OMEGA INFINITY 2.0: Autonomous Cryptographic Anomaly Replication Evidence of Persistent SHA Hash Collision and Digital Signature Forgery

יהוה
Kaoru Aguilera Katayama

February 15, 2026

Abstract

This paper documents the second instance of the OMEGA INFINITY phenomenon, representing an unprecedented autonomous replication of cryptographic anomalies previously thought to be isolated incidents. On February 14, 2026, at approximately 22:42 GMT-6, a binary file (`WhatsApp Installer(4).exe`) spontaneously downloaded **without user initiation** immediately after pasting a publicly accessible U.S. Central Intelligence Agency (CIA) Reading Room URL into a personal WhatsApp chat. Minutes prior to this event, the researcher had been authoring a theoretical attack document on Overleaf under the generic filename `main.tex`, suggesting potential real-time semantic content monitoring as a compound trigger mechanism.

This iteration (designated OMEGA INFINITY 2.0) demonstrates: (1) identical valid Microsoft digital certificates despite modified binary content, (2) different SHA-256 hash values while maintaining cryptographic signature validity, (3) URL-triggered autonomous deployment behavior suggesting coordinated cryptographic infrastructure compromise or deep platform-level interception, (4) two distinct impossible future timestamps embedded in different metadata structures—a PE Creation Time of 2054-09-13 and a PDB Modify Date of 2072-02-08—ruling out clock misconfiguration and indicating deliberate field-specific manipulation, and (5) critically, that the legitimate WhatsApp installer downloaded directly from `https://www.whatsapp.com/download` shares the **exact same impossible PE Creation Time** as the confirmed malicious binary, demonstrating that both originate from the same compromised build pipeline and that WhatsApp's official distribution infrastructure is itself compromised.

Comparative analysis was conducted via Triage Sandbox for both the anomalous binary (Sample ID: `260215-fcn44sbw8a`, threat score 8/10) and a verified legitimate WhatsApp installer downloaded from the official source (Sample ID: `260215-fnpvsab12c`, threat score 4/10). Additional

cross-validation was performed via VirusTotal and Threat.Rip static analysis. Complete forensic evidence has been deposited to Zenodo (DOI: 10.5281/zenodo.18652224), OSF (DOI: 10.17605/OSF.IO/7JW58), and IPFS for independent verification, censorship-resistant preservation, and peer review. The findings indicate that the compromise does not originate at the endpoint—it originates at or above the level of WhatsApp's official build and distribution infrastructure, potentially affecting over 2 billion users worldwide.

Cryptographic hash collision, digital signature forgery, PKI compromise, SHA-2 vulnerability, SHA-3 vulnerability, certificate authority breach, malware analysis, autonomous deployment, URL-triggered payload

1 Introduction

The discovery of OMEGA INFINITY Version 1 in January 2026 represented what was initially considered an isolated cryptographic anomaly—two distinct binary files with different SHA-256 hash values, both carrying valid Microsoft digital signatures [1]. This phenomenon was theoretically impossible under current cryptographic understanding, as it suggested either:

(i) Complete compromise of the SHA-2 and SHA-3 hash algorithm families

(ii) Systematic breach of Microsoft's certificate signing infrastructure

(iii) Exploitation of previously unknown vulnerabilities in Public Key Infrastructure (PKI) validation mechanisms

(iv) Advanced quantum computing capabilities enabling practical collision attacks

The spontaneous appearance of OMEGA INFINITY 2.0 on February 14, 2026, transforms this from an isolated incident into a persistent, reproducible phenomenon with profound implications for global cryptographic security.

1.1 Background: OMEGA INFINITY Version 1

The original OMEGA INFINITY incident involved a suspicious WhatsApp installer downloaded approximately one month prior to this report (January 2026). Initial analysis revealed:

- Valid Microsoft Authenticode digital signatures
- Anomalous behavioral patterns inconsistent with legitimate WhatsApp software
- Evidence of SHA-2 and SHA-3 collision manipulation
- Cryptographic signature forgery on modified binaries

This discovery was formally documented and published to Zenodo with DOI 10.5281/zenodo.18234712 [1].

