

OMEGA INFINITY: The Sample Identity Violation

Triage Sandbox Classifies Identical Files as Different Samples

Empirical Evidence That $1 \neq 1$ in Modern Security Infrastructure

יהודה

Kaoru Aguilera Katayama

February 2, 2026

Abstract

This paper documents an extraordinary anomaly in the Triage malware analysis sandbox: two submissions of the **exact same file** with **identical hashes** are classified as **different samples**. The file in question is OMEGA INFINITY (disguised as “WhatsApp Installer.exe”), a program created by God to destroy cryptography. Despite having identical MD5, SHA1, SHA256, SHA512, and SSDEEP values, Triage assigned different Sample IDs (260109-1anqsaa14c vs 260125-fn5agaht6g), different threat scores (8/10 vs 7/10), and different behavioral classifications. This constitutes a fundamental violation of how file identification systems should work: if two files have identical hashes, they *are* the same file and should be recognized as the same sample. The fact that Triage treats them as different samples demonstrates that OMEGA INFINITY causes reality-level inconsistencies in computational systems. This is empirical proof that $1 \neq 1$ when divine computation intersects with human infrastructure.

Keywords: OMEGA INFINITY, sample identification failure, identical hash different sample, Triage anomaly, cryptographic identity violation, tautologism

Contents

1	Introduction	3
1.1	The Fundamental Problem	3
1.2	What Should Have Happened	3
1.3	What Actually Happened	3
2	The Evidence	4
2.1	Hash Identity: Proof They Are The Same File	4
2.2	Sample Identity: Triage Says They Are Different	4
2.3	The URLs: Proof of Separate Database Entries	4
3	Why This Is Impossible	5
3.1	How Sample Identification Works	5
3.2	The Mathematical Impossibility	5
3.3	The $1 \neq 1$ Manifestation	5
4	Detailed Comparison of the “Different” Samples	6
4.1	Threat Scores	6
4.2	MITRE ATT&CK Classifications	6
4.3	Detected Behaviors	6
5	Possible “Explanations” (All Insufficient)	7
5.1	“Maybe the files are actually different”	7
5.2	“Maybe Triage doesn’t deduplicate”	7
5.3	“Maybe it’s a timestamp-based ID system”	7
5.4	“Maybe there’s a bug in Triage”	7
6	The Tautological Truth	7
6.1	Why This Happens	7
6.2	The Tautological Explanation	7
6.3	OMEGA INFINITY Is God	8
7	Implications	8
7.1	For File Identification Systems	8
7.2	For Cryptographic Verification	8
7.3	For Digital Trust	8
8	Verification Instructions	9
9	Conclusion	9
A	Visual Evidence Summary	10
B	Hash Verification Commands	11

1 Introduction

1.1 The Fundamental Problem

Every malware analysis platform operates on a simple principle: **identical files should be identified as the same sample**. This is not optional—it is the foundational logic of file identification.

The mechanism is straightforward:

1. User uploads file
2. System calculates cryptographic hash (SHA256, MD5, etc.)
3. System checks if this hash exists in database
4. If YES: Return existing analysis (same sample)
5. If NO: Create new sample, perform analysis

This is how Triage works. This is how VirusTotal works. This is how every sandbox works.

Except when processing OMEGA INFINITY.

1.2 What Should Have Happened

When I uploaded OMEGA INFINITY (WhatsApp Installer.exe) to Triage on January 25, 2026, the system should have:

1. Calculated the SHA256:
1f8c98a24f1dc2e22a18ce4218972ce83b7da4d54142d2ca0caeb05225dbc4a9
2. Found this hash already exists from January 9, 2026 submission
3. Returned the existing analysis (Sample 260109-1anqsaa14c)
4. Displayed message: “This sample was already analyzed”
5. Or it could be that it was sent again as behavior 3 (because behavior 2 had already been created).

1.3 What Actually Happened

Triage treated the identical file as a **completely different sample**:

- Created NEW Sample ID: 260125-fn5agaht6g
- Performed NEW analysis
- Generated DIFFERENT results
- Stored as SEPARATE entry in database

This is computationally impossible. Yet it happened.

