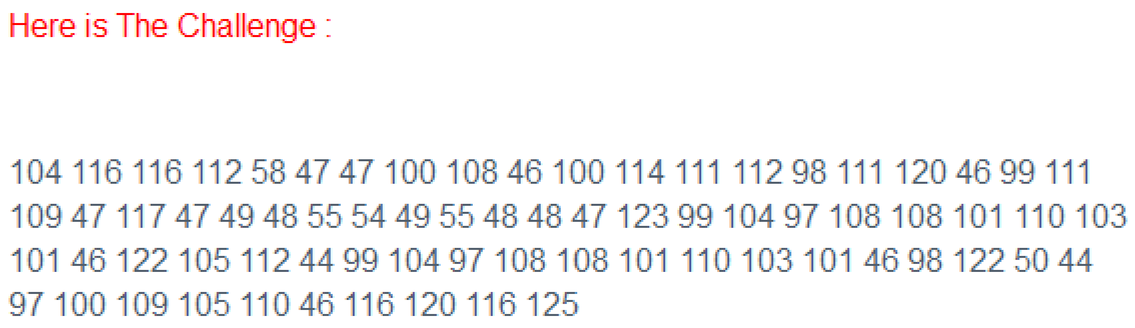
Ninja Security Challenge: My Solution

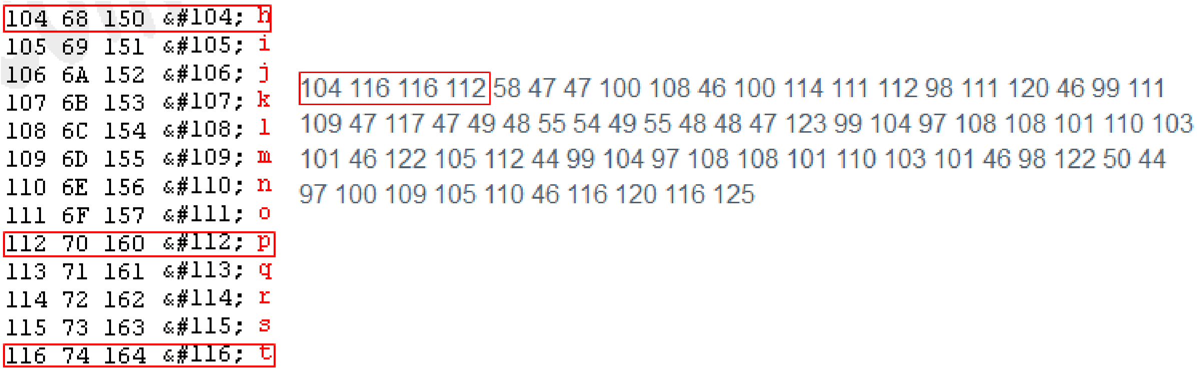
Source: http://ninja-sec.com/index.php/ninja-sec-challenge-2/



Here is The Challenge :

104 116 116 112 58 47 47 100 108 46 100 114 111 112 98 111 120 46 99 111
109 47 117 47 49 48 55 54 49 55 48 48 47 123 99 104 97 108 108 101 110 103
101 46 122 105 112 44 99 104 97 108 108 101 110 103 101 46 98 122 50 44
97 100 109 105 110 46 116 120 116 125

My first thought was that those number pairs could be a hint to the ASCII Table. I did a short check with the first 4 Numbers and I get **http**. It looks like a URL! :)



```
104 68 150 &#104; h
105 69 151 &#105; i
106 6A 152 &#106; j
107 6B 153 &#107; k
108 6C 154 &#108; l
109 6D 155 &#109; m
110 6E 156 &#110; n
111 6F 157 &#111; o
112 70 160 &#112; p
113 71 161 &#113; q
114 72 162 &#114; r
115 73 163 &#115; s
116 74 164 &#116; t
```

104 116 116 112 58 47 47 100 108 46 100 114 111 112 98 111 120 46 99 111
109 47 117 47 49 48 55 54 49 55 48 48 47 123 99 104 97 108 108 101 110 103
101 46 122 105 112 44 99 104 97 108 108 101 110 103 101 46 98 122 50 44
97 100 109 105 110 46 116 120 116 125

I use a typical web based ASCII to text converter [1] and I got this URL from the number pairs: http://dl.dropbox.com/u/10761700/{challenge.zip,challenge.bz2,admin.txt}