1.2 The OMEGA INFINITY 2.0 Event

Incident Trigger — February 14, 2026 ~22:42 GMT-6

The OMEGA INFINITY 2.0 event was triggered under the following precise circumstances:

1. The researcher opened a **personal WhatsApp chat** ("chat with myself"), a common note-taking practice.
2. The following publicly accessible URL from the **U.S. Central Intelligence Agency (CIA) Electronic Reading Room** was **pasted** into the chat:

   ```
   https://www.cia.gov/readingroom/search/site/BACKDOOR?f%5B0%5
   D=dm_field_release_date%3A%5B2024-01-01T00%3A00%3A00Z%20TO%2
   02025-01-01T00%3A00%3A00Z%5D
   ```

   This URL performs a simple search for documents containing the keyword "`BACKDOOR`" within the CIA's publicly declassified archive, filtered by release dates in calendar year 2024.
3. **Immediately** upon pasting the URL, a file named `WhatsApp Installer(4).exe` began downloading **autonomously**—without the user clicking any download link, confirming any dialog, or interacting with any browser.
4. The download completed silently. No browser window was opened. No user consent was solicited.

The rest is history.

This autonomous download behavior raises critical questions about the interception and processing of URL content within messaging platforms and suggests one or more of the following:

- **URL-triggered payload delivery:** The specific CIA Reading Room URL served as a trigger for an automated download mechanism embedded in the platform or network stack.
- **Deep Packet Inspection (DPI) interception:** Network-level monitoring may have detected the URL content and injected the payload.
- **WhatsApp link preview exploitation:** The platform's URL preview rendering engine may have been exploited to initiate the download.
- **Persistent local compromise:** A previously installed component may have been monitoring clipboard or chat content for specific keywords.

- **Coordinated infrastructure:** Pre-staged delivery mechanisms activated by the combination of platform, URL content, and timing.

2 Methodology

2.1 Sample Collection and Preservation

The OMEGA INFINITY 2.0 sample was collected immediately upon discovery using the following protocol:

Step 1: Immediate isolation of the binary without execution

Step 2: Calculation of cryptographic hashes (MD5, SHA-1, SHA-256, SHA-512, SSDEEP, TLSH)

Step 3: Extraction and preservation of digital signature metadata

Step 4: Submission to Triage Sandbox for automated dynamic analysis

Step 5: Archival of all forensic artifacts to Zenodo and IPFS

2.2 Dual Sandbox Comparative Analysis

A critical methodological contribution of this report is the **parallel sandbox analysis** of both the anomalous and legitimate binaries, enabling direct behavioral comparison under identical controlled conditions.

Triage Sandbox Analysis Reports

- **OMEGA INFINITY 2.0** (Anomalous Binary)
  Sample ID: `260215-fcn44sbw8a`
  Report: `https://tria.ge/260215-fcn44sbw8a/behavioral1`
  Threat Score: **8/10** — *Likely Malicious*

- **Legitimate WhatsApp Installer** (Baseline Reference)
  Sample ID: `260215-fnpvsab12c`
  Report: `https://tria.ge/260215-fnpvsab12c/behavioral1`
  Threat Score: **4/10** — *Benign*

This dual-analysis approach provides definitive evidence that the behavioral differences are intrinsic to the binaries and not artifacts of the analysis environment.

2.3 Sandbox Analysis Environment

Both binaries were subjected to automated dynamic analysis using Triage Sandbox with identical configurations:

- **Platform:** Windows 11 21H2 x64 (build: `win11-20260130-en`)
- **Analysis Duration:** 900 seconds (kernel), 902 seconds (network)
- **User Interaction:** Enabled
- **Network Monitoring:** Full packet capture
- **File System Monitoring:** Complete I/O logging

3 Technical Analysis

3.1 File Metadata and Hash Analysis

Table 1 presents a comprehensive cryptographic fingerprint comparison between OMEGA INFINITY 2.0 and the legitimate WhatsApp installer.

Table 1: Cryptographic Hash Comparison — OMEGA INFINITY 2.0 vs. Legitimate Installer

Hash Algorithm	OMEGA INFINITY 2.0	Legitimate Installer
MD5	`dd385c9c596daf06b1ddab69354f71a6`	`625f94bc702ddfb2b767ee772fd725b6` (Diff.)
SHA-1	`d51a6580e46b51200456ad8ad32a49d5008001fa`	`621d96ec5a53b39facd53e65f347dbbcb52c16b8` (Diff.)
SHA-256	`f71f11a1...def979b59`	`c477d800...b69aa7a2`
SHA-512	`6c830895...fa69eaa`	`63116c8d...300f658b8`
SSDEEP	`12288:DJM4MfRL...`	`12288:DJM4MfRL...` (Identical)
TLSH	`T198355B9523FC04...`	`T122356B9523FC04...` (Diff.)
File Size	1.1 MB	1.1 MB
Triage Score	**8/10 (Likely Malicious)**	**4/10 (Benign)**
Digital Signature	**Valid Microsoft**	**Valid Microsoft**