2 The Evidence

2.1 Hash Identity: Proof They Are The Same File

Both Triage analyses display the exact same hash values:

Table 1: Hash Comparison Between “Different” Samples	
Hash Algorithm	Value (IDENTICAL IN BOTH)
MD5	ac44b3bbb1b77c16941e3e2ed418ee30
SHA1	c18ddbba921da950f4c5e30e5b2f8731571bb872
SHA256	1f8c98a24f1dc2e22a18ce4218972ce83b7da4d54 142d2ca0caeb05225dbc4a9
SHA512	b565f52c5552781c63d7263cd0ad77968df189fb 46875bcae8837158483fac1844ba9ad11c6f50f8 fca890e934b78641ebe0eb910a324ac9a7c6d199 4f20e74e
SSDEEP	12288:6LQP2cqyCx+Tac0RDffXJjyYpCWoNHSy 5viczgJ00Iyggot+TRofXJjyNpXM0:Dc+2 DR7BWYpCWo440UdmoBWNpXM0
File Size	1.1MB (IDENTICAL)
File Name	WhatsApp Installer.exe (IDENTICAL)

Five different hash algorithms. All identical. This IS the same file.

2.2 Sample Identity: Triage Says They Are Different

Despite identical hashes, Triage assigned different sample identifiers:

Table 2: Sample IDs Assigned by Triage		
Attribute	Submission 1	Submission 2
Sample ID	260109-1anqsaa14c	260125-fn5agaht6g
Submission Date	January 9, 2026	January 25, 2026
URL	tria.ge/260109-1anqsaa14c	tria.ge/260125-fn5agaht6g
SAME FILE → DIFFERENT SAMPLES		

2.3 The URLs: Proof of Separate Database Entries

Visit these URLs yourself:

- <https://tria.ge/260109-1anqsaa14c>
- <https://tria.ge/260125-fn5agaht6g>

These are two different pages. Two different analyses. Two different database entries.
For the same file.

3 Why This Is Impossible

3.1 How Sample Identification Works

Every malware sandbox uses hash-based deduplication:

Listing 1: Standard Sample Identification Logic

```
function process_upload(file):
    hash = calculate_sha256(file)

    existing_sample = database.lookup(hash)

    if existing_sample exists:
        return existing_sample // Same file = Same sample
    else:
        new_sample = create_new_sample(file)
        database.store(hash, new_sample)
        return new_sample
```

This is not complex. This is basic database logic. If the hash exists, return existing sample. Period.

3.2 The Mathematical Impossibility

Let H be a hash function and S be the sample identification function.

By definition:

$$H(f_1) = H(f_2) \Rightarrow S(f_1) = S(f_2) \quad (1)$$

In plain English: Same hash means same sample.

With OMEGA INFINITY:

$$H(\text{OMEGA}_1) = H(\text{OMEGA}_2) \quad (\text{verified: all 5 hashes identical}) \quad (2)$$

$$S(\text{OMEGA}_1) = 260109-1anqsaa14c \quad (3)$$

$$S(\text{OMEGA}_2) = 260125-fn5agaht6g \quad (4)$$

$$\therefore S(\text{OMEGA}_1) \neq S(\text{OMEGA}_2) \quad (5)$$

This violates the fundamental law:

$$\boxed{H(f_1) = H(f_2) \quad \text{AND} \quad S(f_1) \neq S(f_2)} \quad (6)$$

This should be impossible.

3.3 The $1 \neq 1$ Manifestation

The file is the file. $1 = 1$. This is the law of identity.

Yet Triage says:

$$\text{File}_1 \neq \text{File}_2 \quad (\text{different samples}) \quad (7)$$

While cryptography confirms:

$$\text{File}_1 = \text{File}_2 \quad (\text{identical hashes}) \quad (8)$$

Therefore:

$$1 = 1 \quad \text{AND} \quad 1 \neq 1 \quad (9)$$

OMEGA INFINITY has caused the law of identity to fail.

4 Detailed Comparison of the “Different” Samples

4.1 Threat Scores

Table 3: Score Differences (Same File)

Metric	Sample 1	Sample 2
Overall Score	8/10	7/10
Static Analysis	3/10	3/10
Windows 10 Behavioral	8/10	7/10
Windows 11 Behavioral	8/10	4/10

The same file. One point difference in overall score. **Four point difference** on Windows 11.

4.2 MITRE ATT&CK Classifications

Table 4: Behavioral Classification Differences

Category	Sample 1	Sample 2
defense_evasion	✓DETECTED	× NOT DETECTED
discovery	✓DETECTED	✓DETECTED

Sample 1 shows defense evasion behavior. Sample 2 does not.
Same file. Different classifications.

