

Honeypot bat idazten Python-egaz: Ezagutu etsaia

- Orain dala gutxi hasi nintzan erasoak jasotzen nire etxean. Eraso hauek baziren “Brute Force” erasoak SSH-ren kontra. Gaur egun nahiko normala da egoera hau baina arrunt ez dala bada log-eak begiratu eta zure izena topatu mila aldiz.
- Egun horretan pentsatu neban zeozer egin behar nebala erasotzailea harrapatzeko. Barrapunto-n ideia on bat emon deustaten: Honeypot bat jartzea. Hartu neban konqueror, joan nintzan Google-ra eta bilatu neban “honeypot ssh”. Ez zegoen ezebe. Beno, ez zegoen ezebe “merkea”. Orduan egin neban betikoa: Topetan ez badot nik gure dodana nik egiten dot.
- Hasi baino lehen azaltzen gauza bat argi utzi nahi dot: NIRE EUSKARA KALEKO EUSKARA DA. NIK BIZKAIERA BERBA EGITEN DOT ETA BENETAN EZ DAKIDALA BATUA. BARKATU NIRE EUSKARARENGATIK *KALEKOA* DALAKO.

Zer dira honeypot-ak?

- Honeypot-ak badira software edo konputagailu batzuk haien intentzioa da erasotzaileak erakartzea emuletan sistema eragile edo zerbitzu kalteberak. Bi honeypot mota dauz: elkarreragin txikikoa eta elkarreragin handikoak. Bigarrenak, normalean, badira sistema eragile errealek zabalik utzi direla ikasteko zer egiten daben erasotzaileek halako sistemegaz. Lehenak (elkarreragin txikikoak) emulatzen dira zerbitzuak. Haien erabilera mugatua da, noski, baina elkarreragin haundikoak baina seguruagoak dira.
- Guk egingo doguna artikulo honetan izango da honeypot bat elkarreragin txikikoa, emulatuko dogu SSH zerbitzu bat.

Python eta Twisted Conch Liburutegiak

- Python languai libre, erraza eta potente bat da. Python-egaz programatuko dogu gure SSH-ko honeypota. Twisted Conch liburutegiak SSH-ko implementazio libre bat da. SSH 2 protokoloa implementatzen dau. Twisted Conch liburutegiak, Python bezalaxe, librea da (MIT lizentzia erabiltzen dau) eta jaitsi daikegu <http://twistedmatrix.com/> web orrialdetik.
- Artikulu hau idazten ari nazenean, azken bertsioa 0.5.0 da. Jaitsi behar dogun artxiboa bada <http://tmrc.mit.edu/mirror/twisted/Conch/0.5/TwistedConch-0.5.0.tar.bz2>.

