeniku moramo damo oznako. Ker lahko nastavimo številne parametre, je tudi tukaj najbolje, če si pomagamo s kakšnim grafičnim orodjem.

V nadaljevanju si oglejmo primer datoteke, ki jo je tvorilo grafično orodje, ko smo imenika /home/meolic/test in /home/meolic/test1 dali v skupno rabo pod oznako test oz. test1.

```
$ cat /etc/samba/smb.conf

... najprej je dosti komentarjev, ki razlagajo posamezne opcije ...

[test]
        comment = to je test
        path = /home/meolic/test
;       writeable = no
        browseable = yes
        guest ok = yes

[test1]
        comment = to je test1
        path = /home/meolic/test1
        writeable = yes
        browseable = yes
        guest ok = yes
```

Na računalniku z operacijskim sistemom Linux, na katerem želimo dostopati do deljenih datotek, uporabimo ukaz mount. Priklop je torej podoben priklopu lokalnih particij, le da pri kretnici `-t`, ki določa tip datotečnega sistema, napišemo smbfs. Oblika ukaza je naslednja:

```
mount -t smbfs //racunalnik/oznaka oddaljen/imenik
```

Primer:

```
mount -t smbfs //vso.ms.si/test mojtest
```

Tudi v tem primeru, lahko vrstico shranimo v datoteko `/etc/fstab`.

Na Linuxu imamo običajno nameščen tudi pripomoček smbmount, pri katerem ni potrebno podati kretnice -t, ampak napišemo kar:

```
smbmount //racunalnik/oznaka oddaljen/imenik
```

Če ne podamo uporabniškega imena, se uporabi uporabniško ime na lokalnem računalniku (glede na spremenljivki USER in LOGNAME) oz. uporabniško ime GUEST. Uporabniško ime in geslo lahko podamo kot opcija pri ukazu mount v obliki "`username=...,password=...`". Še boljše pa je, da uporabniško ime in geslo shranimo v datoteko (npr. `.smbpasswd`), ki jo zaščitimo pred branjem tretjih oseb in v katero napišemo naslednji dve vrstici:

```
username= ...
password=...
```

V tem primeru uporabimo opcijo "`credentials=datoteka`".

V povezavi s protokolom SMB imamo na voljo tudi ukaz `smbclient`, s katerim lahko do deljenega imenika po protokolu SMB dostopamo na podoben način kot pri ftp-ju. Poženemo ga lahko na naslednje načine:

```
smbclient //racunalnik/oznaka -U username
smbclient //racunalnik/oznaka -U username%password
smbclient //racunalnik/oznaka -A .smbpasswd
```

19. Varnost računalniških sistemov

Zgodovina „škodljive“ programske opreme je precej dolga. Prvi računalniški virus za računalnike PC z operacijskim sistemom DOS je bil napisan leta 1986. Napisala sta ga dva brata, mlada študenta iz Pakistana. Imenoval se je Brain, rekli pa so mu tudi Lahore po imenu pokrajine v Pakistanu, kjer sta bila brata doma. Njun namen je bil zaščititi svoje programe pred nelegalnim kopiranjem. Kot zanimivost: avtorja prvega virusa sta sedaj že v srednjih letih in imata v Pakistanu podjetje, ki je ponudnik interneta in se imenuje Brain Limited :-) Vendar pa Brain ni bil prvi računalniški virus, ki se je nenadzorovano širil po svetu. Ta „laskavi“ naslov običajno pripisujemo virusu Elk Cloner iz leta 1982, ki je deloval na računalnikih Apple. Sama ideja računalniškega virusa je sicer še dosti starejša. Že v 70-ih letih prejšnjega stoletja je v omrežju ARPANET, ki je predhodnik današnjega interneta, obstajal virus Creeper, a je bil njegov doseg omejen na zaključeno omrežje velikih računalnikov, ker takrat računalnikov še nismo imeli doma.

Računalniški virusi so bili prvi primer škodljive kode, danes pa poleg njih poznamo tudi druge vrste nevarnosti. Zato si najprej oglejmo, kakšne so razlike med njimi.

Računalniški virus je program, ki se lahko „samodejno“ prekopira na druge sisteme. Samodejno pomeni brez privoljenja oz. brez vednosti uporabnika računalnika. Prvi virusi so se širili s pomočjo izmenljivih medijev, predvsem disket. S pojavom interneta so virusi postali bolj gibljivi. Virus je lahko na izmenljivem mediju oz. na računalniku shranjen na več načinov:

- shranjen je lahko v boot sektorju medija oz. v MBR na disku tako, da se požene vsakič, ko priklopimo medij oz. ko prižgemo računalnik,
- shranjen je lahko kot podaljšek izvršilnega programa tako, da se izvede vedno, ko izvedemo okuženi program,
- shranjen je lahko kot skripta, ki se izvede, ko uporabnik uporabi določen ukaz iz lupine,
- shranjen je lahko kot makro v dokumentih, ki podpirajo makroje in se izvede vedno, ko odpremo okužen dokument,
- shranjen je lahko kot skripta na strežniku, ki se izvede vedno, ko z brskalnikom brskamo po straneh na okuženem strežniku.

