

Securiteam: [NEWS] Yet another Plaintext Attack on ZIP's Encryption Scheme (WinZIP)

[NEWS] Yet another Plaintext Attack on ZIP's Encryption Scheme (WinZIP)

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2003-02/0016.html>

From: support@securiteam.com

Date: 02/10/03

From: support@securiteam.com

To: list@securiteam.com

Date: 10 Feb 2003 20:20:14 +0200

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

Beyond Security would like to welcome Tiscali World Online to our service provider team.

For more info on their service offering IP-Secure, please visit http://www.worldonline.co.za/services/work_ip.asp

Yet another Plaintext Attack on ZIP's Encryption Scheme (WinZIP)

SUMMARY

The ZIP format is one of the most widely used compression/archival programs on computers systems. Its use is even more extended on Windows platform, with WinZIP program. A vulnerability in the way WinZIP encrypts files allows quick recovery of the files even if they are encrypted with very long passwords (over 18 characters).

DETAILS

The PKZIP encryption scheme has been proved weak in many papers that are listed at the end of this advisory. Mike and Elisa have found another way to attack the encryption scheme. This new method was found using reversing engineering techniques on WinZIP's IBDL32.dll.

A known problem in finding the password that was used to encrypt a file is knowing its valid plain text [1]. Mikel Stay published in 2001 "ZIP Attack with Reduced Known-Plaintext" [2], that improved the first technique, but the problem still remained, getting the plaintext.

Getting plaintext is even more difficult because the plaintext is in its compressed form and a minimal change on the data would represent a great

Securiteam: [NEWS] Yet another Plaintext Attack on ZIP's Encryption Scheme (WinZIP)

alteration, so if we did not know the full file content included on zip file we could not do anything.

The encryption scheme:

Status change functions:

```
void update_keys(char p) {
    key0=crc32(key0, p);
    key1=key1 + (key0 & 0xff);
    key1=key1 * 134775813 + 1;
    key2=crc32(key2, key1>>24);
}
```

```
char decrypt_byte(char b) {
    unsigned short tmp;
    tmp=key2 | 2;
    return(tmp * (tmp^1)>>8);
}
```

To initialize the keys:

```
.
    key0=305419896;
    key1=591751049;
    key2=878082192;

    for(i=0 ; i<strlen(pass) ; i++) {
        update_keys(pass[i]);
    }
.
```

To code a byte of data:

```
.
    tmp=byte^decrypt_byte(byte);
    update_keys(tmp);
.
```

Besides, 12 random bytes are prepended to the compressed data in order to make plaintext attack more difficult. These bytes are very important to initialization state of the stream code. Because if we know the state of the stream code on any time we can reverse it until we get to the beginning.

The attack:

Most ZIP coders use rand function to generate these 12 random bytes, but other zippers use their own functions, such is the case of WinZIP.

Attacking this encryption scheme using this 12 prepended random bytes is not a new idea, [2] describes this kind of attack but a minimum of five files in a zip are needed in order to succeed.

Securiteam: [NEWS] Yet another Plaintext Attack on ZIP's Encryption Scheme (WinZIP)

If we reverse engineer WinZIP rand() generation code, we find the following code.

```
46c5a0 push ebp
46c5a1 mov ebp, esp
46c5a3 mov eax, [IBDL32.dll:Seed]
46c5a8 mov edx, eax
46c5aa add edx, edx
46c5ac add edx, eax
46c5ae shl edx, 2
46c5b1 add edx, eax
46c5b3 mov ecx, edx
46c5b5 shl ecx, 4
46c5b8 add ecx, eax
46c5ba mov edx, ecx
46c5bc shl edx, 8
46c5bf sub edx, eax
46c5c1 shl edx, 2
46c5c4 lea ecx, [eax+edx]
46c5c7 mov [IBDL32.dll:Seed], ecx
46c5cd mov eax, [IBDL32.dll:Seed]
46c5c2 sar eax, 10h
46c5c5 mov ecx, eax
46c5c7 and ch, 7fh
46c5ca movzx edx, cx
46c5cd mov eax, edx
46c5cf ret
```

