

“FUZZY CLARITY”
Using Fuzzy Hashing Techniques
to Identify Malicious Code



Table of Contents

Overview.....	3
The Malware Threat.....	4
What is Fuzzy Hashing?	5
Fuzzy Hash Test #1: Similarities by Packers.....	7
Fuzzy Test #2: Malware Family by Similarities.....	12
Fuzzy Test #3: Similarities within Malware Families	15
Final Thoughts and Conclusions.....	17
Further Research	17
References.....	18

Overview

Fuzzy hashing is a concept which involves the ability to compare two distinctly different items and determine a fundamental level of similarity (expressed as a percentage) between the two. This paper explores the possibility of utilizing fuzzy hashing as a means for identifying similarities in emerging families of malicious code.

The Malware Threat

Advances in malicious code development have introduced robust plug-and-play malware development platforms. According to VirusTotal.com, the last 24 hours have seen almost 10,000 infected files submitted to their site (shown below). As the global community becomes more and more interconnected and the ease and speed of malware development decreases, new malicious code will continue to grow at an alarming rate.

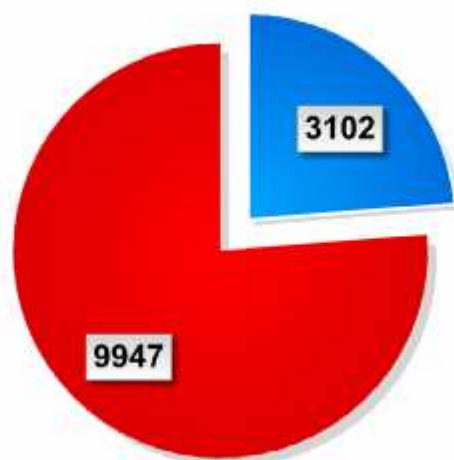


Figure 1: VirusTotal.com 24-Hour Results (Red = Infected)

In such a hostile environment, antivirus defenses have become an absolute necessity. The hard work of the companies involved in antivirus defense reaches far and wide in efforts to protect our servers and our PCs. Malicious code researchers, however, are left in a bind. Due to the wildly inconsistent naming convention of different antivirus vendors, it is necessary to utilize a small army of antivirus tools to rapidly identify as many different samples of malicious code as possible. Sometimes several tools will identify a malicious code sample by similar names. Sometimes they will be completely different. At still other times, none of the tools in our arsenal will identify a new malicious sample. Using fuzzy hashing techniques as an additional layer on top of an existing antivirus infrastructure, security researchers can more quickly tie together different forms of malicious code.

What is Fuzzy Hashing?

The concept of fuzzy hashing stemmed from Dr. Andrew Tridgell's work called "spamsum". In attempting to develop a means to identify commonalities in email-based spam messages, Dr. Tridgell developed a signature algorithm with a number of unique properties. Specifically, he identified these unique properties as described below:

"non-propagation: In most hash algorithms a change in any part of a plaintext will either change the resulting hash completely or will change all parts of the hash after the part corresponding with the changed plaintext. In the spamsum algorithm only the part of the spamsum signature that corresponds linearly with the changed part of the plaintext will be changed. This means that small changes in any part of the plaintext will leave most of the signature the same. This is essential for SPAM detection as it is common for variants of the same SPAM to have small changes in their body and we need to ensure that the matching algorithm can cope with these changes.

Alignment robustness: Most hash algorithms are very alignment sensitive. If you shift the plaintext by a byte (say by inserting a character at the start) then a completely different hash is generated. The spamsum algorithm is robust to alignment changes, and will automatically re-align the resulting signature after insertions or deletions. This works in combination with the non-propagation property to make spamsum suitable for telling if two emails are 'similar'.

The core of the spamsum algorithm is a rolling hash similar to the rolling hash used in 'rsync'. The rolling hash is used to produce a series of 'reset points' in the plaintext that depend only on the immediate context (with a default context width of seven characters) and not on the earlier or later parts of the plaintext. A stronger hash based on the FNV algorithm is then used to produce hash values of the areas between two reset points. The resulting signature comes from the concatenation of a single character from the FNV hash per reset point."

(Tridgell. 2003.)