Računalniški virusi so tem bolj nevarni, čim bolj dolgo ostanejo skriti. Virus, ki takoj, ko poženete okuženi računalnik, zbriše ves sistem, v splošnem ni preveč nevaren, saj se ne uspe dovolj razširiti. Po drugi strani pa virus, ki se npr. do določenega datuma samo širi in šele nato začne delati škodo, predstavlja dosti večjo nevarnost,.

Računalniški črv je program, ki se širi po omrežju in ne okuži datotek, lahko pa jih zbriše oz. pokvari. Črv torej vsebuje dva mehanizma:

- zna se prekopirati na drugi računalnik,
- na ciljnem računalniku se zna samodejno pognati oz. preslepiti uporabnika, da ga požene.

Predvsem drugi cilj je dokaj težko doseči. Običajno se izrabijo napake v strežniških programih, ki tečejo na računalniku. Še bolj uspešen način za širjenje črvov pa je v obliki priponke k elektronski pošti. Priponka ima zanimivo ime (npr. I LOVE YOU) in zato jo uporabnik odpre ter s tem požene črva.

Ker se računalniški črv ne shrani na ciljnem računalniku v splošnem ob okužbi pomaga, da računalnik enostavno ugasnemo in potem spet prižgemo. Problem pa je v tem, da se lahko računalnik takoj po vklopu in povezavi v internet spet takoj okuži.

Trojanski konj je podoben računalniškemu črvu, le da se ne zna sam prekopirati na drug računalnik. Prenašajo ga uporabniki sami, ker jih z imenom zavede in mislijo, da so prenesli nekaj drugega.

Hoax je posebna in lahko tudi nevarna oblika napada na računalnike v obliki lažne sporočila o računalniškem virusu, ki ga dobimo preko elektronske pošte. Ker je sporočilo napisano zelo čustveno, ga uporabniki množično razpošiljajo naprej, lahko pa po navodilih v sporočilu celo sami poškodujejo svoj računalnik. Leta 2000 je bilo npr. razširjeno sporočilo, ki je eno od sistemskih datotek na sistemu MS Windows (SULFNBK.EXE) opisovalo kot nevarni virus, ki ga naj uporabniki čim prej zbrišejo.

Izraz **stranska vrata** oz. **backdoor** označuje metodo, ki nepooblaščeni osebi omogoči dostop do računalnika, z uporabniškimi ali pa z administratorskimi pravicami. Najpogosteje se v ta namen na ciljnem računalniku zažene namenski program, ki pride tja s pomočjo računalniških virusov, črvov ali pa trojanskih konjev. Pogosto napadalci spremenijo običajne uporabniške programe (npr. P2P program) tako, da vsebujejo stranska vrata. Programerji lahko nalašč skrijejo odprta vrata v svoje programe, npr. v spletne igrice in programe za prenašanje datotek. Nevarni so lahko tudi odprtokodni programi, če jih vzamemo v sumljivem odlagališču. Leta 2003 je bil izveden poskus, da bi stranska vrata odprli v jedru Linuxovega jedra. Šlo je za 2 vrstici kode, ki jih je nekdo podtaknil v razvojno različico, a so prevaro odkrili še preden je razvojna različica prišla v uporabo. V literaturi je opisana tudi zanimiva, pri kateri so spremenili običajni prevajalnik tako, da je vsak program,kije bil preveden z njim, imel stranska vrata. Stranska vrata so najpogosteje uporabljena za širjenje nezaželene pošte, za DOS napade in za širjenje druge škodljive kode.

Vohunski programi oz. **spyware** so danes pogosta oblika škodljivih programov, ki so napisani z namenom zaslužka. Ko se poženejo na ciljnem začnejo zbirati podatke o uporabnikovem obnašanju in ga prepričevati, da obišče določene internetne strani oz. kupi določene izdelke. To naredijo tako, da mu prikazujejo reklamna sporočila ali pa spremenijo obnašanje brskalnika tako, da se pri iskanju na spletu na začetku pojavijo povezave do določenih strani. Še bolj nevarni so key loggerji, ki sledijo tipkanju uporabnika in zbirajo gesla ter npr. številke bančnih kartic.

Posebna oblika škodljivega programa je **rootkit program**. Tako s skupnim imenom imenujemo programe, ki se vgradijo v operacijski sistem in tako v celoti prevzamejo nadzor nad računalnikom. Programi so si zelo različni in obstajajo za vse sisteme. Lahko počnejo karkoli, na primer:

- na računalniku odprejo stranska vrata,
- onemogočijo dostop do določene strojne opreme, npr. DVD enote,
- skrijejo posamezne datoteke in cele imenike, tako da jih uporabnik ne vidi.