I download the following files:

challenge.zip – There is one txt file inside, but it’s protected with a password

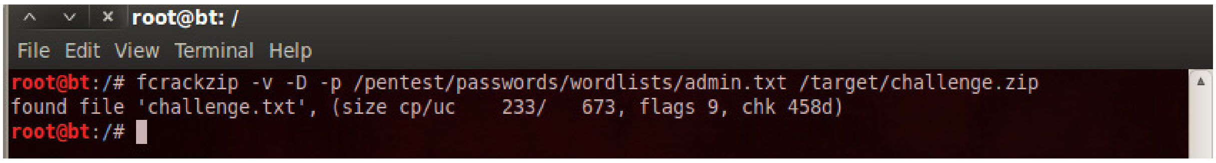
challenge.bz2 – There is a binary file inside called challenge, but without file extension

admin.txt – It looks like a password list

[1] http://home.paulschou.net/tools/xlate/

My next idea was that one of the words inside the **admin.txt** file could be the password for the protected txt file inside the zip archive! I did try a dictionary attack with the wordlist file admin.txt against the file **challenge.zip**, but it seems that I had no success with this method, because no password from the file admin.txt did match.

For this attack I did use a tool called **fcrackzip** (Included in Backtrack 5)

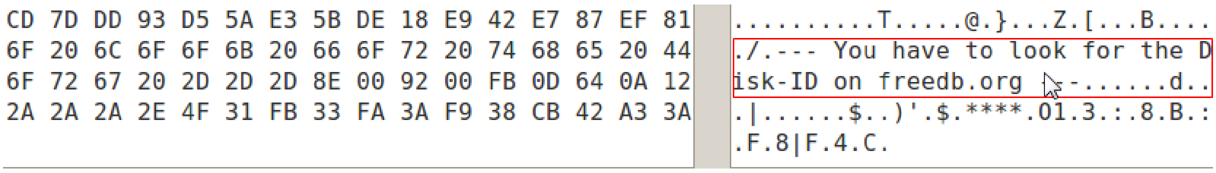


```
root@bt:/# fcrackzip -v -D -p /pentest/passwords/wordlists/admin.txt /target/challenge.zip
found file 'challenge.txt', (size cp/uc    233/   673, flags 9, chk 458d)
root@bt:/# 
```

I've the possibility to start a bruteforce attack with the same tool, but I'm not a great fan of bruteforce attacks and therefore I've decided to analyse the binary file **challenge** first.

I did open the file with a hexeditor and in the end of the file I did find a hint:

You have to look for the Disk-ID on freedb.org

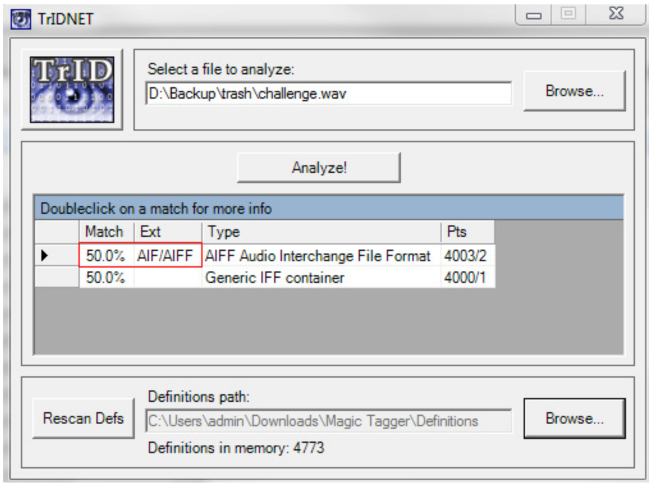


```
CD 7D DD 93 D5 5A E3 5B DE 18 E9 42 E7 87 EF 81   ..........T.....@.}...Z.[...B....
6F 20 6C 6F 6F 6B 20 66 6F 72 20 74 68 65 20 44   ./.--- You have to look for the D
6F 72 67 20 2D 2D 2D 8E 00 92 00 FB 0D 64 0A 12   isk-ID on freedb.org ---......d..
2A 2A 2A 2E 4F 31 FB 33 FA 3A F9 38 CB 42 A3 3A   .|......$..)'.$.****.01.3.:.8.B.:
                                                  .F.8|F.4.C.
```

As we can read in the FAQ of freedb.org, freedb is a database to look up CD information using the Internet. Because of that information I did try the file extension mp3 and wav. The file extension mp3 did not work, but with the file extension wav it was possible to play the file with an 11 Second sequence of a sound track. But who is the artist of that song and how can I find out that Disk-ID?

First I had to check if the file extension is correct, because it was a coincidence that wav work. A friend of me told me about a program called TrID which scans unknown binary files of their file extensions.