Building on this concept of a rolling hash-based algorithm, in 2006 Jesse Kornblum developed an open source tool known as ssdeep. According to the ssdeep website:

"[Ssdeep] computes a checksum based on context triggered piecewise hashes for each input file. If requested, the program matches those checksums against a file of known checksums and reports any possible matches."

(Kornblum. 2006.)

The ssdeep tool, which shall be the primary tool used for experimentation throughout this paper, returns a similarity percentage between two files. The higher the percentage reported by ssdeep, the more similar the two pieces of code. Thus a 100% would indicate that all of the code in one file is also found in the other; while a 50% similarity would imply that only half of the code found in both samples was found to be similar.

Fuzzy Clarity: Using Fuzzy Hashing Techniques to Identify Malicious Code

Fuzzy Hash Test #1: Similarities by Packers

Test Objective: Determine the effect of various binary packers on a sample set of known good executables.

Test Information: Malicious code developers utilize many different techniques to obfuscate their binary code. This increases the difficulty of detecting the true intention of malicious code analysts. These techniques run the gamut from simple XOR encoding to more complex virtual machine-based encryption. This test will examine the effect on fuzzy hash similarity when utilizing different binary packing tools and techniques.

Sample Base: The sample base for this test included no malicious code. Instead, several Windows 32-bit portable executable applications demonstrating functions often seen in malicious code were selected. The names of all tools utilized in these tests are hidden to focus more on specific results.

Application Number	Application Category
1	Command Shell Utility #1
2	Web Browser #1
3	Web Browser #2
4	Internet Relay Chat Client #1
5	Command Shell Utility #2
6	TCP/IP-based Scanning Utility
7	TCP/IP-based Sniffing Utility

Table 1: Application Samples for Test #1

Binary packing utilities were selected to represent a range of techniques typically seen in use in the wild today. In all cases, default options were used. In cases where options were required, such as the use of encryption keys, identical keys were used across the board.

Packer Number	Reported Packing Capabilities
1	Compression
2	Compression
3	Compression, Extensible, Software Protection
4	Compression
5	Anti-crack, License Management, Wrapper
6	Software Protection, Encryption, Compression, License Management
7	Polymorphic Engine, License Manager, Anti-debugging Techniques

Table 2: Packing Utilities for Test #1

Method:

1. Each binary in the sample base was compared utilizing the ssdeep utility.
2. A copy of each binary in the sample was made.
3. Each application copy was packed with packer #1.
4. All packed and unpacked binaries were compared utilizing the ssdeep utility.
5. Steps 2-4 were repeated for each of the remaining 6 packing utilities.

Data:

Application 1: Command Shell Utility #1

	A1	A1-P1	A1-P2	A1-P3	A1-P4	A1-P5	A1-P6	A1-P7
A1	100							72
A1-P1		100						
A1-P2			100					
A1-P3				100				
A1-P4					100			
A1-P5						100		
A1-P6							100	
A1-P7	72							100

Table 3: Application 1 Similarity Results By Packer (Test 1)

Analysis: Across six of the seven tested binary packing utilities, only one demonstrated any fuzzy hashing similarity to the original file.

Application 2: Web Browser #1

	A2	A2-P1	A2-P2	A2-P3	A2-P4	A2-P5	A2-P6	A2-P7
A2	100							
A2-P1		100						
A2-P2			100					
A2-P3				100				
A2-P4					100			
A2-P5						100		
A2-P6							100	
A2-P7								100

Table 4: Application 2 Similarity Results By Packer (Test 1)

Analysis: Across all seven tested binary packing utilities, no fuzzy hash similarity to the original file was demonstrated.

Application 3: Web Browser #2

	A3	A3-P1	A3-P2	A3-P3	A3-P4	A3-P5	A3-P6	A3-P7
A3	100	55	79	57	77		43	47
A3-P1	55	100	55	57	54			
A3-P2	79	55	100	58	88		55	66
A3-P3	57	57	58	100	61		35	40
A3-P4	77	54	88	61	100		54	55
A3-P5						100		
A3-P6	43		55	35	54		100	54
A3-P7	47		66	40	55		54	100

Table 5: Application 3 Similarity Results By Packer (Test 1)

Analysis: Packed versions one, two, three, four, six and seven of application three demonstrate between a 40-79% similarity match. Packer number five, reporting anti-crack, license management and wrapping capabilities demonstrated no similarity to any packed or unpacked versions of application three.

