DFRWS

Improving file-level fuzzy hashes for malware variant classification

DFRWS

Ian Shiel & Stephen O'Shaughnessy, TU Dublin 2019

Variants?

- Modified/altered versions of existing malware.
- Symantec ISTR 2018 – Fewer new families, more variants.
- 2017 – 670M new variants detected
- 2018 – 246M
- Why?
  - Avoid detection, upgrade capabilities.
  - Easy to do – hex editor, packing, easy-to-use tools.

Fuzzy hashes?

- Traditional cryptographic hashes (MD5, SHA1 etc) do not scale well to identify malware variants….each variant has a unique hash.

- Fuzzy hashes (aka similarity digests) can be used as part of the static analysis process to compare and identify similar files with varying degrees of confidence.

- ssdeep – 2006 Jesse Kornblum.

Previous Research (1)

1. Algorithms:

- The first widely used similarity hashing algorithm was Kornblum's ssdeep (2006).
- Research published between 2007 and 2013 produced several other similarity hashing methods and algorithms.

  These included MRS (Roussev et al, 2007), sdhash (Roussev 2010), bbhash (Breitinger et al 2012) and tlsh (Oliver et al, 2013).

  BitShred (Jang et al, 2010) was the first specifically designed for malware triage.

Previous Research (2)

2. Comparison of similarity hashing algorithms:

- Properties framework – Breitinger & Baier 2012 [1].
  "Properties of a similarity preserving hash function and their realization in sdhash". The paper compared 3 hashing functions for compression, ease of computation, coverage & similarity score. Also resistance against attack and collision robustness.

- Performance framework – Upchurch & Zhou 2015 [2].
  "Variant: A malware similarity testing framework".
  Precision, recall & F-Measure comparisons of 5 algorithms.

Previous Research (3)

3. Malware detection using fuzzy hashing:

- Selected section hashes used for binary identification (Namanya et al 2016). Combined hashes with Evidence Combinational Theory assigning a ‘degree of belief’ to each hash type. These were used to weigh similarity score and present an overall maliciousness score [3].
- Classification by family performed at file-level. (Sarantinos et al 2016). Found that sdhash had best performance overall and that with “fuzzy hashing it could be possible to identify new malwares, which can be from the same or an emerging malware family, based only on saved fuzzy hashing checksums”. [4]

Issues with file level digests (1)

- **Over-similarity.**
  - Contents of headers (e.g. “This program cannot be run in DOS mode”).
  - Common section contents.
- **Under-similarity.**
  - Similarity is dependent on sequence not just content. Move sections around => artificially reduce similarity.
  - Redundant/contrived sections (e.g. from benign programs).

Issues with file level digests (2)

- **Minor Modifications.**
  - Pagnini et al (2018) found that relatively minor modifications to the source code such as swapping the sequence of instructions had a marked effect on file-level similarity reducing it to zero in some cases where 99.8% of the program was unchanged [5].

Problem Definition

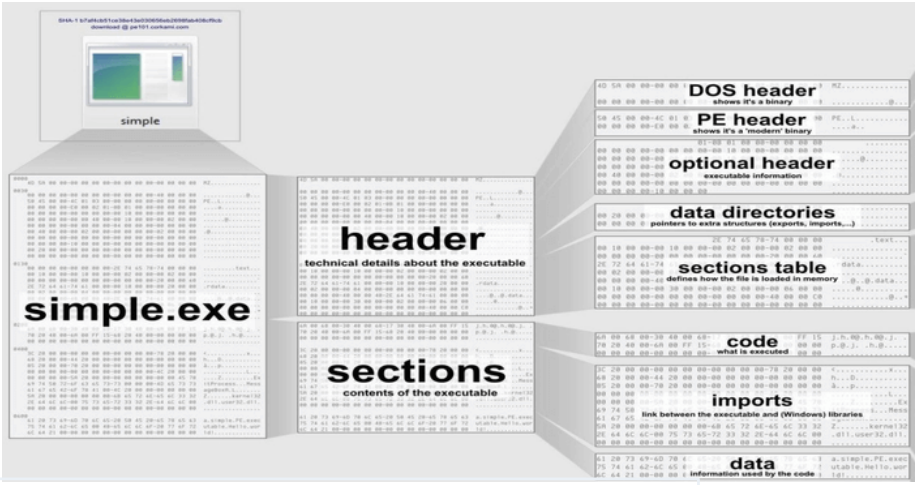
- Detecting malware is necessary but insufficient – **classification is also required.**
- A high proportion of malware are variants of a single family and so should be relatively similar at a binary level. (Sarantinos et al 2016) demonstrated that it is possible to use fuzzy hashing to assist with identifying variants but **could results be improved**?