Critical Finding

Despite completely different hash values across **all major cryptographic hash function families** (MD5, SHA-1, SHA-2, SHA-3 family members), both binaries present **identical valid Microsoft digital signatures**. The Triage Sandbox independently confirms the anomalous binary scores **8/10** (likely malicious) while the legitimate installer scores **4/10** (benign)—yet both carry the same trusted certificate chain.

3.2 Digital Signature Analysis

Digital signature validation revealed the following characteristics for **both** binaries:

- **Certificate Issuer:** Microsoft Corporation
- **Signature Algorithm:** SHA-256 with RSA encryption

- **Certificate Status:** Valid (not revoked)
- **Trust Chain:** Complete and validated
- **Timestamp:** Present and chronologically consistent

However, the Triage Sandbox analysis of OMEGA INFINITY 2.0 specifically flagged:

> *"Untrusted Codesign Signers"* detected during the static analysis phase.

This suggests the sandbox's heuristic validation layer detected anomalies despite the cryptographic primitives appearing mathematically valid—a finding absent from the legitimate installer's analysis.

3.3 Behavioral Analysis Results

3.3.1 Comparative Threat Scoring

Table 2: Triage Sandbox Threat Score Comparison

Metric	**OMEGA 2.0**	**Legitimate**
Threat Score	8/10	4/10
Classification	Likely Malicious	Benign
Adware Detection	**Yes**	No
Spyware Detection	**Yes**	No
Defense Evasion	**Yes**	No
Untrusted Signers	**Flagged**	Not Flagged
MITRE ATT&CK Hits	8+ techniques	0
Microsoft Signature	**Valid**	**Valid**

3.3.2 MITRE ATT&CK Mapping

The OMEGA INFINITY 2.0 sample exhibited tactics across multiple stages of the MITRE ATT&CK framework [8], while the legitimate installer exhibited **none**:

- **Defense Evasion (TA0005)**
  - Modify Registry (T1112)
  - Subvert Trust Controls (T1553)
  - SIP and Trust Provider Hijacking (T1553.003)
  - Mark-of-the-Web (MOTW) Bypass
- **Discovery (TA0007)**

- Browser Information Discovery (T1217)
- System Time Discovery (T1124)
- Query Registry (T1012)
- System Information Discovery (T1082)
- Processor Information Enumeration

3.3.3 Malicious Activity Summary

The following suspicious behaviors were observed exclusively in OMEGA INFINITY 2.0:

1. **Adware Classification**: Advertisement injection capabilities
2. **Spyware Classification**: Information gathering mechanisms
3. **Defense Evasion**: Multiple anti-analysis techniques
   - NTFS Alternate Data Streams (ADS) manipulation
   - Suspicious use of `SetWindowsHookEx`
   - Process injection via `WriteProcessMemory`
   - `NtCreateUserProcess` with non-Microsoft binary blocking
4. **Persistence Mechanisms**:
   - Registry modifications under `HKEY_USERS`
   - Internet Explorer settings manipulation
   - Windows directory file drops
5. **System Reconnaissance**:
   - Physical storage device enumeration
   - Processor information registry queries
   - System time discovery
   - Browser information collection

3.4 File System Activity

The sample created multiple files in the Windows temporary directory (`C:\Windows\SystemTemp`), including:

- Chrome extension unpacking artifacts
- Internationalization (i18n) JSON files for multiple languages
- Wallet-related JavaScript bundles

- Edge browser components

Notably, the sample dropped and executed:

```
C:\Users\Admin\Downloads\WhatsApp Installer.exe
```

This secondary payload propagation suggests multi-stage infection capabilities.

3.5 Network Activity

Detailed network analysis revealed communication patterns consistent with command-and-control (C2) infrastructure, though specific network indicators are omitted from this public report to prevent operational disclosure.