It seems to be an obfuscated code, but if we analyze this, we can see that C code may be of the sorts of:

```
unsigned short rand()
{
    seed=0x343fD * seed;
    return ((seed >> 16)&0x7fff);
}
```

A normal rand implementation looks like:

```
unsigned short rand()
{
    seed=0x343fD * seed + 0x269ec3;
    return ((seed >> 16)&0x7fff);
}
```

Initially it would appear that the person who wrote this code forgot to add 0x269ec3 to seed, but this also leads to a security problem, because the possible sequences are reduced from $2^{(12*8)}$ to $2^{(3*8)}$. Now we will show how to crack a ZIP file using this problem.

Securiteam: [NEWS] Yet another Plaintext Attack on ZIP's Encryption Scheme (WinZIP)

Reducing the sequences makes it easier to do perform a brute force attack and then guess the state of the stream coder (key0, key1, key2). Using these keys and the known formula from pkcrack's source code (stage2 line 175):

```
/* The equation from section 3.3 is used twice here:
 * (1)  $\text{key1}_{\{n-1\}} + \text{LSB}(\text{key0}_n) = \text{rhs} = (\text{key1}_n - 1) * \text{INVCONST}$ 
 * and
 * (2)  $\text{key1}_{\{n-2\}} + \text{LSB}(\text{key0}_{\{n-1\}}) = (\text{key1}_{\{n-1\}} - 1) * \text{INVCONST}$ 
 *
 * At this point we know  $\text{key1}_n$ ,  $\text{MSB}(\text{key1}_{\{n-1\}})$  and  $\text{MSB}(\text{key1}_{\{n-2\}})$ .
 *
 * From (2) follows
 *  $\text{MSB}(\text{key1}_{\{n-2\}}) = \text{MSB}((\text{key1}_{\{n-1\}} - 1) * \text{INVCONST} -$ 
 *  $\text{LSB}(\text{key0}_{\{n-1\}}))$ 
 * Inserting (1) yields
 *  $\text{MSB}(\text{key1}_{\{n-2\}}) = \text{MSB}((\text{rhs} - 1) * \text{INVCONST} -$ 
 *  $\text{LSB}(\text{key0}_n) * \text{INVCONST} - \text{LSB}(\text{key0}_{\{n-1\}}))$ 
 * which means that either
 * (a)  $\text{MSB}(\text{key1}_{\{n-2\}}) = \text{MSB}((\text{rhs} - 1) * \text{INVCONST}) -$ 
 *  $\text{MSB}(\text{LSB}(\text{key0}_n) * \text{INVCONST} - \text{LSB}(\text{key0}_{\{n-1\}}))$ 
 * or
 * (b)  $\text{MSB}(\text{key1}_{\{n-2\}}) = \text{MSB}((\text{rhs} - 1) * \text{INVCONST}) -$ 
 *  $\text{MSB}(\text{LSB}(\text{key0}_n) * \text{INVCONST} - \text{LSB}(\text{key0}_{\{n-1\}})) - 1$ 
 *
 * It can easily be verified that for any two bytes b1, b2:
 *  $\text{MSB}(b1 * \text{INVCONST} + b2) = \text{MSB}(b1 * \text{INVCONST})$ 
 * (simple exhaustive test on  $2^{16}$  combinations)
 *
 * We have computed  $\text{diff} = \text{MSB}((\text{rhs} - 1) * \text{INVCONST}) - \text{MSB}(\text{key1}_{\{n-2\}})$ .
 * Now all we have to do is find values for  $\text{key0}_n$  so that
 * (following from (1))
 *  $\text{MSB}(\text{key1}_{\{n-1\}}) = \text{MSB}(\text{rhs} - \text{LSB}(\text{key0}_n))$ 
 * and (following from (a) and (b)) either
 *  $\text{diff} = \text{MSB}(\text{LSB}(\text{key0}_n) * \text{INVCONST})$ 
 * or
 *  $\text{diff} = \text{MSB}(\text{LSB}(\text{key0}_n) * \text{INVCONST}) + 1$ 
 *
 * Candidate values are selected using the precomputed lookup table
 * mTab2.
 */
```