V praksi rootkit programi niso omejeni na eno od naštetih funkcij ampak počnejo vse in še kaj drugega. Še posebej je nevarna funkcija skrivanja datotek, saj lahko skrijejo sami sebe ali pa druge škodljive programe tako dobro, da jih niti protivirusni programi ne morejo najti.

Rootkit programi se pogosto širijo v obliki računalniških črvov in trojanskih konjev. Svojega cilja ne dosežejo tako, da ukradejo ali ponaredijo administratorsko geslo. To namreč niso običajni programi, ki se izvajajo z administratorskimi pravicami, ampak zmorejo več. Izvajajo se kot del operacijskega sistema in prilagodijo njegovo obnašanje. Na računalniku jih najdemo v različnih, tudi nenavadnih oblikah:

- lahko so v ROM-u od strojne opreme, ki je instalirana v računalniku, npr. v kakšnem čipu na matični plošči ali v kakšni razširitveni kartici, v tem primeru jih ni mogoče izbrisati,
- lahko se izdajajo za jedro operacijskega sistema in uporabijo postopek virtualizacije, da naložijo pravo jedro, na ta način popolnoma nadzirajo dostop do strojne opreme,
- veliko rootkit programov se izdaja za gonilnike oz. module operacijskega sistema in tako pridobijo moč, da spreminjajo obnašanje operacijskega sistema,
- lahko se izdajajo za sistemske knjižnice in tako prilagodijo obnašanje sistema,
- lahko se izdajajo za običajne programe, ki jih uporabnik izvaja z administratorskimi pravicami, v tem primeru lahko počnejo vse, kar lahko naredi administrator.

Glede na to, da na okuženem računalniku spreminjajo datoteke, so rootkit programi podobni računalniškim virusom. Glavna razlika je v tem, da se rootkit program ne trudi razmoževati ampak bolj skrbi za to, da vse njegove funkcije delujejo pravilno (npr. da ima nepooblaščena oseba ves čas dostop do računalnika). Razlika je tudi v tem, da namen rootkit programa ni delati neposredno škodo na računalniku (npr. brisanje datoteke), saj mu je v interesu, da je sistem delujoč in da čimbolj dolgo ostane neopažen.

Znana afera z uporabi rootkit programov se je zgodila leta 2005, ko je podjetje Sony BMG, ki je velik založnik glasbenih zgoščenk, na njih sistematično razširjalo rootkit program za operacijski sistem Windows. Ta se je instaliral na računalniku, na katerem ste poslušali glasbo in preprečeval nelegalno kopiranje. Imel pa je napake in je med drugim omogočal nepooblaščenim osebam, da prevzamejo nadzor nad vašim računalnikom.

20. Varne spletne storitve

20.1. Šifriranje

V splošnem ločimo dva sistema šifriranja, simetričnega in asimetričnega.

Pri **simetričnem šifriranju** imamo isti ključ za šifriranje in dešifriranje. Ta ključ je seveda tajen, pogosto pa je vsaj delno tajen tudi postopek, kako ključ uporabiti. Znani algoritmi za simetrično šifriranje so **DES**, **TDES**, **AES**, **IDEA**, **Blowfish** itd. Pomembna prednost simetričnega šifriranja je v tem, da v splošnem omogoča mnogo učinkovitejšo izvedbo (hitrejše kodiranje in dekodiranje, manj zahtevna strojna oprema). Simetrično šifriranje je uporabljeno npr. za zaščito podatkov v GSM in UMTS omrežjih.

Poseben problem se pojavi, če želimo uporabiti simetrično šifriranje, vendar pa obe strani ne poznata ključa. Vendar pa so matematiki tudi za ta problem našli rešitve, npr. algoritem Diffie-Hellman, s katerim si dve strani med seboj izmenjata skrivni ključ na tak način, da tudi če kdo prestreže celotno komunikacijo, tega ključa ne more pridobiti.

Pri **asimetričnem šifriranju** imamo dva ključa, privatnega in javnega. Zato mu pravimo tudi **šifriranje z javnim ključem**. Ideja je naslednja:

- podatke, ki so šifrirane z javnim ključem lahko dešifriramo samo, če poznamo pripadajoči privatni ključ
- kljub temu, da poznamo javni ključ in vidimo poljubno količino šifriranih podatkov ni znan uporaben algoritem, kako bi podatke dešifrirali, prav tako pa ne moremo ugotovili privatni ključ.

Javni ključ poznajo vsi, privatnega pa samo lastnik. Torej lahko podatke šifrira vsak, prebere pa jih lahko samo lastnik zasebnega ključa.

Primera šifriranj z javnim ključem sta algoritma **RSA** in **DSA**:

- RSA – temelji na principu, da velikega števila ni mogoče enostavno razbiti na prafaktorje,
- DSA – tudi temelji na matematičnih principih, a drugačnih (nekaj v zvezi z logaritmi).