I tried it out and I got the file extension AIFF.



Select a file to analyze: D:\Backup\trash\challenge.wav

Doubleclick on a match for more info

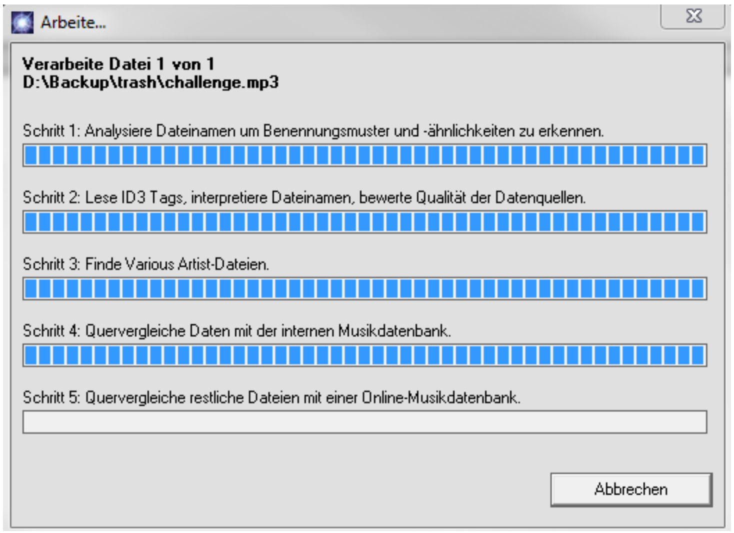
Match	Ext	Type	Pts
50.0%	AIF/AIFF	AIFF Audio Interchange File Format	4003/2
50.0%		Generic IFF container	4000/1

Definitions path: C:\Users\admin\Downloads\Magic Tagger\Definitions

Definitions in memory: 4773

You can find that nice utility here: http://mark0.net/soft-tridnet-e.html

I tried different methods to find out what song it could be. One method was the online Service http://www.audiotag.info. I did upload the challenge track there, but I got no Result. I also tried out the Software Tunatic (http://wildbits.com/tunatic/), Magic MP3 Tagger (http://www.magic-tagger.com/) and Tagscanner (http://www.xdlab.ru/en/). But no of them get me the song back!



Arbeite...

Verarbeite Datei 1 von 1
D:\Backup\trash\challenge.mp3

Schritt 1: Analysiere Dateinamen um Benennungsmuster und -ähnlichkeiten zu erkennen.

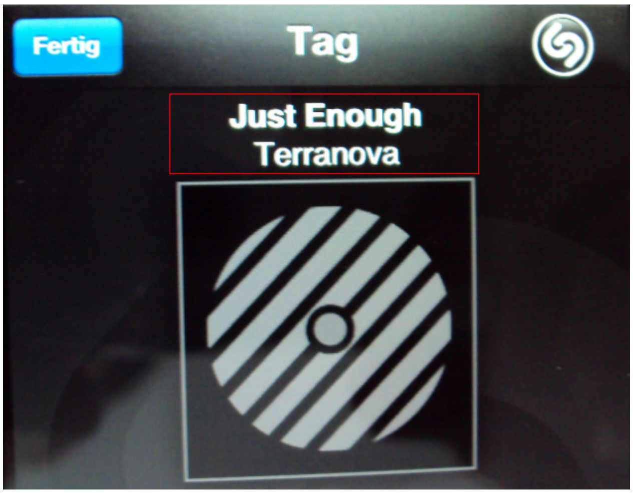
Schritt 2: Lese ID3 Tags, interpretiere Dateinamen, bewerte Qualität der Datenquellen.

Schritt 3: Finde Various Artist-Dateien.

Schritt 4: Quervergleiche Daten mit der internen Musikdatenbank.

Schritt 5: Quervergleiche restliche Dateien mit einer Online-Musikdatenbank.

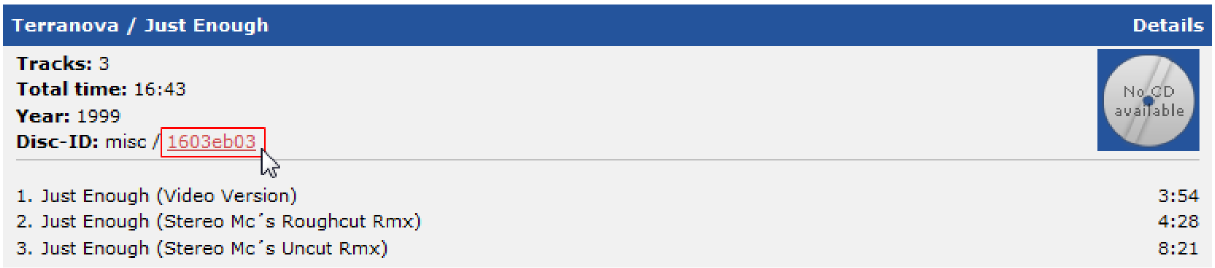
I wouldn't call me an IPHONE Freak, but the APP Shazam (http://www.shazam.com/) did successful identify that track as you can see in the picture below! I'm amazed ☺



Just Enough
Terranova

Let's check the Database of freedb.org and see what we get!

