Security Aspects of Fuzzy Hashing

Frank Breitinger, Harald Baier

Hochschule Darmstadt, CASED

17.03.2011

Frank Breitinger

- Bachelor Degree at Hochschule Mannheim in March 2009
- Master Degree at Hochschule Darmstadt in Februar 2011
  - IT-Security
  - Fuzzy Hashing
- Since March 2011 Research Student at CASED
  - Center for Advanced Security Research Darmstadt
- Publications:
  - User Survey on Phone Security and Usage (BioSIG Sept. 2010)
  - Security Aspects of Piecewise Hashing in Computer Forensics (Accepted at IMF Mai 2011)

Motivation [1/2]

- Main question: Is it possible to identify similar files based on a fingerprint, which depends **only** on the files' byte structure?

- Cryptographic hash functions follow the avalanche effect: Changing a bit in the input affects $\approx 50\%$ of the output bits $\rightarrow$ no match

- Fuzzy hashing promises to overcome this problem and discover similarities based on fingerprints.

- Question addressed in this talk: How reliable are the results of Kornblum's approach for fuzzy hashing with respect to an active adversary?

Motivation [2/2] - Applications

1. Forensics (on the file level): Detect similar files
   - Blacklisting:
     - Detect manipulated suspicious files
     - Find fragments of suspicious data
   - Whitelisting: Find variants of unsuspicious files
2. Biometrics: Template protection
3. Malware: Detect obfuscated malware (e.g. metamorphic malware)
4. Junk mail detection

Agenda

Kornblum's Fuzzy Hashing

Security Aspects

Conclusion

Contact, Discussion

Kornblum's Fuzzy Hashing

Kornblum's Fuzzy Hashing

Security Aspects

Conclusion

Contact, Discussion

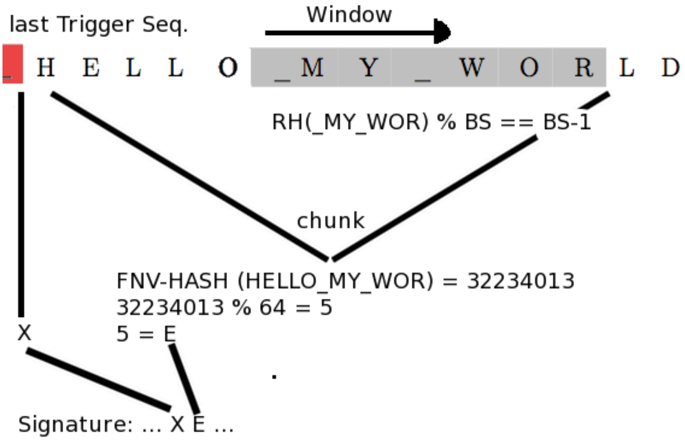
Fuzzy Hashing by Kornblum

- *Context Triggered Piecewise Hashing* (CTPH) (software named `ssdeep`)
- Developed in 2006 based on spamsum-algorithm from A. Tridgell
- Key elements:
  - Block size
  - Rolling hash
  - Traditional hash / piecewise hashing
  - Signature
- Pioneer: *dcfldd*
  - Blocks had a fixed size
  - Non-propagation = yes   alignment robustness = no

Key Elements

- Block size: b
  - $b_{init} = b_{min} \cdot 2^{\lfloor \log_2 (\frac{n}{S \cdot b_{min}}) \rfloor}$
- Rolling Hash at position p in the file:
  - $r_p = F(n_{p-s+1}, n_{p-s+2}, \ldots, n_p)$
  - Allows to compute r_{p+1} cheaply from r_p by removing the influence of n_{p-s+1} and including the new byte n_{p+1}
- Traditional Hash / Piecewise Hashing:
  - Currently, `ssdeep` makes use of Fowler/Noll/Vo (FNV)
  - Alternative hash functions are possible (e.g. SHA-1, MD5)

Workflow


last Trigger Seq.
Window
_ H E L L O _ M Y _ W O R L D
RH(_MY_WOR) % BS == BS-1
chunk
FNV-HASH (HELLO_MY_WOR) = 32234013
32234013 % 64 = 5
5 = E
X
Signature: ... X E ...

Kornblum Signature

- 2 Signatures:
  - Signature 1: Using block size b (at most 64 characters)
  - Signature 2: Using block size $2b$ (at most 32 characters)

- Sample Kornblum signature of `test-file1`:

```
24:TOtUHZbAzIaFG91Y6pYaK3YKqbaCo/6Pqy45kwUnmJrrevqw+oWluBY5b32TpC0:
      TOtU5s7ai6ptg7ZNcqMwUArKvqfZlMC0,"/test-file1"
```

Security Aspects

Kornblum's Fuzzy Hashing

Security Aspects

Conclusion

Contact, Discussion

Characteristics of Kornblum's Implementation

- Signature comparison:
  - Only signatures with the same block size or within a factor of 2 can be compared
  - A successful match needs at least *one common substring in the signature of length 7*
  - A signature has at most 64 characters
- If block size is known, we can calculate trigger sequences:
  - Easy observation: A trigger sequence for b is also a trigger sequence for all block sizes $\frac{b}{2^k}$
  - Concatenation of trigger sequences yields signature characters (e.g. `_MY_WOR` in previous example)
- Attack type depends on the file syntax:
  - BMP / ASCII-files can be changed 'everywhere' (easily)
  - JPG / PDF-files allow a change of header information

Attacks?