RSA je bil prvi splošno uporabljan algoritem z javnim ključem in je še danes zelo pogosto uporabljan. Prvi ga je opisal Anglež Clifford Cocks, a njegovo delo ni prišlo v širšo javnost. Neodvisno od njega so ga leta 1977 opisali Američani Rivest, Shamir in Adleman, ki so ga tudi poimenovali po začetnicah svojih priimkov.

Kreiranje ključa RSA poteka po naslednjem principu:

1. izberemo dve praštevili, npr. p=17 in q=23

2. izračunamo n=p·q, torej n=17·23=391

3. izračunamo koeficient f=(p-1)·(q-1), torej f=16·22=352

4. izberemo praštevilo e, s katerim f ni deljiv, npr. e=13

5. izračunamo število d, za katerega velja d·e = 1 + k·f, kjer je k poljubno (čim manjše?) celo število, v našem primeru po vrsti preizkusimo naslednje račune:

1 + 1*352 = 353 ni deljivo z 13

1 + 2*352 = 705 ni deljivo z 13

1 + 3*352 = 1057 ni deljivo z 13

1 + 4*352 = 1409 ni deljivo z 13

...

1 + 12*352 = 4225 je deljivo z 13 in sicer 4225=325*13

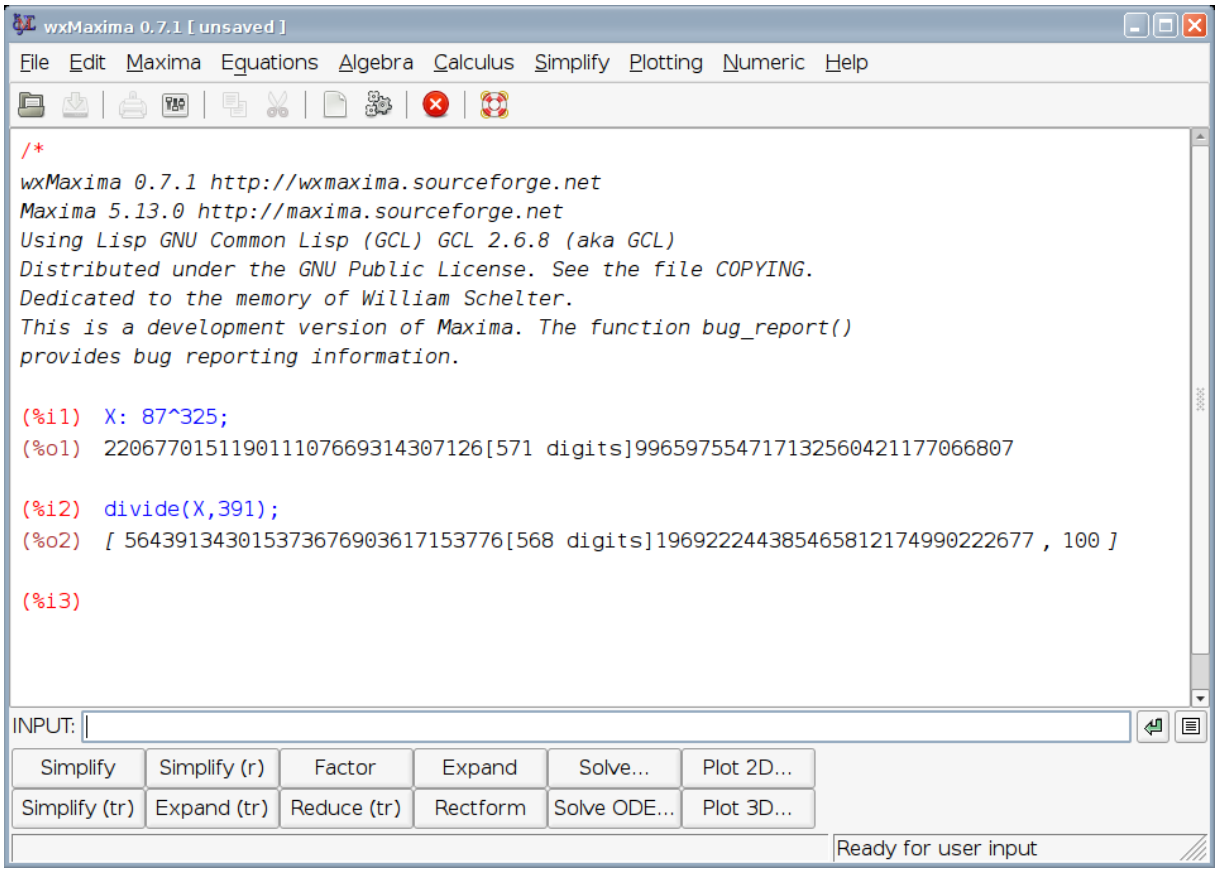
Torej izberemo d=325.

6. Javni ključ je par (n,e), torej v našem primeru (391,13), privatni ključ pa je pa (n,d), torej v našem primeru (391,325).