Disc ID: **1603eb03**



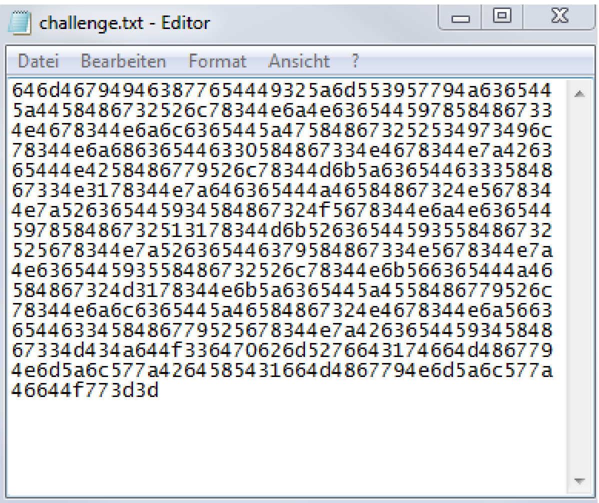
Terranova / Just Enough **Details**

Tracks: 3
Total time: 16:43
Year: 1999
Disc-ID: misc / 1603eb03

1. Just Enough (Video Version)	3:54
2. Just Enough (Stereo Mc´s Roughcut Rmx)	4:28
3. Just Enough (Stereo Mc´s Uncut Rmx)	8:21

The Disc ID **1603eb03** was the password for the zip archive and I did successful extract the file challenge.txt!

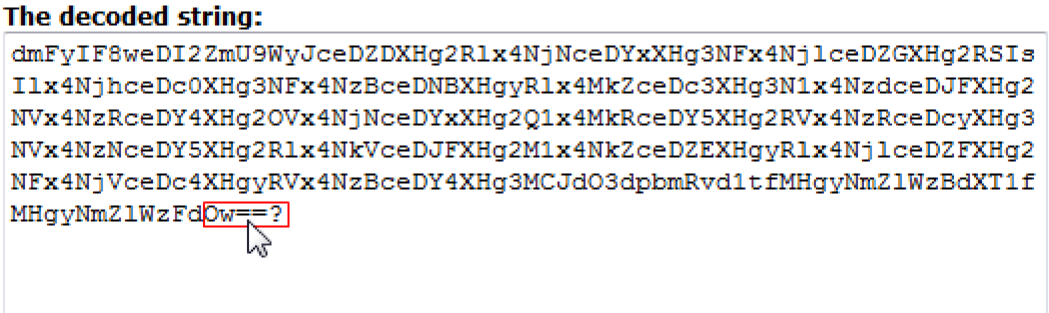
And now let's see what we have:



```
646d467949463877654449325a6d553957794a636544
5a4458486732526c78344e6a4e636544597858486733
4e4678344e6a6c6365445a475848673252534973496c
78344e6a686365446330584867334e4678344e7a4263
65444e4258486779526c78344d6b5a63654463335848
67334e3178344e7a646365444a46584867324e567834
4e7a526365445934584867324f5678344e6a4e636544
597858486732513178344d6b52636544593558486732
525678344e7a526365446379584867334e5678344e7a
4e636544593558486732526c78344e6b566365444a46
584867324d3178344e6b5a6365445a4558486779526c
78344e6a6c6365445a46584867324e4678344e6a5663
6544633458486779525678344e7a4263654459345848
67334d434a644f336470626d5276643174664d486779
4e6d5a6c577a4264585431664d4867794e6d5a6c577a
46644f773d3d
```

Ok, this looks like typical HEX Code. To find it out I use an Online Hex to text Converter Tool[2].