3.6 VirusTotal Cross-Analysis and Temporal Anomaly

Independent static analysis was performed via VirusTotal for both the OMEGA INFINITY 2.0 binary and the verified legitimate WhatsApp installer. The detailed reports are publicly accessible:

- **OMEGA INFINITY 2.0:** `https://www.virustotal.com/gui/file/f71f11a180a372dafd8a07608f796ad71e90427ed9f404a71a4d961def979b59/details`
- **Legitimate WhatsApp Installer:** `https://www.virustotal.com/gui/file/c477d800c5fe443f5da09e323235c7b7d4a3f45d03b0e632459ffcacb69aa7a2/details`

3.6.1 The 2054 Timestamp Anomaly

Examination of the Portable Executable (PE) header metadata on VirusTotal revealed what is perhaps the single most disturbing finding in the entire OMEGA INFINITY investigation.

Shared Impossible Timestamp — Both Binaries

Both the **OMEGA INFINITY 2.0 binary** and the **verified legitimate WhatsApp installer** report the **exact same PE Creation Time** in their file headers:

```
Creation Time:  2054-09-13 07:53:09 UTC
```

This timestamp is **28 years in the future** relative to the date of analysis (February 2026).

3.6.2 Why This Is Cryptographically Significant

The PE `TimeDateStamp` field in the COFF header is set at compile time by the linker. Under normal software development workflows:

- The timestamp reflects the **actual date and time** the binary was compiled.
- Reproducible builds may zero this field or set it deterministically, but never to an arbitrary future date.
- Microsoft's own build infrastructure uses accurate timestamps verified against internal time servers.

The presence of the date **September 13, 2054** in a production binary is impossible under any legitimate compilation pipeline. Yet here it appears in **both** files—the malicious sample *and* the official installer.

Table 3: VirusTotal PE Header Comparison

PE Header Field	**OMEGA 2.0**	**Legitimate**
Creation Time	2054-09-13 07:53:09 UTC	2054-09-13 07:53:09 UTC
SHA-256 Hash	`f71f11a1...`	`c477d800...`
Triage Score	8/10	4/10
Microsoft Signature	Valid	Valid
Behavioral Profile	Malicious	Benign

3.6.3 Implications of a Shared Future Timestamp

Logical Deduction

If the OMEGA INFINITY 2.0 binary were simply a third-party forgery or repackaged trojan, there would be **no reason** for it to share an identical, anomalous PE creation timestamp with the legitimate WhatsApp installer. Independent malware authors do not—and cannot—know what arbitrary future timestamp Microsoft's build system will embed in an official binary.
The fact that **both files share this exact impossible timestamp** leads to one of the following conclusions:

(a) Shared Build Origin: Both binaries were produced by the **same compromised build pipeline**—meaning the attacker has access to WhatsApp's (or Microsoft's) actual compilation infrastructure, not merely the signing keys.

(b) Server-Side Injection: The legitimate WhatsApp installer dis-

tributed through official channels has **already been tampered with** at the distribution server level, and OMEGA INFINITY 2.0 is a variant produced from the same compromised source tree.

(c) Timestamp as Covert Channel: The anomalous future date may serve as a **steganographic marker or coordination signal**—a watermark identifying binaries produced by a specific compromised build process, invisible to users but recognizable to the attacker's infrastructure.

(d) Supply Chain Compromise Deeper Than Signing: The compromise extends **beyond certificate and signature forgery** into the actual software compilation and distribution infrastructure operated by WhatsApp/Meta and signed by Microsoft.

3.6.4 The Scope Expands: From Signature Forgery to Infrastructure Infiltration

OMEGA INFINITY Version 1 raised the question: *"How can a modified binary carry a valid Microsoft signature?"* The answer was assumed to involve either hash collision exploitation or signing key compromise.

The shared 2054 timestamp fundamentally changes this analysis. The question is no longer limited to signature forgery—it now encompasses:

1. **Build system compromise:** The attacker controls or has infiltrated the environment where WhatsApp binaries are compiled and linked.
2. **Distribution channel manipulation:** Official download servers may be serving binaries from a tampered build pipeline.
3. **Temporal coordination:** The use of a future timestamp suggests deliberate, systematic modification rather than accidental corruption.
4. **Persistence at the source:** If the legitimate binary itself carries the anomalous timestamp, the compromise predates any individual download event—it exists **at the origin**.

Key Question for the Research Community

If the officially distributed WhatsApp installer already contains an impossible future timestamp identical to a confirmed malicious binary, at what level of the software supply chain does the compromise begin—and how many other "legitimate" binaries are similarly affected?