7. Če želimo šifrirati neko črko ali število, npr. m=100, potem njeno kodo c izračunamo kot ostanek pri deljenju m^e / n, torej v našem primeru dobimo kodo c=ostanek(100^13/391)=87

8. Da bi dešifrirali število 87, izračunamo ostanek pri deljenju c^d/n, torej v našem primeru dobimo m=ostanek(87^325/391)=100, za izračun tega seveda potrebujemo namenski kalkulator oz. namenski algoritem, saj ima številka 87^325 nič manj kot 631 števk, v praksi pa so številke še mnogo večje (na sliki spodaj je izračun s prostim programom Maxima)

9. če napadalec uspe razstaviti številko 391 na praštevili 17 in 23, potem seveda lahko izračuna število d, saj pozna tudi število e, ki je del javnega ključa.



DSA uporablja za šifiriranje in dešifriranje še bolj zahteven postopek in je zato počasnejši. Njegova slabost je tudi v tem, da ni tako raziskan (v primeru šifriranja je večja raziskanost prednost, saj sklepamo, da če prejšnjih 30 let niso našli postopka za razbijanje, potem ga tudi v naslednjih letih še ne bodo)

20.2. Elektronski podpis in digitalno potrdilo

Elektronski podpis in digitalno potrdilo predstavljata dve zanimivi uporabi asimetričnega šifriranja. Obe uporabljata lastnost, da je možno RSA uporabiti tudi v obratni smeri kot pri klasičnem šifriranju, torej, da za šifriranje uporabimo privatni ključ, za dešifriranje pa javni ključ.

Pri **elektronskem podpisovanju** želimo zagotoviti, da besedila ni mogoče neopazno spremeniti

Predpostavimo da želi oseba Alenka poslati osebi Borut elektronsko podpisano pismo (v angleških besedilih običajno uporabijo imeni Alice in Bob). Pismo sestavi in na osnovi njegove vsebine tvori neko oznako, ki dovolj dobro zagotavlja avtentičnost. Nato to oznako šifrira s svojm privatnim ključem in dobljeno kodo pripne besedilu. Ko to pismo pride k Borutu on iz vsebine tudi sam izračuna isto vsebino, nato pa s pomočjo Alenkinega javnega ključa dešifrira pripeto oznako. Če se izračunana in pripeta oznaka ujemata, potem sporočilu lahko zaupa. Če bi kdo želel spremeniti vsebino, bi moral izračunati novo oznako in jo šifrirati z Alenkinim privatnim ključem, tega pa seveda ne more. Če jo šifrira s kakšnim drugim ključem, seveda tudi ne bo uspelo, kajti dešifriranje z Alenkinim javnim ključem ne bo delovalo.

Kot vidimo iz tega kratkega opisa, elektronsko podpisano sporočilo v splošnem ni skrito. Preberejo ga lahko vsi, le ponarediti ga ne morejo. Seveda pa lahko Alenka podpisano sporočilo kot celoto šifrira z Borutovim javnim ključem in potem bo sporočilo lahko prebral samo Borut.

Eden od razširjenih računalniških programov, ki omogoča podpisovanje elektronske pošte, je PGP iz projekta GNU (`http://www.pgpi.org/`)

Z **digitalnim potrdilom (certifikatom)** želimo zagotoviti, da je nek subjekt (oseba, naprava, računalniši program) res to, za kar se izdaja. Ob tem postavimo še dodatno zahtevo, da če nekdo uporabi digitalno potrdilo za opravljanje neke storitve, potem pozneje svojega dejanja ne more zanikati.

Digitalno potrdilo je enostavna digitalna vizitka, na kateri so zapisani podatki o lastniku, do kdaj potrdilo velja in njegov javni ključ. Vendar pa problema avtorizacije ne moremo rešiti brez posredovanja tretje osebe, saj se lahko vsak kadarkoli izdaja za nekoga drugega, npr. tvori potrdilo s sosedovim imenom in svojim javnim ključem. Zato je vsako digitalno potrdilo elektronsko podpisano s strani neke ustanove, ki ji oba zaupata. Ker je nemogoče imeti eno samo takšno ustanovo za cel svet, je v uporabi hierarhični sistem. Oseba Alenka se predstavi s potrdilom, ki ga je podpisala ustanova AlenkinCA (kratica **CA** pomeni **Certificate Authority**). Oseba Borut ne ve, ali je ustanova AlenkinCA res prava ali pa si jo je Alenka kar izmislila. Zato ima digitalni podpis s strani ustanove AlenkinCA v sebi kazalec na svojo digitalno potrdilo, ki pa je podpisano s strani ene druge, bolj

globalne ustanove, ki ji tudi Borut zaupa. Ko Borut preveri, da ima AlenkinCA ustrezno digitalno potrdilo in da je torej res veljavna ustanova, potem zaupa, da je tudi Alenka prava, saj za njo jamči AlenkinCA.