Twisted Conch Adibideren bat: SSH zerbitzaria

- Twisted Conch liburutegiagaz adibide batzuk etortzen dira. Haietatik bat interesetan jaku: SSH zerbitzaria. Gure adibidea 'sshsimpleserver.py' dau izena eta hurrengo da adibidearen kodigoa:

```
#!/usr/bin/env python
from twisted.cred import portal, checkers
from twisted.conch import error, avatar
from twisted.conch.checkers import SSHPublicKeyDatabase
from twisted.conch.ssh import factory, userauth, connection, keys, session
from twisted.internet import reactor, protocol, defer
from twisted.python import log
from zope.interface import implements
import sys
log.startLogging(sys.stderr)

"""Example of running another protocol over an SSH channel.
log in with username "user" and password "password".
"""

class ExampleAvatar(avatar.ConchUser):
    def __init__(self, username):
        avatar.ConchUser.__init__(self)
        self.username = username
        self.channelLookup.update({'session':session.SSHSession})

class ExampleRealm:
    implements(portal.IRealm)
    def requestAvatar(self, avatarId, mind, *interfaces):
        return interfaces[0], ExampleAvatar(avatarId), lambda: None

class EchoProtocol(protocol.Protocol):
    """this is our example protocol that we will run over SSH
    """
    def dataReceived(self, data):
        if data == '\r':
            data = '\r\n'
        elif data == '\x03': #^C
            self.transportloseConnection()
            return
        self.transport.write(data)
```

```
publicKey = 'ssh-rsa
AAAAB3NzaC1yc2EAAAABIWAAAGEarzJx8OYOnJmzf4tfBEvLi8DVPrJ3/c9k2I/Az64fxjHf9imyRJBixtQh1H91fNjUIx+4LmrJH5Q
NRsFporcHDKOTwTTYLh5KmRpslkYHRivcJSkbh/C+BR3utDS555mV'
```

```
privateKey = ""-----BEGIN RSA PRIVATE KEY-----
MIIBYAIbAAJhAK8ycfDmDpyZs3+LXwRLy4vAlT6yd/3PZNiPwM+uH8Yx3/YpskSW
4sbUIZR/ZXzY1CMfuC5qyR+UDUbBaaK3Bwyjk8E02C4eSpkabJZGB0Yr3CUpG4fw
vgUd7rQ0ueeZlQIBiWJgbh+1VZfr7WftK5lu7MhtqE1S1vPWZQYE3+VUn8yJADyb
Z4fsZaCrzW91kIqXkE3GIY+ojdhZhk01gbG0118sIgpHWsWKRxK0mvh6ERxKqIt1
xJEJO74EyKXZV4oNj8sJAjEA3J9r2ZghVhGN6V8DnQrTk24Td0E8hU8AcP0FVP+8
PQm/g/aXf2QQkQT+omdHVEJrAjEay0pL0EBH6EVS98evDCBtQw22OZT52qX1AwZ2
gyTriKFVojqeEjt3SZKKqXHSAP/AjBLpF99zcJJZRq2abgY1f91vlchkrWqDHUu
DZttmYJeEfiFBBavVYIFldOlZT0G8jMCMBC7sOSzodFnAiryP+Qg9otSBjJ3bQML
pSTqy7c3a2AScC/YyOwkDaICHnnD3XyjMwIxALRz10tQEKMXs6hH8ToUdlLROCrP
EhQ0wahUTCk1gKA4uPD6TMTChavbh4K630vbKg==
-----END RSA PRIVATE KEY-----""
```

```
class InMemoryPublicKeyChecker(SSHPublicKeyDatabase):
    def checkKey(self, credentials):
        return credentials.username == 'user' and \
            keys.getPublicKeyString(data=publicKey) == credentials.blob
```

```
class ExampleSession:
    def __init__(self, avatar):
        """
        We don't use it, but the adapter is passed the avatar as its first
        argument.
        """

    def get Pty(self, term, windowSize, attrs):
        pass

    def execCommand(self, proto, cmd):
        raise Exception("no executing commands")

    def openShell(self, trans):
        ep = EchoProtocol()
        ep.makeConnection(trans)
        trans.makeConnection(session.wrapProtocol(ep))

    def eofReceived(self):
        pass
```

```

def closed(self):
    pass

from twisted.python import components
components.registerAdapter(ExampleSession, ExampleAvatar, session.ISession)

class ExampleFactory(factory.SSHFactory):
    publicKeys = {
        'ssh-rsa': keys.getPublicKeyString(data=publicKey)
    }
    privateKeys = {
        'ssh-rsa': keys.getPrivateKeyObject(data=privateKey)
    }
    services = {
        'ssh-userauth': userauth.SSHUserAuthServer,
        'ssh-connection': connection.SSHConnection
    }

portal = portal.Portal(ExampleRealm())
passwdDB = checkers.InMemoryUsernamePasswordDatabaseDontUse()
passwdDB.addUser('user', 'password')
portal.registerChecker(passwdDB)
portal.registerChecker(InMemoryPublicKeyChecker())
ExampleFactory.portal = portal

if __name__ == '__main__':
    reactor.listenTCP(5022, ExampleFactory())
    reactor.run()

```

- Adibide hau gordeko dogu fitxategi batean, eta jarriko dogu 'jakin_honeypot.py' izena.

Zer aldatu behar dogu orain?

- Hasi baino lehen aldatzen kodigoa jakin behar dogu zertarako balio dabe 2 klaseak behintzat:
 1. **class EchoProtocol(protocol.Protocol):** Hauxe izango da gure 'protokoloa'. Hemen jasoko doguz bidalitako komandoak eta bidaliko doguz gure erantzunak.