Proof of concept:

<<mailto:mikestevens@softhome.net>> Mike Stevens – and
<<mailto:eflanders@springfield.com>> Elisa Flanders have been able to
exploit this weak random generation on ZIP files with more than 3 files
($12 * 3 \Rightarrow 36$ bytes of known plaintext), but with less than two hours on a
Pentium 500Mhz with 128Mb.

Attacks to ZIPs with less than 3 files might be also possible because in
WinZIP the two CRC most important bytes are stored within the 12 "random"

Securiteam: [NEWS] Yet another Plaintext Attack on ZIP's Encryption Scheme (WinZIP)

numbers.

tocrack.zip is a file created with WinZIP 8.0, the password is "&THPOL101%ISLAME@|1" which is a 19 length password , (which impossible to crack if we brute force the password) , but what we are brute forcing is the 3 keys that are generated with the password.

```
[rmn@mightsoft ~]$ zipinfo tocrack.zip
Archive: tocrack.zip 679 bytes 3 files
-rw-rw-rw- 2.0 fat 138 T- defX 8-Feb-03 01:31 file3.txt
-rw-rw-rw- 2.0 fat 163 T- defX 8-Feb-03 01:31 file2.txt
-rw-rw-rw- 2.0 fat 270 T- defX 8-Feb-03 01:31 file1.txt
3 files, 571 bytes uncompressed, 339 bytes compressed: 40.6%
[rmn@mightsoft ~]$ ./zipproof -p tocrack.zip
[*] generating posible secuences.. DONE.
[*] reducing number of posible Key2.. DONE.
[*] Bruteforcing:
    [-] Key2 => 0x6a54f21e
[*] Generating initial keys:
    [-] Key0 => 0xaca8571c
    [-] Key1 => 0x439e8759
    [-] Key2 => 0x508d8f22
[*] Tryng to get password.. Not found!
[E] Is not possible to find a password for these keys, you can use
    findkeys tool (from pkcrack) to get it.
[rmn@mightsoft ~]$
```

It was not possible to recover the password because it was too large, but with those keys, it is very easy to extract data files from ZIP (using pkcrack zipdecrypt).

```
[rmn@mightsoft ~]$ ./zipdecrypt 0xaca8571c 0x439e8759 0x508d8f22
tocrack2.zip result.zip
Decrypting file1.txt (bb6c9531638e70d425ebe60b)... OK!
Decrypting file2.txt (0f57542dbf0e18517a5cee0b)... OK!
Decrypting file3.txt (2d2bc008f19607809298f90b)... OK!
```

Affected targets:

The problem is in IBDL32.dll that is used in some ZIP compressors like WinZIP.

Recommendations:

Even if this problem is not present in your ZIP compressor. As the ZIP encryption scheme is very weak and should not be used to encrypt sensitive data. Use zip and then encrypt with your favorite encryption program.

ADDITIONAL INFORMATION

Bibliography:

[1] – "Known Plaintext Attack on the PKZIP Stream Cipher". Eli Biham and Paul C. Kocher

Securiteam: [NEWS] Yet another Plaintext Attack on ZIP's Encryption Scheme (WinZIP)

[2] – "ZIP Attack with reduced Known-Plaintext". Mikel Stay

[3] – PKZIP Specs: APPNOTE.txt.

The information has been provided by <mailto:mikestevens@softhome.net>
Mike Stevens – and <mailto:eflanders@springfield.com> Elisa Flanders.

=====

This bulletin is sent to members of the SecuriTeam mailing list.

To unsubscribe from the list, send mail with an empty subject line and body to:

list-unsubscribe@securiteam.com

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====

=====

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.

In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.