Hierarhični sistem CA-jev je bistveni del tako imenovane **Public Key Infrastrukture (PKI)**, ki jo obravnavamo kot tehnologijo za zagotavljanje varnega poslovanja preko nezavarovanih internetnih povezav. Za vsako digitalno potrdilo, ki ga CA odobri (v praksi običajno CA-ji sami tvorijo pare javnega in privatnega ključa in ne dovolijo, da bi jim ti prinesel svoj ključ), CA tudi poskrbi, da se javni ključ zabeleži v posebnem **javnem imeniku**, tako da lahko kdorkoli dostopa do njega in ti tako pošilja zaupna sporočila. Del PKI so tudi tako imenovane **RA** (**Registration Authority**), ki preverjajo in zavedejo zahteve za izdajo digitalnega potrdila in s tem preprečujejo zlorabe. Brez takega preverjanja bi se namreč lahko zgodilo, da bi kdo zahteval potrdilo v imenu drugega, po drugi strani pa bi lahko nekdo zanikal, da je res on pridobil določeno digitalno potrdilo in posledično tudi, da ga je uporabljal. Zelo znano podjetje, ki podpisuje digitalna potrdila ostalim CA-jem je **VeriSign** iz ZDA.

21. Programske licence

Programska licenca je pravni termin, ki podeljuje oz. omejuje pravice kupca programske opreme.
Z njo avtor programa določi pogoje uporabe programa in njegovo razširjanje.

Če se ozremo na materialne stvari, (npr. hrana, knjige, avtomobili), potem v pravu velja pravilo, da so stvari, ki jih kupiš, tvoje in lahko z njimi počneš, kar želiš. Problem se pojavi le, če kupljena stvar vsebuje avtorsko delo kot npr. knjige ali glasba. V tem primeru je kupcu, ki je lastnik stvari, vsebino avtorskega dela dovoljeno uporabljati le za osebne potrebe, ne sme pa npr. kopiranih knjig brez dovoljenja avtorja širiti okoli ali kako drugače izkoriščati. Ta princip imenujemo **zaščita avtorskih pravic** in nima neomejenega roka trajanja. Določeno število let po avtorjevi smrti njegovi izdelki niso več zaščiteni in se jih lahko poljubno izkorišča. Kaj je avtorsko delo in kaj ni, je opredeljeno v zakonodaji v t.i. avtorskem pravu. V splošnem je avtorsko delo možno na področju književnosti, znanosti in umetnosti.

Današnje zakonodaje poznajo še en podoben princip zaščite, ki pa ima v osnovi čisto drugačne cilje in vlogo. Gre za **patent**, s katerim lahko zaščitimo nove tehnologije in izume. Če pridobimo patent pomeni, da enake tehnologije nek določen čas ne sme izkoriščati nihče drug, kljub temu, da smo mi vsem povedali, kako zadeva deluje. V Sloveniji patenti veljajo 20 let, vodijo pa jih na Uradu republike Slovenije za lastnino (http://www.uil-sipo.si/). Namen patentov je spodbujati razvoj novih tehnologij, saj se na ta način avtorjem povrnejo stroški raziskav. Patenti za stvari, ki se ne prodajajo, niso smiselni. Primer patenta so npr. nova zdravila. Podjetje, ki si izmisli novo zdravilo ima pravico, da podobnega zdravila določen čas ne sme izdelovati nihče drug. Ko ta čas poteče, pa lahko tudi drugi delajo podobna zdravila (imenujemo jih generična zdravila).

Danes se pogosto pojavlja sporni izraz zaščita intelektualne lastnine, ki nima osnove v pravu in samo zamegljuje stvari. Daje namreč vtis, da imajo različni mehanizmi kot so npr. avtorske pravice in patenti, enake cilje oz. da gre celo za enako stvar. Zmedo na tem področju spodbujajo podjetja, ki želijo za svoj večji zaslužek prestrašiti kupce tako, da bi se ti odpovedali svojim pravicam.

S pojavom programske opreme se je izkazalo, da jo je težko uvrstiti v obstoječe pravne kategorije. To, kar bi si proizvajalci programske opreme želeli je, da se za njihovo delo uporabi princip patentov. Npr. Microsoft je razvil urejevalnik besedil in bi si želel, da drugi proizvajalci (vsaj nekaj časa) ne bi ponudili ekvivalentnega izdelka. Vendar pa zakonodaje o patentih ni mogel uporabiti, kajti ta je dovoljevala le patentiranje materialnih izdelkov, ki jih lahko primeš v roko. Zato so se zatekli k drugačnim rešitvam in so naredili program tako, da je shranjeval podatke v formatu, ki ni javno znan. Hkrati so se varovali tako, da je bil program na voljo le v binarni obliki in se ga zato ne da enostavno analizirati ter posnemati. Ker pa ta dva mehanizma nista zagotavljajo dovolj zaščite, torej, samo z njima ne bi bilo možno na dolgi rok preprečiti konkurence, so začeli za programsko opremo uporabljati zakonodajo o avtorskih pravicah. To sicer za silo gre, vendar pa je programska oprema precej različna od klasičnih avtorskih del in zato so se pojavili problemi. Izkazalo se je, da je klasična zakonodaja prešibka, saj če kupec kupi program in postane njegov lastnik, potem zakonodaja govori le o tem, da se ga ne sme razmnoževati, ne pa tudi o tem, da se ga ne sme analizirati in potem narediti podobnih ter konkurenčnih izdelkov.