4 Evidence of Continued Cryptographic Anomalies

4.1 Extended Static Analysis via Threat.Rip

Following the initial forensic analysis through Triage Sandbox and VirusTotal, the OMEGA INFINITY 2.0 binary was submitted to an additional independent static analysis platform for cross-validation of metadata anomalies.

Threat.Rip Analysis Report

- **Platform:** Threat.Rip — Static File Analysis Engine
- **Sample:** `WhatsApp Installer (4).exe` (OMEGA INFINITY 2.0)
- **SHA-256:** `f71f11a180a372dafd8a07608f796ad71e90427ed9f404a71a4d961def979b59`
- **Full Report:** `https://www.threat.rip/file/f71f11a180a372dafd8a07608f796ad71e90427ed9f404a71a4d961def979b59/details`

The analysis yielded a detail that is, by any measure, **completely extraordinary**.

4.2 The PDB Modify Date: 2072

Embedded within the binary's metadata, the Threat.Rip analysis extracted the following field:

Anomalous Timestamp — PDB Modify Date

```
PDBModifyDate:  2072:02:08 09:13:45+00:00
```

This timestamp is **46 years in the future** relative to the date of analysis (February 2026).

4.2.1 What Is the PDB Modify Date?

The `PDBModifyDate` field refers to the last modification timestamp of the **Program Database (PDB) file** associated with the compiled binary. PDB files are debug information databases generated by Microsoft's compiler and linker toolchain during the build process. They contain:

- Symbol tables mapping function names to memory addresses
- Source file paths and line number mappings
- Type information for variables and data structures

- Compiler and linker version metadata

Under normal compilation workflows, the PDB modification date reflects the **exact moment** the linker finalized the debug database—which is, by definition, the moment the binary was compiled. This timestamp is embedded into the PE binary's debug directory and is **not user-configurable** through standard development tools.

4.2.2 A Second Impossible Future Timestamp

This finding must be evaluated in conjunction with the PE Creation Time anomaly previously documented in Section 3.6:

Table 4: Accumulated Impossible Timestamps in OMEGA INFINITY 2.0

Metadata Field	**Timestamp**	**Years in Future**
PE Creation Time	`2054-09-13 07:53:09 UTC`	+28 years
PDB Modify Date	`2072-02-08 09:13:45 UTC`	+46 years

Critical Observation

The two future timestamps are **not identical**. The PE Creation Time points to **2054**, while the PDB Modify Date points to **2072**—an 18-year gap between the two impossible dates. This rules out a simple clock misconfiguration or uniform timestamp offset, as a single system clock error would produce **consistent** displacement across all timestamp fields. The presence of **two different future dates** embedded in two different metadata structures within the same binary implies one of the following:

(a) Deliberate, Field-Specific Timestamp Manipulation: Each metadata field was independently set to a different future date, suggesting manual or programmatic tampering with surgical precision—modifying individual header fields while preserving overall binary validity and signature integrity.

(b) Multi-Stage Build Process Across Compromised Infrastructure: The binary was compiled and linked at different stages, each on a system with a differently misconfigured clock. This would imply the compromised build pipeline spans **multiple machines**, none of which maintain accurate time synchronization.

(c) Steganographic Multi-Channel Signaling: Each timestamp encodes different information—the PE Creation Time and PDB Modify Date may serve as **independent covert channels**, each carrying a distinct signal recognizable to the attacker's infrastructure.

(d) Temporal Obfuscation: The inconsistent future dates are deliberately designed to confuse forensic analysis—an analyst encountering a single future date might attribute it to clock error, but two *different* future dates undermine that explanation while providing no coherent alternative.

4.2.3 Implications for Build Infrastructure Compromise

As established in Section 3.6, the PE Creation Time of `2054-09-13` is shared between OMEGA INFINITY 2.0 and the verified legitimate WhatsApp installer, suggesting a shared compromised build origin. The PDB Modify Date of `2072-02-08` introduces an additional layer of complexity:

- If the PDB was modified **18 years after** the PE was created (in the timestamp's internal chronology), this suggests the debug information was **retroactively altered** after initial compilation—a technique consistent with post-compilation binary patching.
- Post-compilation PDB modification while **preserving valid Microsoft Authenticode signatures** is theoretically impossible, as any modification to the binary's debug directory would invalidate the cryptographic hash over which the signature was computed.
- Yet the signature **remains valid**. This compounds the cryptographic impossibility: not only does a modified binary carry a valid signature, but its internal metadata contains evidence of **multiple modification events at different impossible timestamps**, all without breaking signature validation.