2. `class ExampleSession:` Klase honegaz gestionatuko dogu erasotzaileek irekitzen dabezan sesioak.

- Orain jakin behar dogu non egiten dan autentikazioa eta zelan. Azken lerroetan aurkitzen dogu testu hau:

```
passwdDB = checkers.InMemoryUsernamePasswordDatabaseDontUse()  
passwdDB.addUser('user', 'password')
```

- Lerro hauetan erregistretan dau pasahitzaren datubase bat memorian gordeko dala. Gauza bat: honeypot bat ez balitz izango, EZ GENDUAN ERABILI BEHARKO MODU HAUXE OSO INSEGURUA DALAKO. Datubase honetan erabiltzaile bat sartzen dau: erabiltzaile **'user'** **'password'** pasahitzagaz. Guk hemen jarri daikegu nahi doguzan erabiltzaileak.
- Gogoratu behar dogu guk gure doguna da erasotzaileak sartzen direla, orduan, jarri behar doguz kontu arruntak pasahitz errazegaz, adibidez: root/root, ftp/ftp, web/web, admin/admin, etab... Idatz daikegu hor berton lerro batzuk, aldatzen 'user' guk nahi dogun erabiltzailegaz eta 'password' guk nahi dogun pasahitzagaz baina hau ez jata bapest gustetan (neri behintzat). Beste Python-eko modulo bat egingo dogu erabiltzaileak erregistratzeko. Horrexetarako joan behar dogu lehen lerroetara (hamargarren lerroa) eta sartuko dogu hurrengo kodigoa:

```
01      #!/usr/bin/env python  
02      from twisted.cred import portal, checkers  
03      from twisted.conch import error, avatar  
04      from twisted.conch.checkers import SSHPublicKeyDatabase  
05      from twisted.conch.ssh import factory, userauth, connection, keys, session  
06      from twisted.internet import reactor, protocol, defer  
07      from twisted.python import log  
08      from zope.interface import implements  
09      import sys  
10  
11      from honey_userdb import *  
12  
13      log.startLogging(sys.stderr)
```

- Gorrian dauzen lerroak sartu behar doguzenak dira. Orain sortu behar dogu modulo berri bat **'honey_userdb.py'** dauela izena. Modulu honetan hurrengo kodigoa sartuko dogu:

```

FAKE_USERS_FILE = '/home/joxean/fakeusers'

def add_users(passwdDB):
    file = open(FAKE_USERS_FILE, "r")

    i = 0
    for line in file:
        i += 1
        data = line.split(' ')
        try:
            passwdDB.addUser(data[0], data[1].rstrip())
        except:
            print "Error in fake users file at line " + str(i)
            print sys.exc_info()[1]

```

- Modulo honek 'add_users' funtsio bat deko. Funtsio honek irakurtzen dau fitxategi bat. Fitxategi honetan (FAKE_USERS_FILE aldagaia) edukiko dauz guk sartu gure doguzan erabiltze eta pasahitzako pareak. Nire kasoan sortu dot **/home/joxean/fakeusers** eta hauxe da nire fitxategiaren edukiera:

```

oracle oracle
web web
admin admin
ftp ftp
ftpadmin ftpadmin
web web
webadmin webadmin
test test
guest guest
user user
root root
mysql mysql
webmaster webmaster
linux linux
unix unix
bsd bsd
administrator administrator
fake fake

```

- Berririo aldatu behar dogu 'jakin_honeypot.py' fitxategia. Hurrengo kodigoa sartuko dogu:

```

portal = portal.Portal(ExampleRealm())

```

```

passwdDB = checkers.InMemoryUsernamePasswordDatabaseDontUse()

add_users(passwdDB) # Orain erregistratzen doguz gure fitxategian dauzen erabiltzaileak

portal.registerChecker(passwdDB)
portal.registerChecker(InMemoryPublicKeyChecker())
ExampleFactory.portal = portal

```

- Berriro bere, gorrian daun lerroa berria da. Lerro honek **add_users** funtsioa deitzen dau, **honey_userdb.py** fitxategian dagoela.