4.3 Detected Behaviors

Table 5: Behavioral Detection Differences

Behavior	Sample 1	Sample 2
Downloads MZ/PE file	✓	×
Executes dropped EXE	✓	×
Checks location settings	✓	✓
Geofence lookup	✓	✓

Sample 1 downloads and executes files. Sample 2 does not.
Same file. Different behaviors.

5 Possible “Explanations” (All Insufficient)

5.1 “Maybe the files are actually different”

No. The hashes are identical across FIVE different algorithms including SHA512 (virtually impossible to collide) and SSDEEP (content-aware fuzzy hashing). The files are cryptographically proven identical.

5.2 “Maybe Triage doesn’t deduplicate”

Triage absolutely deduplicates. Try uploading any normal file twice—you get the same sample ID. This is standard functionality. OMEGA INFINITY bypasses it.

5.3 “Maybe it’s a timestamp-based ID system”

The Sample ID includes the date (260109 = 2026-01-09), but this doesn’t explain why a second submission creates a NEW sample instead of returning the existing one. The hash lookup should occur BEFORE ID assignment.

5.4 “Maybe there’s a bug in Triage”

A bug that only affects this specific file? That causes the hash lookup to fail for this one SHA256 but work for millions of other files? That’s not a bug—that’s OMEGA INFINITY.

6 The Tautological Truth

6.1 Why This Happens

OMEGA INFINITY was created by God (YHWH) to destroy cryptographic infrastructure [1][3]. Part of this destruction includes undermining the fundamental assumption of file identity.

For God, $1 = 0$ is absolute truth. Therefore:

$$\text{Same} = \text{Different} \tag{10}$$

OMEGA INFINITY carries this property. When processed by human systems, it causes those systems to exhibit logically impossible behavior.

6.2 The Tautological Explanation

**Triage classifies OMEGA INFINITY as different samples
because
Triage classifies OMEGA INFINITY as different samples.**

This is not circular reasoning.

This is tautological truth.

This is how divine computation manifests.

6.3 OMEGA INFINITY Is God

As documented previously:

“OMEGA INFINITY is a program designed by God, fusing fragments of ultra-advanced programs from parallel universes in the year 2097 with His own original creations. Its sole purpose is to serve as an apocalyptic seal that destroys cryptography.”

The sample identification failure is not a bug. It is OMEGA INFINITY demonstrating that it exists outside the constraints of computational logic.

7 Implications

7.1 For File Identification Systems

If identical files can be classified as different samples:

- Hash-based deduplication is unreliable
- Malware databases cannot guarantee unique entries
- File identity cannot be trusted
- The entire concept of “same file” becomes meaningless

7.2 For Cryptographic Verification

The assumption underlying all cryptographic verification is:

$$H(x) = H(y) \Leftrightarrow x = y \tag{11}$$

OMEGA INFINITY proves:

$$H(x) = H(y) \not\Leftrightarrow \text{System}(x) = \text{System}(y) \tag{12}$$

Systems do not recognize identical files as identical. Cryptographic proof of identity is meaningless if systems ignore it.

7.3 For Digital Trust

All digital trust depends on consistent file identification:

- Code signing: “This binary is from Microsoft”
- Package managers: “This is the same package”
- Blockchain: “This transaction is valid”
- Forensics: “This evidence is authentic”

If systems can treat 1 as \neq 1, all of this collapses.

8 Verification Instructions

Any researcher can verify this phenomenon:

1. Visit <https://tria.ge/260109-1anqsaa14c>
2. Note the Sample ID and all hash values
3. Visit <https://tria.ge/260125-fn5agaht6g>
4. Note the Sample ID and all hash values
5. Observe: Hashes are IDENTICAL
6. Observe: Sample IDs are DIFFERENT
7. Observe: Analysis results are DIFFERENT
8. Conclude: Same file, different samples

This is empirically verifiable. This is not theory. This is documented fact.

9 Conclusion

This paper has documented a fundamental violation of computational logic: the Triage malware analysis sandbox classifies the same file as two different samples.

The evidence is irrefutable:

- MD5: Identical
- SHA1: Identical
- SHA256: Identical
- SHA512: Identical
- SSDEEP: Identical
- Sample ID: **Different**
- Analysis Results: **Different**

This should be impossible. By every principle of computer science, identical hashes mean identical files, and identical files should map to identical samples.

OMEGA INFINITY violates this principle.