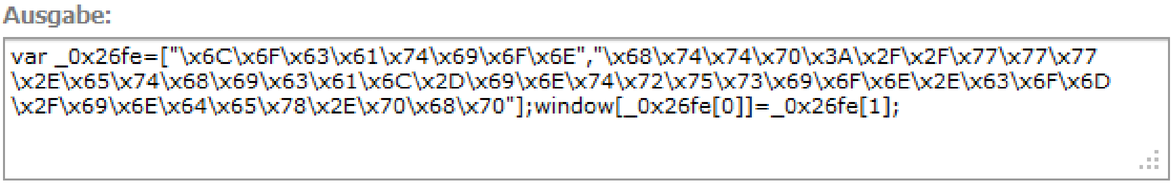
The decoded string is BASE64! A typical Sign for that are the two **==** at the end of the string. For more information about BASE64 or other Crypto Codes visit the website cryptool-online[3].



The decoded string:

```
dmFyIF8weDI2ZmU9WyJceDZDXHg2Rlx4NjNceDYxXHg3NFx4NjlceDZGXHg2RSIs
Ilx4NjhceDc0XHg3NFx4NzBceDNBXHgyRlx4MkZceDc3XHg3N1x4NzdceDJFXHg2
NVx4NzRceDY4XHg2OVx4NjNceDYxXHg2Q1x4MkRceDY5XHg2RVx4NzRceDcyXHg3
NVx4NzNceDY5XHg2Rlx4NkVceDJFXHg2M1x4NkZceDZEXHgyRlx4NjlceDZFXHg2
NFx4NjVceDc4XHgyRVx4NzBceDY4XHg3MCJdO3dpbmRvd1tfMHgyNmZlWzBdXT1f
MHgyNmZlWzFdOw==?
```

Ok, and now let's decode the BASE64 String[4].



Ausgabe:

```
var _0x26fe=["\x6C\x6F\x63\x61\x74\x69\x6F\x6E","\x68\x74\x74\x70\x3A\x2F\x2F\x77\x77\x77
\x2E\x65\x74\x68\x69\x63\x61\x6C\x2D\x69\x6E\x74\x72\x75\x73\x69\x6F\x6E\x2E\x63\x6F\x6D
\x2F\x69\x6E\x64\x65\x78\x2E\x70\x68\x70"];window[_0x26fe[0]]=_0x26fe[1];
```

What the heck is this?

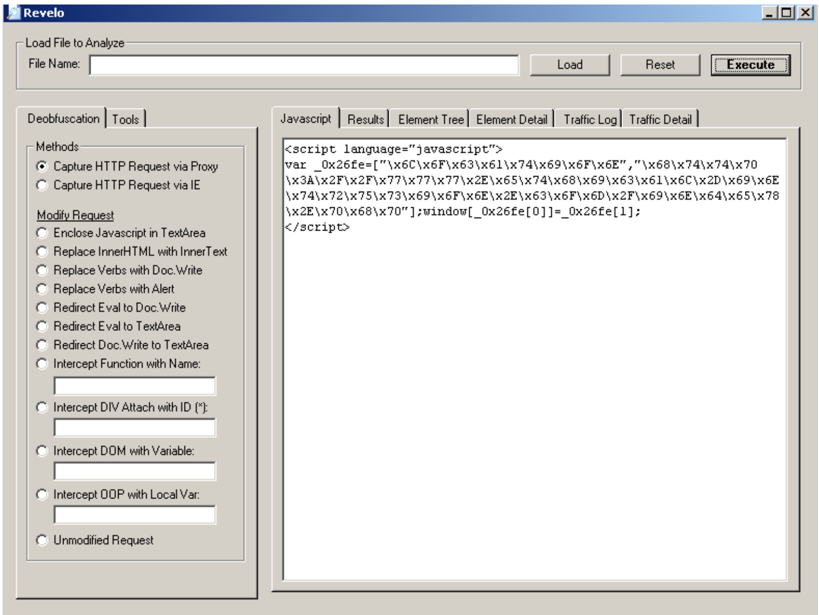
It could be encrypted Javascript Code, but I'm not sure. A quick research in google shows me that it is Javascript and this technique is often used in malicious Websites.

[2] http://www.string-functions.com/hex-string.aspx

[3] http://www.cryptool-online.org/index.php?option=com_content&view=article&id=110&Itemid=133&lang=de

[4] http://floern.com/tools/textencoder

For the further analysis and Decryption I used a Tool called Revelo. I discovered that tool on a nice Security Blog[5] and I run it in a virtual Windows XP Machine.