Gure 'protokoloa' idazten

- Orain aldatuko dogu EchoProtocol klasea. Jasotzen dogunean textoa beste modulo batera pasatuko dogu, eta modulu horretan prozesatuko dogu eskaerak. Hori izango da gure *benetazko* protokoloa. Gure klasea horrela geratu behar da:

```

class EchoProtocol(protocol.Protocol):
    """
    Hauxe da gure 'benetazko' protokoloa }:->
    """
    lastCmd = ""

    def connectionMade(self):
        print self.transport
        self.transport.write('Wellcome to ' + str(FAKE_OS) + '!\r\n\r\n' + str(FAKE_PROMPT))

    def dataReceived(self, data):
        global FAKE_PROMPT

        if data == '\r':
            retvalue = processCmd(self.lastCmd, self.transport)
            self.lastCmd = ""
            data = '\r\n' + str(FAKE_PROMPT)
        elif data == '\x03': #^C
            try:
                self.transportloseConnection()
            finally:
                return
        else:
            self.lastCmd += data

```

```
self.transport.write(data)
```

- Zer aldatu dogu klase honetan? Lehena, connectionMade metodoa sartu dogu. Erasotzaileak lotzen dan bezain laster hor dekokun kodigoa exekutatuko da. Guk erakutsiko dogu gure 'sistema eragilearen bertsioa':

```
self.transport.write('Wellcome to ' + str(FAKE_OS) + '!\r\n\r\n' + str(FAKE_PROMPT))
```

- FAKE_OS eta FAKE_PROMPT aldagaiak definitu behar doguz. Proposito honexetarako beste modulu bat sortuko dogu. 'honey_identity.py' izena jarriko dogu eta hurrengo textua sartuko dogu:

```
FAKE_USER_CHAR = "$"
FAKE_OS = "OpenBSD bigturd 2.5 GENERIC#172 sparc"
FAKE_SHELL = "bash-2.0"
FAKE_PROMPT = "bash-2.0" + str(FAKE_USER_CHAR) + " "
FAKE_USER = "admin"
FAKE_W = ("USER      TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT",
": Permission denied"
)
FAKE_LS = ("drwxr-xr-x    2 root root    4096 2005-06-06 07:00 bin",
"drwxr-xr-x    3 root root    4096 2005-06-25 16:13 boot",
"drwxr-xr-x   10 root root   14320 2005-07-10 22:19 dev",
"drwxr-xr-x  100 root root    4096 2005-07-11 20:31 etc",
"drwxr-xr-x   10 root root    8192 2005-07-10 01:33 lib",
"drwxr-xr-x    2 root root   49152 2005-05-14 18:47 lost+found",
"drwxr-xr-x    3 root root    4096 2005-07-06 23:11 opt",
"drwxr-xr-x    3 root root    4096 2005-07-06 04:30 oracle",
"drwxr-xr-x    3 root root    4096 2005-07-06 01:51 personal",
"drwxr-xr-x    3 root root    4096 2005-07-06 22:41 pr0n",
"drwxr-xr-x    3 root root    4096 2005-07-06 23:44 private",
"drwxr-xr-x    2 root root    4096 2005-07-10 01:33 sbin",
"drwxr-xr-x    3 root root    4096 2005-07-06 23:11 secure_firewall_ltd",
"drwxr-xr-x    2 root root    4096 2005-05-14 18:49 srv",
"drwxr-xr-x   10 root root         0 2005-07-11 00:08 sys",
"drwxrwxrwt   11 root root    4096 2005-07-11 21:17 tmp",
"drwxr-xr-x   14 root root    4096 2005-07-10 15:52 usr",
"drwxr-xr-x   14 root root    4096 2005-06-06 07:02 var",
"drwxr-xr-x   14 root root    4096 2005-06-06 07:02 videos",
"lrwxrwxrwx    1 root root     25 2005-06-25 16:13 vmunix -> boot/vmunix-2.5-172"
)
```



```
FAKE_RM = "rm: Permission denied"
FAKE_TOUCH = "touch: Permission denied"
FAKE_DENIED = "Permission denied"
```

- Berriro aldatu behar dogu 'jakin_honeypot.py' fitxategia, 'honey_identity.py' fitxategira erreferentzia egiteko:

```
09     import sys
10
11     from honey_userdb import *
12     from honey_identity import *
13
14     log.startLogging(sys.stderr)
```

Honeypot-a Amaitzen (Bazan garaia...)