Application 4: Internet Relay Chat Client

	A4	A4-P1	A4-P2	A4-P3	A4-P4	A4-P5	A4-P6	A4-P7
A4	100							
A4-P1		100						
A4-P2			100					
A4-P3				100				
A4-P4					100			
A4-P5						100		
A4-P6							100	
A4-P7								100

Table 6: Application 4 Similarity Results By Packer (Test 1)

Analysis: Across all seven tested binary packing utilities, no fuzzy hash similarity to the original file was demonstrated.

Application 5: Command Shell Utility #2

	A5	A5-P1	A5-P2	A5-P3	A5-P4	A5-P5	A5-P6	A5-P7
A5	100							
A5-P1		100						
A5-P2			100					
A5-P3				100				
A5-P4					100			
A5-P5						100		
A5-P6							100	
A5-P7								100

Table 7: Application 5 Similarity Results By Packer (Test 1)

Analysis: Across all seven tested binary packing utilities, no fuzzy hash similarity to the original file was demonstrated.

Application 6: TCP/IP-Based Scanning Utility

	A6	A6-P1	A6-P2	A6-P3	A6-P4	A6-P5	A6-P6	A6-P7
A6	100							30
A6-P1		100						
A6-P2			100					

Fuzzy Clarity: Using Fuzzy Hashing Techniques to Identify Malicious Code

A6-P3				100				
A6-P4					100			
A6-P5						100		
A6-P6							100	
A6-P7	30							100

Table 8: Application 6 Similarity Results By Packer (Test 1)

Analysis: Across six of the seven tested binary packing utilities, only one demonstrated any fuzzy hashing similarity to the original file.

Application 7 TCP/IP-Based Sniffing Utility

	A7	A7-P1	A7-P2	A7-P3	A7-P4	A7-P5	A7-P6	A7-P7
A7	100							75
A7-P1		100						
A7-P2			100					
A7-P3				100				
A7-P4					100			
A7-P5						100		
A7-P6							100	
A7-P7	75							100

Table 9: Application 7 Similarity Results By Packer (Test 1)

Analysis: Across six of the seven tested binary packing utilities, only one demonstrated any fuzzy hashing similarity to the original file.

Packer 5 Similarities

	A1-P5	A2-P5	A3-P5	A4-P5	A5-P5	A6-P5	A7-P5	A8-P5
A1-P5	100		41		43	43		
A2-P5		100						
A3-P5	41		100		49	41		
A4-P5				100				
A5-P5	43		49		100	40		
A6-P5	43		41		40	100		
A7-P5							100	
A8-P5								100

Table 10: Packer 5 Similarity Results By Application (Test 1)

Analysis: Applications one, three, five and six, when packed with packer number five, demonstrated between a 40-49% similarity. This would seem to indicate that some level of the resulting packed code is nearly identical across multiple samples.

Packer 6 Similarities

	A1-P6	A2-P6	A3-P6	A4-P6	A5-P6	A6-P6	A7-P6	A8-P6
--	-------	-------	-------	-------	-------	-------	-------	-------

Fuzzy Clarity: Using Fuzzy Hashing Techniques to Identify Malicious Code

A1-P6	100		63		54	43		
A2-P6		100						
A3-P6	63		100		66	47		
A4-P6				100				
A5-P6	54		66		100	43		
A6-P6	43		47		43	100		
A7-P6							100	
A8-P6								100

Table 11: Packer 6 Similarity Results By Application (Test 1)

Analysis: Applications one, three, five and six, when packed with packer number six, demonstrated between a 43-63% similarity. This would seem to indicate that some level of the resulting packed code is nearly identical across multiple samples.