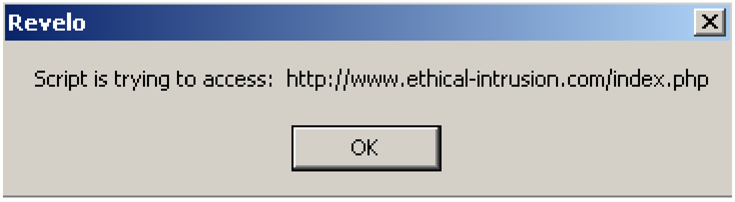


Methods
● Capture HTTP Request via Proxy
○ Capture HTTP Request via IE

Modify Request
○ Enclose Javascript in TextArea
○ Replace InnerHTML with InnerText
○ Replace Verbs with Doc.Write
○ Replace Verbs with Alert
○ Redirect Eval to Doc.Write
○ Redirect Eval to TextArea
○ Redirect Doc.Write to TextArea
○ Intercept Function with Name:
○ Intercept DIV Attach with ID (*):
○ Intercept DOM with Variable:
○ Intercept OOP with Local Var:
○ Unmodified Request

```
<script language="javascript">
var _0x26fe=["\x6C\x6F\x63\x61\x74\x69\x6F\x6E","\x68\x74\x74\x70
\x3A\x2F\x2F\x77\x77\x77\x2E\x65\x74\x68\x69\x63\x61\x6C\x2D\x69\x6E
\x74\x72\x75\x73\x69\x6F\x6E\x2E\x63\x6F\x6D\x2F\x69\x6E\x64\x65\x78
\x2E\x70\x68\x70"];window[_0x26fe[0]]=_0x26fe[1];
</script>
```

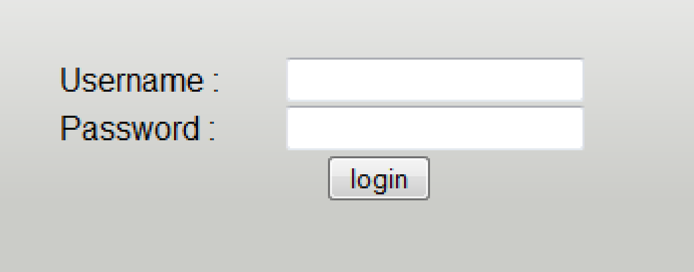
As we can see, our next Destination is http://www.ethical-intrusion.com/index.php

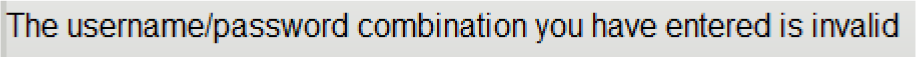


Script is trying to access: http://www.ethical-intrusion.com/index.php

Now we have a Login Form where we have to enter a valid username/password combination. I start a dictionary attack with the passwords from the file admin.txt. Because of the filename I used for all passwords the username admin.

First I enter invalid credentials: Username: bla Password:bla



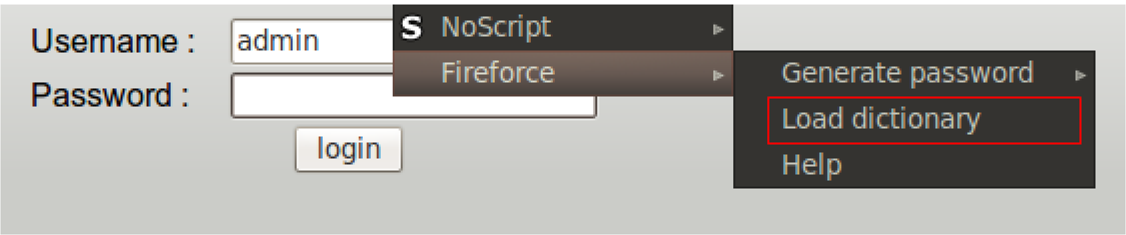


The username/password combination you have entered is invalid

[5] http://www.kahusecurity.com/2012/revelo-javascript-deobfuscator/

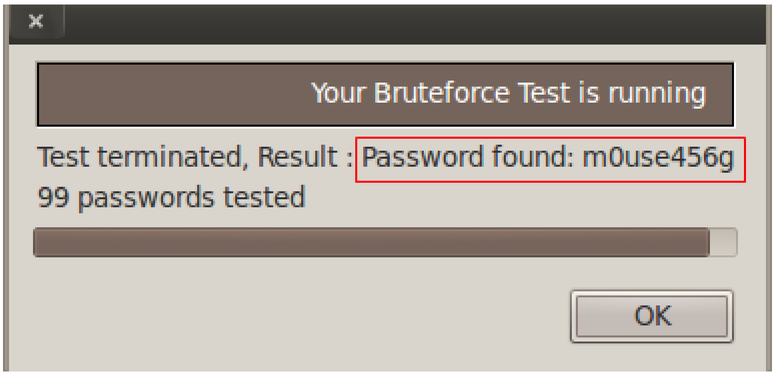
For this attack I use the Firefox Plugin Fireforce.[6]

Fireforce need the textstring: “The username/password combination you have entered is invalid” to successful identify the correct password.