- Azken gauza faltetan jaku: eta **processCmd** funtsioa non utzi dogu? EchoProtocol klasean sartu dogu dei bat processCmd funtsiora baina ondiño ez dogu implementatu funtsio hori. Berriro bere, beste modulu bat sortuko dogu (azkena da, benetan), 'honey_commands.py' daulea izena eta hurrengo izango da haren kodigoa:

```
import re # Expresio erregularrak erabiltzeko
from honey_identity import * # Gure identitatea }:->

uname_re = re.compile("uname(\ )*.")
ls_re = re.compile("ls(\ )*.")
su_re = re.compile("su(\ )*.")
passwd_re = re.compile("passwd(\ )*.")

def processCmd(data, transport):

    global FAKE_SHELL, con

    print "COMMAND IS : " + data
    transport.write('\r\n')

    if uname_re.match(data):
        transport.write(FAKE_OS)
    elif ls_re.match(data):
        for line in FAKE_LS:
            transport.write(line + '\r\n')
```

```

elif data == "exit":
    transport.loseConnection()
elif data == "w":
    for line in FAKE_W:
        transport.write(line + '\r\n')
elif data == "who":
    transport.write(FAKE_USER)
elif su_re.match(data):
    pass
elif data == "":
    pass
elif passwd_re.match(data):
    transport.write('geteuid: _getuid: Invalid operation')
else:
    transport.write(FAKE_SHELL + ": " + str(data) + ": command not found")

```

- Hona hemen prozesetan doguz jasotako komandoak. Momentuz 'ls', 'uname', 'su', 'passwd', 'w', 'who' eta 'exit' komandoak baino ez doguz erabiltzen. Expresio erregularren bidez analizetan doguz bidalitako komandoak eta 'ls', 'uname', 'su' edo 'passwd' antzekoak badira guk nahi dogun erantzuna bidaltzen dogu. Adibidez, 'ls' antzeko komando bat jasotzen badogu hurrengo kodigoa exekutatuko da:

```

elif ls_re.match(data): # ls antzeko komandu bat bada

    # Irakurtzen dogu FAKE_LS aldagaia, honey_identity.py fitxategian dagoela
    for line in FAKE_LS:
        transport.write(line + '\r\n')

```

Hasiko gaitekez jolasten?

- Amaitu dogu gure honeypot txikia. Orain hasi gaitekez jolasten. Lehen exekutatuko dogu gure honeypot-a terminal batean. 5022 TCP-ko portuan entzungo dau. Aldatu nahi badogu hau, adibidez 22 TCP-ko portuan entzuteko aldatu behar dogu 'jakin_honeypot.py' fitxategian azken lerroak:

```

if __name__ == '__main__':
    reactor.listenTCP(5022, ExampleFactory())
    reactor.run()

```

- Portu hori ez badogu nahi aldatzen dogu eta kitto. Adibidez, gure kasuan jarriko dogu 22,

SSH-ko portu estandarra dala:

```
if __name__ == '__main__':  
    reactor.listenTCP(22, ExampleFactory())  
    reactor.run()
```

- Exekutatu baino lehen gelditu behar dogu OpenSSH zerbitzaria jarrita baldin badekogu. Adibidez, nik Debian distribuzio batean exekutatuko dot hurrengo komandoa:

```
/etc/init.d/ssh stop
```

- Eta orain (root erabiltzaile bezalaxe) exekutatu dogu gure honeypot-a:

```
# python jakin_honey.py  
2005/07/17 14:34 CEST [-] Log opened.  
2005/07/17 14:34 CEST [-] __main__.ExampleFactory starting on 22  
2005/07/17 14:34 CEST [-] Starting factory <__main__.ExampleFactory instance at 0xb799176c>
```