Conclusions:

1. In 5.6% of tested cases, a certain application X, when packed with packer Y, resulted in a slight to moderate fuzzy hash similarity to other packed versions of the same application and/or the original executable.
 - a. Implication: Fuzzy hashing does not appear to be a good solution for detection of malicious code similarities across different packing utilities.
2. In 3.6% of tested cases, a certain packer Y demonstrated slight levels of similarities to another application also packed with the same binary packing utility.
 - a. Implication: Fuzzy hashing does not generally appear to identify similarities across different applications packed with the same packer.
 - b. Implication: When similarities are identified based upon a given packer, fuzzy hashing techniques do not appear to indicate a high level of similarity across multiple applications.
 - c. Implication: Moderate levels of confidence (< 66%) are not enough to uniquely identify a particular application and may be triggered by packer similarities.

Fuzzy Test #2: Malware Family by Similarities

Test Objective: Determine whether antivirus results identify as members of the family a set of malicious code identified as similar when utilizing fuzzy hashing techniques.

Test Information: This test will utilize a small collection of malicious code to examine various antivirus vendor results when fuzzy hashing techniques identify a set of malicious code as similar.

Sample Base: This test began with a small collection of 5,000 samples of malicious code provided to the author for the purposes of this test. No prior knowledge of the contents of the sample was provided. All files were named only by their MD5 cryptographic hash.

A fuzzy hash comparison across all 5,000 malicious files identified grouping as large as 108 files which shared some level of similarity (69-83%). A single file was chosen, along with the top 30 similar files for analysis. These 31 files were then scanned with four different anti-virus utilities to attempt to determine whether various vendors have defined these tools as similar.

Again, the names of the files, anti-virus tools, and specific virus detection names are masked to focus primarily on the results of the testing.

Anti-Virus Tool	Tool Category
1	Commercial Antivirus Vendor #1
2	Free Antivirus Vendor #1
3	Commercial Antivirus Vendor #2
4	Free Antivirus Vendor #2

Table 12: Antivirus Utilities

Method:

1. Create fuzzy hash list from 5000 files.
2. Select one file from list.
3. Conduct fuzzy hash matching between 5000 files and single selected file.
4. Gather top 30 similar files.
5. Conduct anti-virus scans on all 31 files.

Data:

File	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Subject	83	80	80	83	82	80	80	80	85	85	82	82	82	80	82

Table 13: Top 30 File Matches by Percentage Similarity (Test 2)

File	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Subject	80	82	80	80	80	80	80	82	80	83	83	82	85	82	82

Table 14: Top 30 File Matches by Percentage Similarity (cont'd) (Test 2)

Analysis: Out of 5000 files, 107 files were identified as similar using fuzzy hash techniques. The 30 most similar files, each having a different MD5 hash, shared between 80-85% identical code.

Fuzzy Clarity: Using Fuzzy Hashing Techniques to Identify Malicious Code

File	AV1	AV2	AV3	AV4
Subject	Virus 1	Virus 2	Undetected	Undetected
1	Virus 1	Virus 2	Undetected	Undetected
2	Virus 1	Virus 2	Undetected	Undetected
3	Virus 1	Virus 2	Undetected	Undetected
4	Virus 1	Virus 2	Undetected	Undetected
5	Virus 1	Virus 2	Undetected	Undetected
6	Virus 1	Virus 2	Undetected	Undetected
7	Virus 1	Virus 2	Undetected	Undetected
8	Virus 1	Virus 2	Undetected	Undetected
9	Virus 1	Virus 2	Undetected	Undetected
10	Virus 1	Virus 2	Undetected	Undetected
11	Virus 1	Virus 2	Undetected	Undetected
12	Virus 1	Virus 2	Undetected	Undetected
13	Virus 1	Virus 2	Undetected	Undetected
14	Virus 1	Virus 2	Undetected	Undetected
15	Virus 1	Virus 2	Undetected	Undetected
16	Virus 1	Virus 2	Undetected	Undetected
17	Virus 1	Virus 2	Undetected	Undetected
18	Virus 1	Virus 2	Undetected	Undetected
19	Virus 1	Virus 2	Undetected	Undetected
20	Virus 1	Virus 2	Undetected	Undetected
21	Virus 1	Virus 2	Undetected	Undetected
22	Virus 1	Virus 2	Undetected	Undetected
23	Virus 1	Virus 2	Undetected	Undetected
24	Virus 1	Virus 2	Undetected	Undetected
25	Virus 1	Virus 2	Undetected	Undetected
26	Virus 1	Virus 2	Undetected	Undetected
27	Virus 1	Virus 2	Undetected	Undetected
28	Virus 1	Virus 2	Undetected	Undetected
29	Virus 1	Virus 2	Undetected	Undetected
30	Virus 1	Virus 2	Undetected	Undetected