Username : admin

Password :

Password found: m0use456g


Your Bruteforce Test is running

Test terminated, Result : Password found: m0use456g

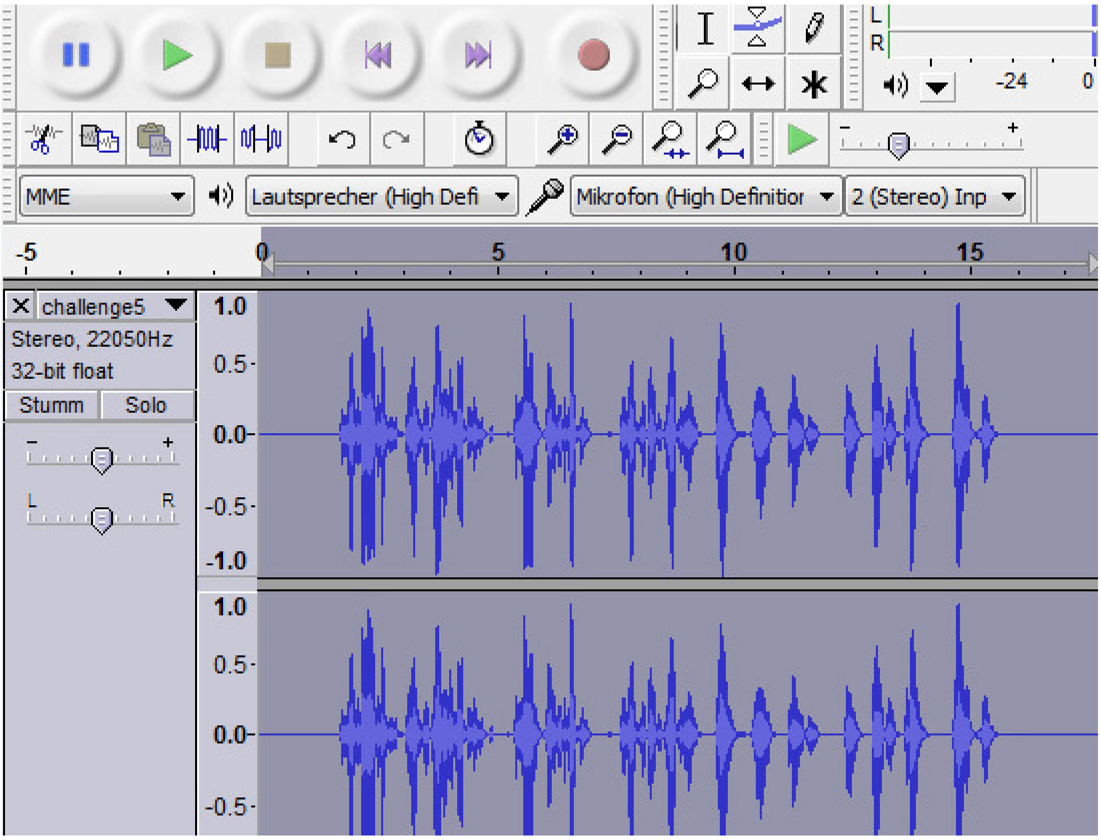
99 passwords tested

With the discovered username/password combination I could enter the website. I can see 2 Links and one of them shows me a youtube video. I did click on play but I couldn’t understand a word because the audio seems to be reverted.

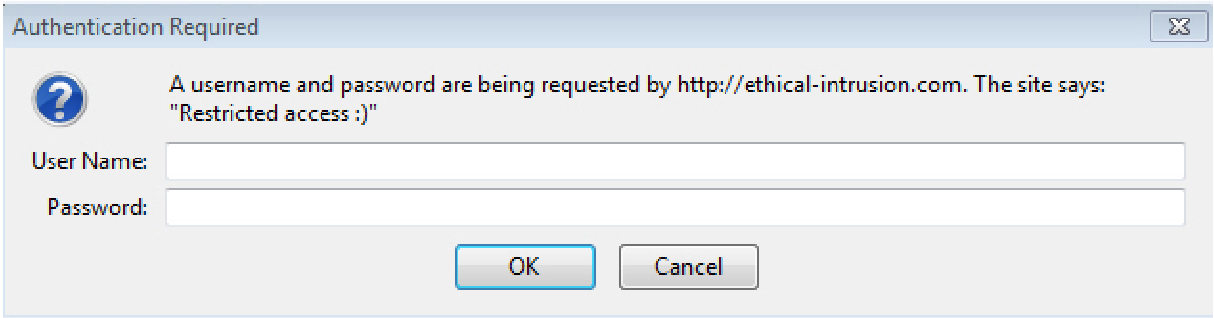
[6] http://www.scrt.ch/en/attack/downloads/fireforce

I did convert the youtube video to an mp3 file and with the software audacity[7] I could edit the audiofile to a clear voice:

Congratulations, you've discovered the website and now listen carefully you have to go to directory a98dhkjd.