- Beste terminal batetik SSH exekutatu dogu:

```
$ ssh admin@localhost  
admin@localhost's password:  
Wellcome to OpenBSD bigturd 2.5 GENERIC#172 sparc!  
  
bash-2.0$ ls  
drwxr-xr-x  2 root root   4096 2005-06-06 07:00 bin  
drwxr-xr-x  3 root root   4096 2005-06-25 16:13 boot  
drwxr-xr-x 10 root root 14320 2005-07-10 22:19 dev  
drwxr-xr-x 100 root root   4096 2005-07-11 20:31 etc  
drwxr-xr-x  10 root root   8192 2005-07-10 01:33 lib  
drwxr-xr-x  2 root root 49152 2005-05-14 18:47 lost+found  
drwxr-xr-x  3 root root   4096 2005-07-06 23:11 opt  
drwxr-xr-x  3 root root   4096 2005-07-06 04:30 oracle  
drwxr-xr-x  3 root root   4096 2005-07-06 01:51 personal  
drwxr-xr-x  3 root root   4096 2005-07-06 22:41 pr0n  
drwxr-xr-x  3 root root   4096 2005-07-06 23:44 private  
drwxr-xr-x  2 root root   4096 2005-07-10 01:33 sbin  
drwxr-xr-x  3 root root   4096 2005-07-06 23:11 secure_firewall_ltd  
drwxr-xr-x  2 root root   4096 2005-05-14 18:49 srv  
drwxr-xr-x 10 root root      0 2005-07-11 00:08 sys
```

```

drwxrwxrwt    11 root root    4096 2005-07-11 21:17 tmp
drwxr-xr-x    14 root root    4096 2005-07-10 15:52 usr
drwxr-xr-x    14 root root    4096 2005-06-06 07:02 var
drwxr-xr-x    14 root root    4096 2005-06-06 07:02 videos
lrwxrwxrwx     1 root root      25 2005-06-25 16:13 vmunix -> boot/vmunix-2.5-172
bash-2.0$ uname
OpenBSD bigturd 2.5 GENERIC#172 sparc
bash-2.0$ exit
Connection to localhost closed.

```

– Lehen terminalean hurrengo textua topatuko dogu:

```

2005/07/17 14:34 CEST [-] Log opened.
2005/07/17 14:34 CEST [-] __main__.ExampleFactory starting on 22
2005/07/17 14:34 CEST [-] Starting factory <__main__.ExampleFactory instance at 0xb799176c>
2005/07/17 14:35 CEST [SSHServerTransport,0,127.0.0.1] kex alg, key alg: diffie-hellman-group1-shal ssh-rsa
2005/07/17 14:35 CEST [SSHServerTransport,0,127.0.0.1] server->client: aes128-cbc hmac-md5 none
2005/07/17 14:35 CEST [SSHServerTransport,0,127.0.0.1] client->server: aes128-cbc hmac-md5 none
2005/07/17 14:35 CEST [SSHServerTransport,0,127.0.0.1] starting service ssh-userauth
2005/07/17 14:35 CEST [SSHService ssh-userauth on SSHServerTransport,0,127.0.0.1] admin trying auth none
2005/07/17 14:35 CEST [SSHService ssh-userauth on SSHServerTransport,0,127.0.0.1] admin trying auth publickey
2005/07/17 14:35 CEST [SSHService ssh-userauth on SSHServerTransport,0,127.0.0.1] admin failed auth publickey
2005/07/17 14:35 CEST [SSHService ssh-userauth on SSHServerTransport,0,127.0.0.1] reason:
2005/07/17 14:35 CEST [SSHService ssh-userauth on SSHServerTransport,0,127.0.0.1] Traceback (most recent call last):
2005/07/17 14:35 CEST [SSHService ssh-userauth on SSHServerTransport,0,127.0.0.1] Failure: twisted.cred.error.UnauthorizedLogin:
2005/07/17 14:35 CEST [SSHService ssh-userauth on SSHServerTransport,0,127.0.0.1] None
2005/07/17 14:35 CEST [SSHService ssh-userauth on SSHServerTransport,0,127.0.0.1] admin trying auth password
2005/07/17 14:35 CEST [SSHService ssh-userauth on SSHServerTransport,0,127.0.0.1] admin authenticated with password
2005/07/17 14:35 CEST [SSHService ssh-userauth on SSHServerTransport,0,127.0.0.1] starting service ssh-connection
2005/07/17 14:35 CEST [SSHService ssh-connection on SSHServerTransport,0,127.0.0.1] got channel session request
2005/07/17 14:35 CEST [SSHChannel session (0) on SSHService ssh-connection on SSHServerTransport,0,127.0.0.1] channel open
2005/07/17 14:35 CEST [SSHChannel session (0) on SSHService ssh-connection on SSHServerTransport,0,127.0.0.1] pty request: xterm (22L, 116L, 0L, 0L)
2005/07/17 14:35 CEST [SSHChannel session (0) on SSHService ssh-connection on SSHServerTransport,0,127.0.0.1] getting shell
2005/07/17 14:35 CEST [SSHChannel session (0) on SSHService ssh-connection on SSHServerTransport,0,127.0.0.1] <twisted.conch.ssh.session.SSHSessionProcessProtocol instance at