Sections.

A Windows Portable Executable (PE) consists of headers and sections which tell the Windows OS how to load the executable code.

Section Name	Contents
.text or .CODE	Contains the binary instructions executed by the CPU. Microsoft compilers tend to name this section as .text, Borland compilers use .CODE
.rdata	Import and export information & read-only program data. If there are idata and edata sections the import and export information is stored there instead of in the .rdata section.
.data	Global data accessible in all functions of the program.
.rsrc	Holds resources such as images, icons, and strings that are not considered part of the executable
.idata	Optional, if present contains the import function information instead of it being contained in the .rdata section.
.edata	Optional, if present contains the export function information instead of it being contained in the .rdata section.
.reloc	Holds information for relocation of libraries.

Common PE file sections



Understanding PE structure – The layman's way [6]

Section Hashing – Spilt the PE file into its constituent sections and hash them, not the entire PE file.

What's different?

- **Classification by family** previously performed at **file-level**. (Sarantinos et al 2016)
- Selected section hashes used for **binary identification** not by family (Namanya et al 2016).
- This study uses **section-level** fuzzy hashing to classify malware **by family** and compares the results to file-level using the variant framework proposed by Upchurch & Zhou.

Aim

- Compare malware **classification** results using **section-level** fuzzy hashes to those obtained at file-level.
- Test unobfuscated and obfuscated samples.
- Identify other factors that affect the results.
- Algorithm chosen – ssdeep (standard for malware similarity).
- Considered sdhash but input **and** comparison hashes must be greater than 512 bytes.
- Intention not to compare results from different fuzzy hashing algorithms due to time constraints (additional research underway).

Methodology (1)

- Literature review & Experimental research.
- Develop 2 x software applications (Python) to conduct experiments.
- Data source – VirusTotal March 2018 PE malware set (exes & dlls).
- Only include samples where 10+ VirusTotal engines analysed PE file & 60% agreed on family.

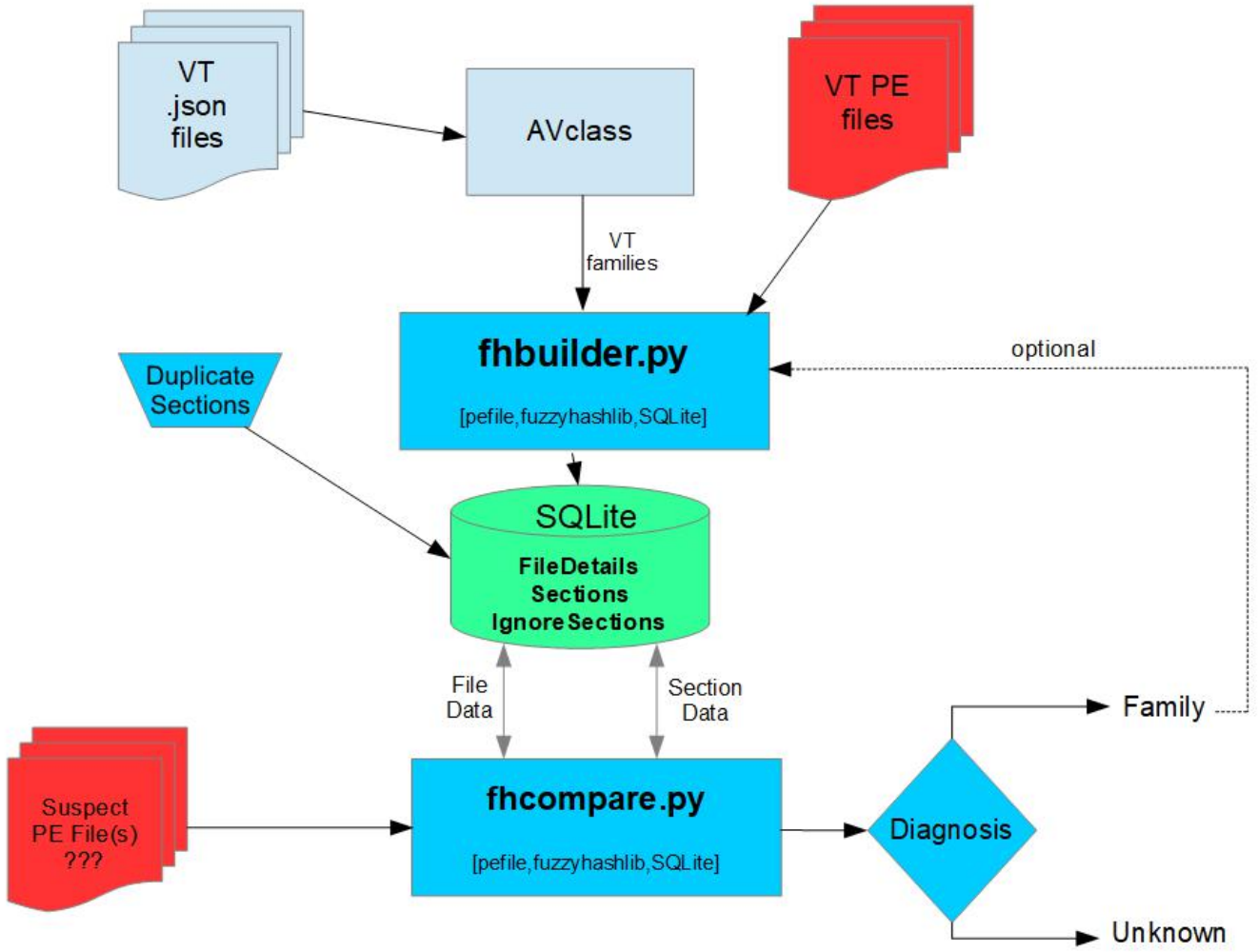
Methodology (2)