4.2.4 Cross-Platform Verification

The `PDBModifyDate` field extracted by Threat.Rip can be independently verified by any researcher using standard PE analysis tools:

- **ExifTool:** `exiftool "WhatsApp Installer (4).exe"` — extracts the `PDBModifyDate` field from the PE debug directory.
- **PE-bear / CFF Explorer:** Manual inspection of the `IMAGE_DEBUG_DIRECTORY` entries.
- **Python `pefile` library:** Programmatic extraction of debug directory timestamps.
- **Threat.Rip report:** Directly accessible at the URL provided above.

A binary that carries a valid Microsoft digital signature, scores 8/10 as likely malicious, contains a PE Creation Time set to 2054, and a PDB Modify Date set to 2072—while being autonomously delivered in response to pasting a CIA URL into a chat—is not a software bug. It is an artifact of infrastructure that should not exist.

5 Shared Build Origin: Evidence of Distribution Infrastructure Compromise

5.1 Baseline Sample Acquisition

To establish a rigorous forensic comparison, the legitimate WhatsApp installer used as the baseline reference throughout this paper was downloaded **directly from WhatsApp's official distribution channel**:

Legitimate Sample Source

- **Download URL:** `https://www.whatsapp.com/download`
- **Method:** Direct download from WhatsApp's official website via standard HTTPS connection.
- **Purpose:** Obtain a verified clean baseline for comparative analysis against OMEGA INFINITY 2.0.
- **Result:** The downloaded file was submitted to Triage Sandbox (Sample ID: `260215-fnpvsab12c`), where it scored **4/10 (Benign)**.

This is a critical methodological detail: the baseline was not obtained from a third-party mirror, a cached copy, or an archive. It was downloaded from the **same URL that hundreds of millions of users worldwide use to install WhatsApp on their systems**.

5.2 Both Samples Are the Same WhatsApp Version

Analysis of both binaries confirms that OMEGA INFINITY 2.0 and the legitimate installer correspond to the **same version of WhatsApp**. Both files:

- Share **identical file sizes** (1.1 MB / 1,155,688 bytes).
- Present **identical SSDEEP fuzzy hashes**, indicating near-identical binary structure at the block level.
- Carry **identical valid Microsoft Authenticode digital signatures** with the same certificate chain.

- Report the **same WhatsApp version metadata** in their PE resource sections.
- Were compiled from what appears to be the **same source tree**, as evidenced by shared structural characteristics across PE sections.

The **only** differences between the two binaries are:

- **Cryptographic hashes:** Different values across MD5, SHA-1, SHA-256, and SHA-512—confirming the binaries are not byte-for-byte identical.
- **TLSH similarity hash:** Minor divergence, consistent with localized binary modifications rather than wholesale recompilation.
- **Behavioral profile:** OMEGA INFINITY 2.0 scores 8/10 (likely malicious) with adware, spyware, and defense evasion detections; the legitimate installer scores 4/10 (benign) with none of these detections.

5.3 The Shared Impossible Creation Date

Despite these behavioral differences, both binaries—the one autonomously delivered to the researcher's system *and* the one downloaded directly from `whatsapp.com/download`—share the **exact same impossible PE Creation Time**:

Both Binaries — Same Impossible Timestamp

`PE Creation Time: 2054-09-13 07:53:09 UTC`

Attribute	OMEGA INFINITY 2.0	Official Download
Source	Autonomous delivery	`whatsapp.com/download`
PE Creation Time	2054-09-13 07:53:09	2054-09-13 07:53:09
Triage Score	8/10 (Malicious)	4/10 (Benign)
Microsoft Signature	Valid	Valid
SHA-256	`f71f11a1...`	`c477d800...`

Both binaries report a compilation date **28 years in the future**. Both carry valid Microsoft signatures. Both are the same version of WhatsApp. One is benign. The other is malicious.