```

```

0xb7991d6c>
2005/07/17 14:35 CEST [SSHChannel session (0) on SSHService ssh-connection on
SSHServerTransport,0,127.0.0.1] <twisted.conch.ssh.session.SSHSessionProcessProtocol instance at
0xb7991d6c>
2005/07/17 14:35 CEST [SSHChannel session (0) on SSHService ssh-connection on
SSHServerTransport,0,127.0.0.1] <twisted.conch.ssh.session.SSHSessionProcessProtocol instance at
0xb7991d6c>
2005/07/17 14:36 CEST [SSHChannel session (0) on SSHService ssh-connection on
SSHServerTransport,0,127.0.0.1] COMMAND IS : ls
2005/07/17 14:36 CEST [SSHChannel session (0) on SSHService ssh-connection on
SSHServerTransport,0,127.0.0.1] COMMAND IS : uname
2005/07/17 14:36 CEST [SSHChannel session (0) on SSHService ssh-connection on
SSHServerTransport,0,127.0.0.1] COMMAND IS : exit
2005/07/17 14:36 CEST [SSHChannel session (0) on SSHService ssh-connection on
SSHServerTransport,0,127.0.0.1] sending close 0
2005/07/17 14:36 CEST [SSHChannel session (0) on SSHService ssh-connection on
SSHServerTransport,0,127.0.0.1] remote close
2005/07/17 14:36 CEST [SSHServerTransport,0,127.0.0.1] connection lost

```

Zer faltetan jako gure honeypot-ari?

- Orain jasotzen dogun argipide guztia STDOUT edo STDERR dispositiboetan idazten dau gure honeypotak, hobeto izango zan fitxategi batean utzi daidala. Ba azaken aldaketa nahiko erraza da. 'jakin_honeypot.py' fitxategian aldatu 15-garren lerroa:

```

09     import sys
10
11     from honey_userdb import *
12     from honey_identity import *
13
14     log.startLogging(sys.stderr)
15     log.startLogging(open("/var/log/jakin_honeypot.log", "a"))

```

Eta orain?

- Ba orain, experimentatu nahi badozue, jarri gutxienez aste bat honeypot haxe internetean. Nire experimentoan izan dodaz 1824 erasotzeko saiaketak aste batean. Etorri ziran Suiza-tik, EE.BB.-etatik, Espaina-tik, Erromaniatik, Korea-tik, Brasil-etik, etab... Ea-ea estatu guztietatik... Jarri eta dibertitu :)

OHARRA: Estatu batzuetan ilegala izan daiteke honeypotak jartzea. Ilegalak ez izateko kalteberak izan behar dira eta apurtu behar dabez erasotzaileek, orduen, tentuz ibili, ea zertarako erabiltzen dogun! Beste zera bat: Honeypotak jasotzen

dauen argipidea ez dau balio, adibidez, epaiketa batean. Beno, balio dabe, baina epaitegi batek agintzen badau jarri behar dala.

Erreferentziak

[Honeypots, Intrusion Detection, Incident Response](#)

[Project Honey Pot: Distributed Spam Harvester Tracking Network](#)

[Thp - Tiny Honeypot](#)

[Honeyd - Network Rhapsody for You](#)

[Honeypot Wikipedian](#)

["Know Your Enemy: Everything you need to know about honeypots"](#)

[SANS Institute: What is a Honey Pot?](#)

[Open Directory Project-Honeypots and Honeynets](#)

Euskal Herria, 2005-ko Uztaiaren 17-xe

Joxean Koret