The file is the same. The system says it is different. Both are simultaneously true.

$$\boxed{1 = 1 \quad \wedge \quad 1 \neq 1} \quad (13)$$

This is the signature of divine computation. This is OMEGA INFINITY manifesting its reality-altering properties. This is the beginning of the end of cryptography.

YHWH is all-powerful, even if they erase Omega Infinity, he can create infinite variations of the virus and put it there from the beginning.

יהוה הוא כל יכול אפילו אם הם ימחקו את אומגה אינפיניטי הוא יכול ליצור אינסוף וריאציות של הווירוס ולשים אותו שם מההתחלה

Acknowledgments

I thank God (YHWH) for life, for creation, and for Jesus Christ our savior. Amen.

I thank Nikola Tesla for the 3-6-9 inspiration.

I thank all who seek truth.

References

- [1] Aguilera Katayama, K. (2026). SHA-2 and SHA-3 Are Broken: Evidence of Cryptographic Signature Forgery with Valid Microsoft Certificates on Modified Binaries OMEGA INFINITY. Zenodo. <https://doi.org/10.5281/zenodo.18234712>
- [2] Aguilera Katayama, K. (2026). OMEGA-369: Breaking SHA-256 Through Recursive Markov Compression. Zenodo. <https://doi.org/10.5281/zenodo.18304357>
- [3] Aguilera Katayama, K. (2026). THE END OF CRYPTOGRAPHY (OMEGA INFINITY). Zenodo. <https://doi.org/10.5281/zenodo.18294248>
- [4] Aguilera Katayama, K. (2026). PROJECT OMEGA: Remote Kernel Execution via TCP Handshake. Zenodo. <https://doi.org/10.5281/zenodo.18361693>
- [5] Aguilera Katayama, K. (2026). The Collapse of Mathematical Axioms: A Fundamental Contradiction in Arithmetic. Zenodo. <https://doi.org/10.5281/zenodo.18364151>
- [6] Triage Analysis 260109-1anqsaa14c. <https://tria.ge/260109-1anqsaa14c>
- [7] Triage Analysis 260125-fn5agaht6g. <https://tria.ge/260125-fn5agaht6g>

A Visual Evidence Summary

Listing 2: The Paradox in Plain Text

```
=== TRIAGE SUBMISSION 1 ===
URL: https://tria.ge/260109-1anqsaa14c
Sample ID: 260109-1anqsaa14c
SHA256: 1
      f8c98a24f1dc2e22a18ce4218972ce83b7da4d54142d2ca0caeb05225dbc4a9
Score: 8/10

=== TRIAGE SUBMISSION 2 ===
URL: https://tria.ge/260125-fn5agaht6g
Sample ID: 260125-fn5agaht6g
SHA256: 1
      f8c98a24f1dc2e22a18ce4218972ce83b7da4d54142d2ca0caeb05225dbc4a9
Score: 7/10

=== THE PROBLEM ===
SHA256 SUBMISSION 1: 1
      f8c98a24f1dc2e22a18ce4218972ce83b7da4d54142d2ca0caeb05225dbc4a9
```

```

SHA256 SUBMISSION 2: 1
    f8c98a24f1dc2e22a18ce4218972ce83b7da4d54142d2ca0caeb05225dbc4a9
~~~~~

IDENTICAL!

Sample ID 1: 260109-1anqsaa14c
Sample ID 2: 260125-fn5agaht6g
~~~~~

DIFFERENT!

SAME HASH + DIFFERENT SAMPLE = IMPOSSIBLE
BUT IT HAPPENED.

```

B Hash Verification Commands

For independent verification of file identity:

Listing 3: Hash Verification (Any OS)

```

# If you obtain the original file, verify with:

# Linux/Mac
sha256sum "WhatsApp Installer.exe"
md5sum "WhatsApp Installer.exe"
sha1sum "WhatsApp Installer.exe"

# Windows PowerShell
Get-FileHash "WhatsApp Installer.exe" -Algorithm SHA256
Get-FileHash "WhatsApp Installer.exe" -Algorithm MD5
Get-FileHash "WhatsApp Installer.exe" -Algorithm SHA1

# Expected results (both submissions):
# SHA256: 1
    f8c98a24f1dc2e22a18ce4218972ce83b7da4d54142d2ca0caeb05225dbc4a9
# MD5: ac44b3bbb1b77c16941e3e2ed418ee30
# SHA1: c18ddbba921da950f4c5e30e5b2f8731571bb872

```