Zaradi pomanjkanja dobre zaščite so si podjetja potem izmislila **lastniške programske licence**, s katerimi dodatno omejijo pravice uporabnikov. Glavna ideja pri tem je, da kupec ne postane lastnik programske opreme ampak dobi samo dovoljenje za njeno uporabo pod določenimi pogoji. Da ne bi prihajalo do izigravanja, ko gre za distribucijo na medijih, so se uveljavile zaščitne nalepke. Če namreč nekdo kupi nosilec DVD, potem lahko trdi, da je to njegovo in da lahko z njim počne kar hoče. Toda na tem DVD-ju je nalepka, ki jo je treba pretrgati, če želiš prebrati vsebino Na njej pa piše, da takoj, ko jo pretrgate, sprejmete pogoje navedene v programski licenci. DVD torej je vaš in z medijem lahko počnete kar hočete, le vsebine ne morete prebrati brez tega, da se odpoveste določenim svojim pravicam, npr. ne smete uporabljati programov.

Imamo pa na področju programske opreme še drugo skupino proizvajalcev, ki si tudi želijo neke zaščite svojih izdelkov, a jim je zakonodaja o avtorskih pravicah prestroga oz. si želijo drugačne zaščite. Tukaj gre za proizvajalce **proste oz. odprte kode**. Ti želijo omogočiti, da lahko lastniki programe pod določenimi pogoji razmnožujejo oz. kako drugače izkoriščajo brez posebnega dovoljenja avtorja. V tem primeru je torej programska licenca namenjena temu, da ti da več pravic, kot ti pripadajo kot kupcu običajnega izdelka, če se seveda strinjaš s pogoji. Če se z njimi ne strinjaš, je program vseeno tvoj, ti si njegov lastnik in z njim lahko počneš, kaj hočeš, ne smeš ga pa razmnoževati.

Preden se malo bolj posvetimo programskim licencam za prosto oz. odprto kodo, omenimo še **programske patente**. Proizvajalci, ki uporabljajo lastniške programske licence, se zavedajo, da je uporaba zakonodaje o avtorskih pravicah na področju programske opreme le zasilna rešitev, saj na svoje izdelke gledajo kot na izume oz. novo tehnologijo in bi zato želeli uporabiti mehanizem patentov. Nekatere državah (npr. ZDA) jim je že uspelo prepričati in tam že obstajajo programski patenti. Če si izmisliš npr. nov algoritem za stiskanje podatkov ga lahko patentiraš in s tem za določeno obdobje vsem ostalim programerjem prepoveš uporabo istega algoritma. Torej v splošnem rešitve ni treba več niti skrivati niti omejevati pravice kupcev do analiziranja programov. Ker pa zaenkrat ni mogoče zagotoviti spoštovanja patentov na celem svetu, ti dodatni ukrepi še vedno obstajajo. V Evropski uniji programski patenti ne veljajo, saj zaenkrat prevladuje mnenje, da njihova

uveljavitev ne bi prispevala k napredku ampak bi, prav nasprotno, zavirala napredek zaradi povečevanja monopolov.

Za prosto oz. odprto kodo obstaja zelo veliko različnih licenc, saj si lahko vsak programer izmisli svojo. Zato obstajajo organizacije, ki pregledujejo razne licence in podajajo svoje mnenje. Trenutno so najbolj uveljavljene naslednje štiri organizacijie oz. skupine:

- **FSF**, Free Software Foundation, ustanovljena 1985,
- **OSI**, Open Source Iniciative, ustanovljena 1998,
- **DFSG**, Debian Free Software Guidelines, gre za smernice, ki so bile prvič objavljene leta 1997 in jih vzdržujejo skrbniki projekta Debian.
- **Creative Commons**, ustanovljena 2001, gre za organizacijo, ki poskuša množico raznih licenc združiti v nekaj najbolj osnovnih.

Najbolj pogosto uporabljane licence za prosto oz. odprto programsko opremo so **GPL**, **LGPL** in **BSD**. Licenca GPL ne dovoljuje niti predelave programa v komercialnega niti njegove vključitve v komercialni program. Licenca BSD vse to dovoljuje. Licenca LGPL je nekje vmes, saj dovoljuje, da program v nespremenjeni obliki (običajno kot prevedena knjižnica) uporabimo kot del komercialnega projekta.