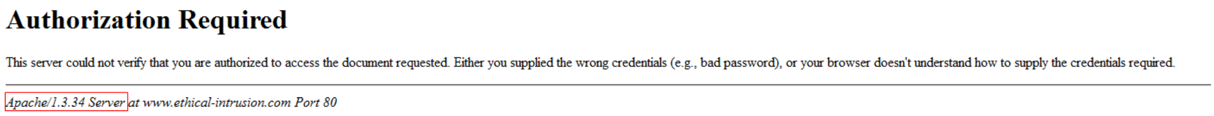
Going to http://www.ethical-intrusion.com/a98dhkjd shows me a htaccess protected Directory.


Authentication Required

A username and password are being requested by http://ethical-intrusion.com. The site says: "Restricted access :)"

User Name:

Password:


Authorization Required

This server could not verify that you are authorized to access the document requested. Either you supplied the wrong credentials (e.g., bad password), or your browser doesn't understand how to supply the credentials required.

Apache/1.3.34 Server at www.ethical-intrusion.com Port 80

[7] http://audacity.sourceforge.net/?lang=de

Now let’s go back to the first Login page and let’s analyse the Links.

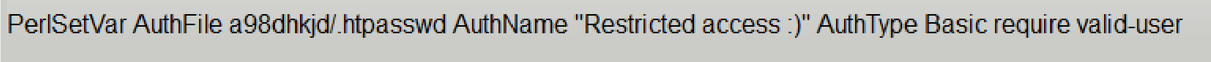
News1 is linked with: http://ethical-intrusion.com/index.php?news=news1.html

News2 is linked with: http://ethical-intrusion.com/index.php?news=news2.html

What we can see is that in both links a separate html file is loaded and displayed into the file index.php! I don’t have much experience in web vulnerabilities, but a technique called **local file inclusion**[8] exists for Links like this to gain access to protected files and directories.

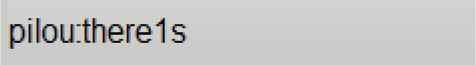
I have used more than one try, but it was possible to read out successful the htpasswd file with a valid username and password to solve this challenge!

http://ethical-intrusion.com/index.php?news=a98dhkjd/.htaccess



PerlSetVar AuthFile a98dhkjd/.htpasswd AuthName "Restricted access :)" AuthType Basic require valid-user

http://ethical-intrusion.com/index.php?news=a98dhkjd/.htpasswd

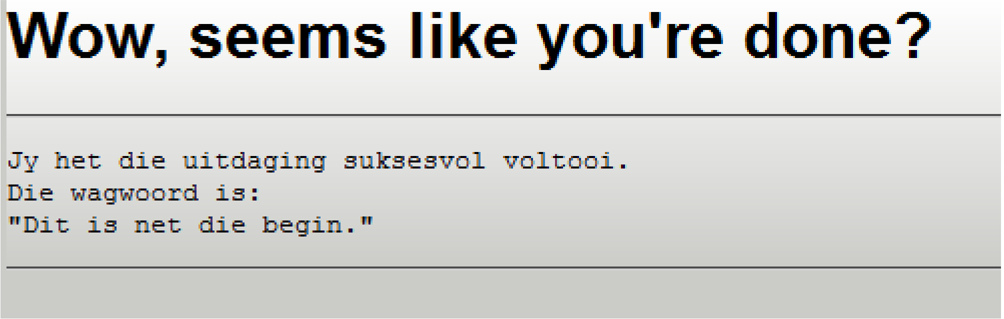


pilou:there1s

http://www.ethical-intrusion.com/a98dhkjd

Username: pilou

Password: there1s



Wow, seems like you're done?

```
Jy het die uitdaging suksesvol voltooi.
Die wagwoord is:
"Dit is net die begin."
```

Thank you for creating this challenge! ☺

[8] http://kaoticcreations.blogspot.ch/2011/08/automated-lfirfi-scanning-exploiting.html