What do we like to achieve?

1. **False negatives** for **blacklisting** → anti-blacklisting
   - Modified incriminated files are not detected by the blacklist although perceptual identical to the original known-to-be-bad file
2. **False positives** for **whitelisting** → anti-whitelisting
   - Incriminated files are modified to get a signature of a known-to-be-good file
   - Modified incriminated file is perceptual identical to the original known-to-be-bad file

Attacks for Anti-Blacklisting

- **Blow up a file**: Block size gets larger
- **Edit trigger seq.**: Block size gets different (unpractical)
- **Edit between trigger seq.**: Change one byte in every 7^{th} chunk
- **Adding trigger seq.**: Add several trigger seq. in the beginning of a file
- No semantic attacks like rotations, colour changes, ...

Anti-Blacklisting: Blow up a File

```
$ ls -la hacker_siedlung.jpg
-rw-r--r-- 1 user user 68650 2011-02-23 13:57 hacker_siedlung.jpg

$ dd if=/dev/urandom of=hacker_siedlung.hacked.jpg bs=1 count=280000
280000+0 records in
280000+0 records out
280000 bytes (280 kB) copied, 1.39661 s, 200 kB/s

$ dd if=hacker_siedlung.jpg of=hacker_siedlung.hacked.jpg conv=notrunc
69653+0 records in
69653+0 records out
69653 bytes (70 kB) copied, 0.20225 s, 344 kB/s

$ ssdeep -l hacker_siedlung.jpg hacker_siedlung.hacked.jpg
ssdeep,1.0--blocksize:hash:hash,filename
1536:FLVoUaX+ns+6iAuLNdElzt/CclGbn2OCFN8DXg1BSXHaL++:
   F3l6ew33lG2OMBSXa6+,"hacker_siedlung.jpg"
6144:F6jOMBEjZML1AecfyqefFgQ5wDg+b7LQ7vZOubiPZ:
   F4OMq6i8qefFgUlTsub6Z,"hacker_siedlung.hacked.jpg"
```

Attacks for Anti-Whitelisting

- **Edit between trigger seq.**: Change one byte in each chunk
- **Adding trigger seq.**: Add several trigger seq. in the beginning
- Difference: Adding information vs. editing information
- More computational power than for anti-blacklisting

Example: Editing Between Trigger Sequences

H E L L O _ M Y _ W O R L D

Example: Adding Trigger Sequences

- File need to be changed in the beginning
- One may use global trigger sequences:

Trigger Sequence	Base64 Char.	Trigger Sequence	Base64 Char.
AAAD?Hp	9	AAAV?Hf	l
AAAD?Og	v	AAAf?Ft	p
AAAD?QI	7	AAAr?xj	V
AAAJ?MW	P	AAAx?Fj	1
AAAJ?PJ	F	AAAx?OC	n
AAAJ?VO	Z	AAAx?tx	5

Table 3.1.: Sample pre-computed global trigger sequences and their corresponding Base64 signature characters

- Example: Insertion of concatenation of trigger sequences `AAAD?HpAAAD?OgAAAD?QIAAAJ?MWAAAJ?PJAAAJ?VO` yields Kornblum's signature: `9v7PFZ`

Conclusion

Kornblum's Fuzzy Hashing

Security Aspects

Conclusion

Contact, Discussion

Summary

- CTPH from Kornblum does not withstand an active adversary with respect to
  - blacklisting
  - whitelisting
- Doubtful if piecewise hashing can fulfill the expectations of fuzzy hashing
  - Typically it is possible to flip one bit in each chunk
- In order to create a viable new fuzzy hash function, it will be necessary to find different approaches

Future Work

- Conduct a study if CTPH is applicable in forensics
- Clear definition of:
  - What we expect from a fuzzy hash function?
  - What is a metric for similarity?
- Find a more general approach, which also addresses images, videos, ... not only txt files
- Proof if this might be possible on byte level
- Otherwise new techniques might be needed:
  - E.g. first extract features then hash (e.g. FFT for images)

Contact, Discussion

Thank you for your attention!

- Frank Breitinger, Harald Baier
- `{frank.breitinger,harald.baier}@cased.de`

Source: www.cartoonStock.com