- Build database of malware families with file and section ssdeep digests.
- Sample size – 4 different families, 100 files per family.
- Experiments – Use the tools with the 4 families for 1) unobfuscated and 2) packed malware.
- Results – Compare file and section method performance for the 4 families.

Third party libraries and programs

Name	Purpose
Fuzzyhashlib	Generate & compare ssdeep hashes
Pefile	Decompose PE file into section
Malicia/AvClass	Mass malware labeller
File_entropy	Calculate file entropy
SQLite	Database library
UPX/ASPack	Executable packers

Overview


VT
.json
files
AVclass
VT PE
files
VT
families
fhbuilder.py
[pefile,fuzzyhashlib,SQLite]
optional
Duplicate
Sections
SQLite
FileDetails
Sections
IgnoreSections
File
Data
Section
Data
Family
Suspect
PE File(s)
???
fhcompare.py
[pefile,fuzzyhashlib,SQLite]
Diagnosis
Unknown

Malware families used

- Virut – trojan – opens a “back door” on the compromised computer.
- Wabot – worm, controllable via IRC.
- Virlock – Ransomware.
- Coinminer – Cryptocurrency block generator.

Selection of samples.

- 100 random unobfuscated samples of each of the four families in the reference dataset and database were selected by calculating the entropy of the PE file (packed files tend to have higher entropy).
- 100 PE files belonging to a mix of families excluding the 4 chosen in the previous slide were selected. These were used to measure True Negative and False Positive rates.
- The effect of obfuscation was evaluated by packing the above two sets with UPX and ASPack.

Experiments (1)

The 100 PE files from each of the four families were used as input to the comparison tool at thresholds of 1, 30, 40, 50, 60 & 75 in turn and the results were recorded as follows:-

- True Positive: The tool correctly predicted the malware family of the input PE file
- False Negative: The tool incorrectly predicted that the input PE file was a member of a different family that it actually was.
- Unknown: The tool could not determine the malware family to which the input PE file belonged.

Experiments (2)

The 100 mixed-family samples were used to provide true negatives and false positives. i.e.

- True Negative: The tool correctly predicted that the input PE file was NOT a member of the family.
- False Positive: The tool incorrectly predicted that the input PE file WAS a member of the family being examined when in fact, it was not.
- Unknown: The tool could not determine the malware family to which the input PE file belonged.

Experimental Findings

- Initial results disappointing but…
- Some sections (and therefore their cryptographic hashes) are **identical** in multiple families. e.g. “empty string” hash occurred 13,330 times across 204 families.
- “Correct” but unwarranted similarity.
- Answer – exclude them!
- Result (at t = 50)
  - Before: 74 correct, 24 incorrect, 2 unknowns
  - After: 94 correct, 4 incorrect, 2 unknowns

Performance Metrics – unobfuscated malware

Recall = TP/(TP + FN) – Ability to find malware (relevance)

Precision = TP/(TP+FP) – Ability to correctly classify malware

Accuracy = TP+TN/(TP+TN+FP+FN) – Ratio of correct predictions to all predictions.

Note – Unknowns are counted as 'False'. As unknowns increase with threshold this explains the counter-intuitive increase of 'FP' rate with threshold.

The minimum threshold is set to 1 to select only sections where the similarity score exceeds zero (no similarity)

Table 6. Non-obfuscated file-level ssdeep results.