Table 15: Antivirus Detection Results of Similar Files (Test 2)

Analysis: Of the 31 files scanned, commercial scanner one identified all 31 as virus 1. Similarly, free antivirus scanner one detected all 31 files as malicious, but part of a different virus family. Neither commercial antivirus scanner number two, nor free antivirus scanner number two detected any of the malicious code. All similar files, when detected by an antivirus scanner, were identified by the same virus name.

Conclusion:

1. A single piece of malware, when compared against a list of fuzzy hashes of known threats can demonstrate similarities which fall within a grouping of known virii with a high level of confidence.
 - a. Implication: In the event of a new malicious code sample, which has not yet been identified by antivirus vendors, it may be possible to determine a virus family based upon historical fuzzy hash data.

Fuzzy Clarity: Using Fuzzy Hashing Techniques to Identify Malicious Code

- b. Shortfall: A list of known malicious code fuzzy hash values must exist in order for this technique to be successful.

Fuzzy Test #3: Similarities within Malware Families

Test Objective: Determine whether fuzzy hashing techniques can accurately identify members of a family of malicious code identified.

Test Information: This test will utilize a small collection of malicious code identified as being part of a family to examine fuzzy hashing results of a known family.

Sample Base: This test began with a small collection of 5,000 samples of malicious code provided to the author for the purposes of this test. No prior knowledge of the contents of the sample was provided. All files were named only by their MD5 cryptographic hash.

An antivirus scan was performed across all 5,000 malicious files which identified 4 distinct virus families, with one family having two variants. 17 files remained undetected. From this list, one virus family was selected which contained 440 files bearing distinct cryptographic hashes. The selected virus family was not identical to the set chosen for test #2.

Method:

1. Scan all 5,000 files with an antivirus scanner.
2. Identify a unique grouping of malicious files.
3. Conduct fuzzy hash matching comparisons.

Data: (See table data on following page)

Analysis: Of the 440 malicious files identified as being part of the same virus family, only 33 files demonstrated some degree of similarity. These 33 files ranged between 30-86% similarity, with a median of 67%. The data shows approximately four different groupings of similar files within this family of detections.

Conclusion:

1. Malicious code within a given family (as identified by antivirus vendors) does not necessarily demonstrate a highly level of similarity. Malicious code samples within a given family may vary wildly.
 - a. Implication: Levels of confidence over 69% may serve as a relevant baseline for identifying a particular family of malicious code to which a particular sample belongs. Further testing should be conducted on a larger sample base to confirm.
 - c. Shortfall: A list of known malicious code fuzzy hash values must exist in order for this technique to be successful.