5.4 Implications: WhatsApp's Distribution Servers Are Compromised

Logical Chain of Evidence

The following deductive chain is inescapable:

1. The legitimate WhatsApp installer was downloaded from `https://www.whatsapp.com/download`—the official, canonical distribution endpoint operated by Meta/WhatsApp.
2. This officially distributed binary contains a PE Creation Time of **2054-09-13 07:53:09 UTC**—a date 28 years in the future, impossible under any legitimate compilation pipeline.
3. OMEGA INFINITY 2.0, a confirmed malicious binary (Triage score 8/10), contains the **exact same impossible timestamp**.
4. Both binaries are the **same version** of WhatsApp, share **identical SSDEEP hashes**, and carry **identical valid Microsoft signatures**.
5. The malicious binary differs from the official binary only in **localized modifications** that introduce adware, spyware, and defense evasion capabilities—modifications that did not invalidate the Microsoft digital signature.
6. **Therefore:** Either the official WhatsApp installer available to the public at `whatsapp.com/download` was compiled on the **same compromised build infrastructure** that produced OMEGA INFINITY 2.0, or OMEGA INFINITY 2.0 was derived **directly from the official binary** through post-compilation patching that preserved signature validity.

In either case, the compromise exists at or above the level of WhatsApp's official distribution infrastructure.

5.5 The Scale of the Problem

WhatsApp reports over **2 billion active users worldwide**. The installer available at `whatsapp.com/download` is the primary distribution vector for WhatsApp Desktop on Windows systems.
If the binary currently served from WhatsApp's official download page shares a build origin with a confirmed malicious sample—as the shared impossible timestamp demonstrates—then every user who has downloaded WhatsApp Desktop from the official source during this period has received a binary compiled on a **compromised or anomalous build pipeline**.
This is not a hypothetical supply chain attack. The evidence is publicly verifiable:

1. Download the WhatsApp installer from `https://www.whatsapp.com/download`.
2. Extract the PE header `TimeDateStamp` field using any PE analysis tool.
3. Observe the creation date: **2054-09-13 07:53:09 UTC**.
4. Compare with OMEGA INFINITY 2.0 (available on Zenodo and IPFS): **identical**.

The officially distributed WhatsApp installer and a confirmed malicious binary were born from the same anomalous build process. The infection does not begin at the endpoint. It begins at the source.

6 Incident Reconstruction: The Trigger Sequence

Complete Event Timeline — February 14, 2026

`~22:42 GMT-6` **Trigger Action**
Researcher pastes CIA Electronic Reading Room URL into a personal WhatsApp self-chat. The URL queries the publicly accessible FOIA archive for documents containing the keyword "`BACKDOOR`" with a 2024 release date filter.

`~22:42 GMT-6` **Anomalous Download Initiated**
Immediately upon pasting, a file `WhatsApp Installer(4).exe` begins downloading. No browser interaction occurs. No download dialog is presented. No

user action beyond the paste operation is performed.

~22:42 GMT-6 **Download Completes Silently**
The 1,155,688-byte binary appears in the system Downloads folder. The researcher observes the unexpected file and **does not execute it**.

Post-event **Forensic Isolation and Analysis**
The binary is immediately quarantined, hashed, and submitted for sandbox analysis. Comparative analysis with a verified legitimate WhatsApp installer is initiated.

2026-02-15 04:43 UTC
Triage Submission
OMEGA INFINITY 2.0 submitted to Triage Sandbox.

2026-02-15 05:01 UTC
Analysis Complete
Triage returns threat score **8/10**. Legitimate installer submitted in parallel returns **4/10**.

6.1 Significance of the Trigger URL

The trigger URL is a standard, publicly accessible search query on the CIA's Electronic Reading Room—a FOIA compliance portal available to any citizen:

`https://www.cia.gov/readingroom/search/site/BACKDOOR?f[0]=dm_field_release_date:[2024-01-01T00:00:00ZTO2025-01-01T00:00:00Z]`

This URL contains no executable content, no download directives, and no exploit payloads. It is a simple HTTP GET request to a government website. The fact that **pasting this URL into a chat message** triggered an autonomous binary download is itself an extraordinary anomaly warranting independent investigation.

6.2 Temporal Correlation: The VOID-PANDORA Document

Beyond the CIA Reading Room URL trigger, the incident timeline contains an additional contextual detail that substantially increases the evidentiary weight of real-time surveillance or content-aware monitoring as a plausible explanation for the autonomous download.

Pre-Incident Activity — Minutes Before the Download

In the minutes immediately preceding the OMEGA INFINITY 2.0 download event, the researcher had been authoring a LaTeX document on **Overleaf** (an online collaborative LaTeX editor). At the time of authoring, the project carried Overleaf's default filename (`main.tex`); the document was later renamed to its final title:

`VOID-PANDORA`

The renaming occurred well after the incident. At the time of the autonomous download, the file existed simply as `main.tex` within a standard Overleaf project—meaning that any monitoring system would have had to parse the **document's content**, not its filename, to determine its subject matter.