Bistvena lastnost licence GPL je tako imenovani „**copyleft**“, to je zahteva, da mora imeti licenco GPL tudi vsako izpeljano delo oz. vsako delo, ki uporabi en sam košček, ki je pod licenco GPL.

Za dokumentacije, ki spremlja proste programe, lahko uporabimo licenco **GFDL**. Pod to licenco je npr. objavljeno vso besedilo v spletni enciklopediji Wikipedia. Ta licenca zahteva, da:

- zasluge za dodano besedilo morajo biti jasno podeljene njenim avtorjem,
- vse spremembe besedila morajo biti zavedene,
- vsaka izpeljanka besedila mora imeti enako licenco, torej GFDL,
- katerikoli avtor lahko zahteva, da določenega dela teksta v nobeni izpeljanki ne smemo odstraniti ali spremeniti.

Za fotografije, slike, knjige in podobno, so se uveljavile licence organizacije Creative Commons. Njihove licence so sestavljene iz štirih osnovnih gradnikov:

- Priznanje avtorstva - Attribution (**BY**) – Pri uporabi dela morate navesti izvirnega avtorja na način, ki ga določi izvirni avtor oziroma dajalec licence.
- Nekomercialno - Noncommercial (**NC**) – Tega dela ne smete uporabiti v komercialne namene.
- Brez predelav - No Derivative Works (**ND**) – Dela ne smete spreminjati, preoblikovati ali ga uporabiti v svojem delu.
- Deljenje pod enakimi pogoji - ShareAlike (**SA**) – Če spremenite, preoblikujete ali uporabite to delo v svojem delu, lahko distribuirate predelavo dela le pod licenco, ki je enaka tej.

Posamezna licenca je potem sestavljena kot kombinacija zgornjih zahtev. Vse kombinacije niso smiselne, tako da ostane 11 možnih licenc. Ker pa licence brez gradnika BY niso priljubljene, v praksi običajno naletimo na eno od naslednjih šest licenc (vse oznake se začenjo s črkama CC):

1. CC-BY
2. CC-BY-NC
3. CC-BY-ND
4. CC-BY-SA
5. CC-BY-NC-ND
6. CC-BY-NC-SA

Če pogledamo na portal **flickr** (www.flickr.com), ki je eno večjih skladišč digitalnih fotografij, lahko ugotovimo, da je tam trenutno:

- 31,4 miljona fotografij pod licenco CC-BY-NC-ND
- 28,3 miljona fotografij pod licenco CC-BY-NC-SA
- 13,6 miljona fotografij pod licenco CC-BY-NC
- 11,9 miljona fotografij pod licenco CC-BY
- 8,1 miljona fotografij pod licenco CC-BY-SA
- 4,1 miljona fotografij pod licenco CC-BY-ND

Fotografije, ki niso označene z ND lahko vzamemo, spremenimo in uporabimo v nekomercialnem projektu. Fotografije, ki poleg tega nimajo niti NC (takšni sta CC-BY in CC-BY-SA, skupaj okoli 20 miljonov slik) lahko uporabimo tudi v komercialne namene, npr. na jumbo plakatih. Tiste slike na portalu flickr, ki niso označene z licenco, so avtorsko zaščitene in jih ne smemo uporabiti oz. širiti brez dovoljenja avtorja.

Licenca CC-BY-SA je zelo podobna (a v podrobnostih nezdružljiva) z licenco GFDL. Licenca CC-BY-NC-SA je podobna licenci GPL, licenca CC-BY pa licenci BSD. Vendar pa licence Creative Commons ne uporabljamo za prosto programsko opremo, saj temu namenu niso namenjene, ker ne vsebujejo potrebnih elementov, ki bi ustrezno zaščitile avtorja (npr. pred tožbami zaradi napake v programu).

Če želimo spremeniti licenco računalniškega programa, dokumentacije, fotografije oz. kakšnega drugega digitalnega izdelka v takšno, ki ni kompatibilna z njo, potem se morajo s tem strinjati vsi avtorji. Le pri dokumentaciji so glede tega pravila možna manjša odstopanja. Če je nova licenca **kompatibilna** z originalno licenco, potem lahko spremembo naredi vsak, tudi tisti, ki ni avtor. Npr. program pod licenco BSD (tukaj je mišljena današnja varianta BSD, ki ni enaka originalni varianti) lahko spremenimo in mu damo licenco GPL.

Zamimiva opcija je **dvojno licenciranje** (dual-licensing). V tem primeru avtor za svoje delo ponudi dve licenci, uporabnik pa si izbere, kaj bo uporabil (eno od ponujenih variant seveda mora uporabiti).

Če za katerokoli delo ni izrecno zapisano, da ima licenco, potem se po zakonodaji šteje, da za to avtor ni podelil nobenih pravic drugim. Zato npr. besedilo in slike na spletu, za katere ni izrecno označena prosta licenca, brez dovoljenja avtorja ne smemo uporabiti v svojem delu, ne glede na to, ali je naš projekt prost ali komercialni.