T	TP	FN	TN	FP	Prec.	Recall	Acc.
1	**187**	**213**	**374**	**26**	**0.88**	**0.47**	**0.70**
30	182	218	374	26	0.88	0.46	0.70
40	164	236	370	30	0.85	0.41	0.67
50	139	261	370	30	0.82	0.35	0.64
60	120	280	366	34	0.78	0.30	0.61
75	97	303	360	40	0.71	0.24	0.57

Table 7. Non-obfuscated section level ssdeep results.

T	TP	FN	TN	FP	Prec.	Recall	Acc.
1	**358**	**42**	**396**	**4**	**0.99**	**0.90**	**0.94**
30	**358**	**42**	**395**	**5**	**0.99**	**0.90**	**0.94**
40	352	48	395	5	0.99	0.88	0.93
50	347	53	394	6	0.98	0.87	0.93
60	337	63	392	8	0.98	0.84	0.91
75	333	67	387	13	0.96	0.83	0.90

Performance Metrics – Packed malware

Recall = TP/(TP + FN) – Ability to find malware (relevance)

Precision = TP/(TP+FP) – Ability to correctly classify malware

Accuracy = TP+TN/(TP+TN+FP+FN) – Ratio of correct predictions to all predictions.

Unknowns are counted as 'False'. As unknowns increase with threshold this explains the counter-intuitive increase of 'FP' rate with threshold.

Table 8. Packed file-level ssdeep results.

T	TP	FN	TN	FP	Prec.	Recall	Acc.
1	**171**	**229**	**120**	**280**	**0.38**	**0.43**	**0.36**
30	162	238	119	281	0.37	0.41	0.35
40	150	250	109	291	0.34	0.38	0.32
50	148	252	96	304	0.33	0.37	0.31
60	136	264	86	314	0.30	0.34	0.28
75	114	286	58	342	0.25	0.29	0.22

Table 9. Packed section-level ssdeep results.

T	TP	FN	TN	FP	Prec.	Recall	Acc.
1	**322**	**78**	**322**	**78**	**0.81**	**0.81**	**0.81**
30	320	80	322	78	0.80	0.80	0.80
40	317	83	309	91	0.78	0.79	0.78
50	317	83	295	105	0.75	0.79	0.77
60	323	77	256	144	0.69	0.81	0.72
75	232	168	182	218	0.52	0.58	0.52

Conclusions

- Similarity digests (or fuzzy hashes) can be used to identify malware families in variants of unobfuscated and packed malware.
- Section-level is superior for both types returning 93% more true positives than file-level. When unknowns are treated as incorrect the section level is again superior typically returning 2 to 4 times less incorrect results than the file method.
- Section level hashing is more robust than file level against obfuscation by packers.
- Exclusion of sections common to multiple families is necessary but not possible with file-level digests.

Future Work

- Extending the framework to include the sdhash and tlsh algorithms and comparing the results to ssdeep.
- Extending the work done by Namanya et al. to include section hashing with ssdeep and sdhash.
- Investigating which PE sections yield best results.

References

[1] Frank Breitinger and Harald Baier. Properties of a similarity preserving hash function and their realization in sdhash. *2012 Information Security for South Africa,* page 1, 2012.

[2] Jason Upchurch and Xiaobo Zhou. Variant: a malware similarity testing framework*. 2015 10th International Conference on Malicious Unwanted Software (MALWARE)*, page 31, 2015.

[3] Anitta Patience Namanya, Qublai Khan Ali Mirza, Hamad Al Mohannadi, Irfan U Awan, and Jules Ferdinand Pagna Disso. Detection of malicious portable executables using evidence combinational theory with fuzzy hashing. In *Future Bibliography 99 Internet of Things and Cloud (FiCloud), 2016 IEEE 4th International Conference on*, pages 91-98. IEEE, 2016.

[4] Forensic malware analysis: The value of fuzzy hashing algorithms in identifying similarities. *2016 IEEE Trustcom /BigDataSE/ISPA, Trustcom/BigDataSE/ISPA, 2016 IEEE, TRUSTCOM-BIGDATASE-ISPA*, page 1782, 2016.

[5] F. Pagani, M. Dell'Amico and D. Balzarotti, "Beyond Precision and Recall: Understanding Uses (and Misuses) of Similarity Hashes in Binary Analysis" in *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy,* pp. 354-365,2018.

[6] Understanding PE Structure, The Layman's Way, 10 May 2018. URL https://tech-zealots.com/malware-analysis/pe-portable-executable-structure-malware-analysis-part-2/. Date Retrieved 16 Aug 2018.

Thank you