Fuzzy Clarity: Using Fuzzy Hashing Techniques to Identify Malicious Code

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33		
1	100	63	60		66	61	66		60	68				65	68	65	68	66		66	68	69		65	65	62	68	49	68			68		1	
2	63	100	68		71	68	74		74	68				65	75	72	79	72		74	75	74		71	75	72	82	49	72			65		2	
3	60	68	100		69	58	66		69	68				57	72	65	69	61		63	69	72		65	71	63	66	49	68			61		3	
4				100							41		44						41												41		43		4
5	66	71	69		100	68	71		74	72				69	80	77	80	71		74	75	75		72	77	71	75	50	80			75		5	
6	61	68	58		68	100	68		71	63				63	69	69	72	65		68	72	71		66	69	74	72	47	72			63		6	
7	66	74	66		71	68	100		79	75				63	80	77	75	71		74	75	75		72	74	71	75	47	74			68		7	
8								100				77											69						30						8
9	60	74	69		74	71	79		100	72				69	75	77	75	68		74	80	75		69	80	71	75	50	80			66		9	
10	68	68	68		72	63	75		72	100				61	80	71	77	69		72	74	75		68	74	66	74	49	74			69		10	
11				41							100		40						36												41		40		11
12								77				100																	30						12
13				44							40		100						36												38		38		13
14	65	65	57		69	63	63		69	61				100	68	68	68	66		66	68	66		65	68	66	68	47	65			68		14	
15	68	75	72		80	69	80		75	80				68	100	83	82	72		75	86	80		74	83	75	82	50	79			77		15	
16	65	72	65		77	69	77		77	71				68	83	100	79	77		74	83	77		74	75	74	79	47	80			71		16	
17	68	79	69		80	72	75		75	77				68	82	79	100	75		75	86	80		74	83	79	86	47	79			74		17	
18	66	72	61		71	65	71		68	69				66	72	77	75	100		71	75	71		72	71	71	79	47	74			69		18	
19				41							36		36						100												41		43		19
20	66	74	63		74	68	74		74	72				66	75	74	75	71		100	75	79		66	77	68	80	50	77			69		20	
21	68	75	69		75	72	75		80	74				68	86	83	86	75		75	100	80		74	83	75	82	47	83			71		21	
22	69	74	72		75	71	75		75	75				66	80	77	80	71		79	80	100		69	77	74	85	49	82			72		22	
23								69															100												23
24	65	71	65		72	66	72		69	68				65	74	74	74	72		66	74	69		100	71	69	71	46	71			68		24	
25	65	75	71		77	69	74		80	74				68	83	75	83	71		77	83	77		71	100	74	79	47	80			71		25	
26	62	72	63		71	74	71		71	66				66	75	74	79	71		68	75	74		69	74	100	75	49	71			66		26	
27	68	82	66		75	72	75		75	74				68	82	79	86	79		80	82	85		71	79	75	100	47	79			71		27	
28	49	49	49		50	47	47	30	50	49		30		47	50	47	47	47		50	47	49	35	46	47	49	47	100	49	30		46		28	
29	68	72	68		80	72	74		80	74				65	79	80	79	74		77	83	82		71	80	71	79	49	100			71		29	
30																													30		100				30
31				41							41		38						41												100		40		31
32	68	65	61		75	63	68		66	69				68	77	71	74	69		69	71	72		68	71	66	71	46	71			100		32	
33				43							40		38						43												40		100		33

Table 16: Similarity within Malware Family (Test 3)

Final Thoughts and Conclusions

Fuzzy hashing techniques are not the silver bullet to save the world from computer security threats. However, as compared to positional-based cryptographic fingerprinting algorithms such as MD5 and SHA-1, fuzzy hashing techniques are a significant step in the direction of more quickly identifying new samples of malicious code. Research data shows that it is possible to identify a group of similar malicious files by utilizing these techniques.

Binary packers, on average, prove troublesome for fuzzy hashing techniques. Multiple copies of a single source, packed with different binary packers do not necessarily resemble either each other, or the original file. Thus, the overall effectiveness of fuzzy hashing is reduced when varied packers are used.

In order to facilitate fuzzy hash usage with any degree of effectiveness, a source of fuzzy hashes must be identified. The sharing of fuzzy hash values for newly discovered malicious code samples by security researchers could allow for an increase in overall detection and identification abilities, without requiring these individuals and groups to specifically share malicious code samples.

Overall, fuzzy hashing techniques for malware detection and identification are believed to net an increase in information security posture. They are simple to implement and simple to share. While the results of this research are not conclusive given the sample sizes in use, the basic concepts have proven sound.

Further Research

Integration of fuzzy hashing techniques into a larger collection and analysis infrastructure could potentially draw correlations to other security-related data points. Some examples of potential data points for research might be:

- Malware similarity by geographical distribution
- Social networking inferences of similar malware
- Botnet family similarity
- Malware similarity by phishing event
- Malware similarity by e-fraud event.

References

Kornblum, Jesse. (2006). *Ssdeep*. <http://ssdeep.sourceforge.net/manpage.html>

Shadowserver Foundation. (2007). <http://www.shadowserver.org>

Tridgell, Dr. Andrews. (2003). *SpamSum*.
<http://samba.org/ftp/unpacked/junkcode/spamsum/README>

Virustotal. (2007). <http://www.virustotal.com>