Known plaintext attack on encrypted ZIP files

Barosan Dragos Laurentiu

Supervisor: Armijn Hemel
Why?

• There is no open source implementation
  • Source Code available for PkCrack by Peter Conrad

• Interesting to study a successful attack on encryption
Research Questions

• How feasible is to obtain plaintext?

• What implementation options are for the attack?
Is it still used?

- Winzip
  - default AES, can use classic ZIP encryption
- Winrar
  - only classic ZIP encryption for ZIP format
- 7ZIP
  - default classic ZIP encryption, can use AES
- PKZIP
  - default AES, can use classic ZIP encryption
- ZIP utility on Linux
  - Only classic ZIP encryption
Zip encryption

• Stream cipher
• Internal state represented by 3 variables on 32 bits each
  • Key0, key1, key2
• Default known internal state updated by the password
  • Afterwards updated by plaintext
• key3 is the actual encryption key and is derived from key2
• 12 bytes encryption header prepended
Internal state keys dependency

- $\text{key0}_i = f(\text{key0}_{i-1}, \text{char})$
- $\text{Key1}_i = g(\text{key0}_i, \text{key1}_{i-1})$
- $\text{Key2}_i = f(\text{key2}_{i-1}, \text{key1}_i)$
- $\text{Key3}_i = h(\text{key2}_i)$
- $\text{Ciphertext}_i = \text{key3}_i \text{ XOR plaintext}_i$
Attacks

• Original attack
  • Eli Biham and Paul C. Kocher
  • Requires 13 plaintext bytes

• ZIP Attacks with Reduced Known Plaintext
  • Michael Stay
  • Requires only 2 plaintext bytes at the cost of complexity
  • Can exploit the PRNG used by InfoZIP

• Yet another plaintext attack to ZIP encryption scheme
  • Mike Stevens and Elisa Flanders
  • Exploit in the PRNG from IBDL32.dll used by WinZip
Compressed?

- Some files are not compressed
  - Even with maximum compression level
- Because the compression algorithm needs redundancy
- In the table: maximum size of a file so it is not compressed

<table>
<thead>
<tr>
<th></th>
<th>Deflate level 1-9</th>
<th>Bzip2</th>
</tr>
</thead>
<tbody>
<tr>
<td>One letter</td>
<td>8</td>
<td>43</td>
</tr>
<tr>
<td>Lorem Ipsum</td>
<td>56</td>
<td>129</td>
</tr>
<tr>
<td>Kafka</td>
<td>64</td>
<td>140</td>
</tr>
<tr>
<td>Pangram</td>
<td>78</td>
<td>162</td>
</tr>
<tr>
<td>Random symbols</td>
<td>127</td>
<td>237</td>
</tr>
</tbody>
</table>

Values are in bytes
Plaintext

• The last byte of the encryption header
  • MSB of the data CRC

• File headers
  • Executable files
  • ZIP files

• Known files from the Internet
  • Pictures
  • Setup files
Chosen plaintext attack

• We have a list of key3’s from the plaintext
• The goal is to find internal state (key2ᵢ, key1ᵢ, key0ᵢ) for some i

1. From the list of key3’s find possible key2 lists
2. For each key2 list find possible key1 lists
3. For each key1 list find one key0 list
4. Discover true key0 list
Locate data

Zip archive format

http://www.codeproject.com/Articles/8688/Extracting-files-from-a-remote-ZIP-archive

en.Wikipedia.org/wiki/Zip_(file_format)
Stage 1

- From the list of key3’s find possible key2 lists

- For efficiency precompute a number of tables as hash maps
  - The inverse of the CRC function

- Using the equations in the paper we come to $2^{22}$ possible key$2_n$
  - trim the plaintext and select n as 13
  - use extra plaintext to reduce the number of key2’s
Implementation

- key2 reduction is computation heavy in this stage
  - The function iterates for the number of extra plaintext bytes and returns the reduced list of keys

- Serial

- Parallel
  - Python Global Interpreter Lock does not allow use of threads in parallel
  - Use parallel processes
  - Creating new processes at every iteration
  - Using shared data between processes
Parallel

• Parallel with new processes every iteration
  • The parallel reduction computation runs 4 times faster than the serial one
  • The program as a whole runs slower as the amount of plaintext becomes larger
  • The cost of managing new processes stays constant while the gains of running parallel become smaller

• Parallel with shared data
  • 80 times slower than previous solution
# Measurements

<table>
<thead>
<tr>
<th>Plaintext (bytes)</th>
<th>Execution time Parallel (minutes)</th>
<th>Execution time Serial (minutes)</th>
<th>System/User time Parallel</th>
<th>System/User time Serial</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>0:34.44</td>
<td>1:03.6</td>
<td>0.0647</td>
<td>0.0026</td>
</tr>
<tr>
<td>122</td>
<td>1:08.5</td>
<td>1:38</td>
<td>0.1648</td>
<td>0.0017</td>
</tr>
<tr>
<td>309</td>
<td>1:49.3</td>
<td>1:56</td>
<td>0.3411</td>
<td>0.0014</td>
</tr>
<tr>
<td>506</td>
<td>2:29</td>
<td>2:07.2</td>
<td>0.5066</td>
<td>0.0012</td>
</tr>
<tr>
<td>1002</td>
<td>3:28</td>
<td>2:22</td>
<td>0.7455</td>
<td>0.0011</td>
</tr>
<tr>
<td>3990</td>
<td>10:07</td>
<td>3:02.1</td>
<td>1.4550</td>
<td>0.0009</td>
</tr>
</tbody>
</table>
Conclusions

• While difficult there are ways of obtaining the necessary amount of plaintext
  • Using the newer attacks, in some cases it is not even necessary

• The attack can be implemented by taking advantage of multiple cores
  • Python makes it difficult because processes must be used instead of threads
Future work

• Implement full attack and release under open source license
  • In C to take advantage of the parallel sections of the algorithm
• Compare performance with PkCrack
• Detailed analysis of the other attacks
Questions ?

Contact: Barosan.dragos@gmail.com