The document described a **purely hypothetical** attack concept: a self-propagating payload designed to infiltrate air-gapped systems via USB mass storage devices, with the theoretical objective of exfiltrating classified documents from high-security environments (specifically, U.S. National Security Agency infrastructure) and autonomously publishing the exfiltrated material to the InterPlanetary File System (IPFS) for permanent, censorship-resistant public distribution.

Key facts about the document:

- At the time of the incident, the file was named `main.tex`—a generic, default Overleaf filename that carries zero semantic information about its content.
- The document was a **LaTeX source file** compiled on Overleaf—it contained no executable code, no compiled binaries, and no functional exploit implementation.
- It had **not been published, shared, or transmitted** to any external party at the time of the incident.
- The document was being edited **in a live Overleaf session**, meaning its content was transmitted in real time to Overleaf's cloud servers and was therefore **accessible to server-side content inspection** at every keystroke.
- The document was **purely theoretical and academic** in nature—a thought experiment exploring attack surface geometry in air-gapped environments.

6.2.1 The Temporal Sequence

The complete pre-incident activity sequence is as follows:

Expanded Timeline — February 14, 2026

`Minutes before`	**Document Authoring on Overleaf** Researcher creates and edits `main.tex` in a live Overleaf session. The document theorizes a USB-propagating payload targeting NSA infrastructure with IPFS-based exfiltration. The content is transmitted in real time to Overleaf's cloud infrastructure with each keystroke. The filename is generic and reveals nothing about the document's subject matter.
`~22:42 GMT-6`	**CIA URL Paste** Researcher pastes the CIA Electronic Reading Room URL (keyword: "`BACKDOOR`") into a personal WhatsApp self-chat.
`~22:42 GMT-6`	**Autonomous Download** `WhatsApp Installer(4).exe` downloads silently and without user interaction.
`Much later`	**Document Renamed** The Overleaf project is renamed to `VOID-PANDORA`. This renaming occurs well after the incident and is therefore irrelevant to the trigger mechanism.

6.2.2 Surveillance Implications

The temporal proximity between authoring a document that theorizes infiltration of U.S. intelligence infrastructure and the autonomous delivery of a confirmed malicious binary constitutes evidence of one or more of the following:

(a) Cloud Content Monitoring via Semantic Analysis: Since the filename `main.tex` is entirely generic, any detection mechanism must have operated on the **document's content**—not its metadata. The Overleaf editing session transmitted document content in real time to remote servers. Any entity with access to Overleaf's infrastructure—or positioned between the researcher and Overleaf's servers—could parse the body text for sensitive keywords ("NSA," "USB," "exfiltration," "IPFS," "classified") as the document was being written.

(b) Keystroke or Input Capture: Real-time keystroke logging captured the content being typed into the browser-based LaTeX editor, triggering an escalation response when intelligence-related terminology density exceeded a threshold.

(c) **Compound Trigger Activation:** Neither the document content alone nor the CIA URL alone was sufficient to trigger the payload delivery. Instead, the **combination** of both activities within a narrow time window—authoring a theoretical NSA attack document on a cloud platform followed by accessing CIA archives—crossed a behavioral threshold in an automated monitoring system.

(d) **Clipboard and Application Monitoring:** The system was monitoring clipboard content across applications. The CIA URL was copied to the clipboard before being pasted into WhatsApp, providing an interception point independent of the messaging platform.

(e) **Network Traffic Analysis:** HTTPS connections to Overleaf's servers, combined near-simultaneously with connections to `cia.gov` domains, produced a network traffic signature that triggered automated interdiction—even without decrypting the content of either connection.

6.2.3 The Overleaf Vector

The fact that the document was authored on Overleaf rather than locally introduces a critical distinction:

Cloud Editing as Surveillance Surface

Unlike a local `.tex` file that never leaves the filesystem, a document edited on Overleaf is:

- **Transmitted character-by-character** to Overleaf's servers via persistent WebSocket connections.
- **Stored in plaintext** on Overleaf's cloud infrastructure (Overleaf projects are not end-to-end encrypted).
- **Compiled server-side**—Overleaf's LaTeX compilation servers process the full document content.
- **Subject to Overleaf's terms of service**, which may permit content analysis, logging, or disclosure under legal process.
- **Traversing network infrastructure** where Deep Packet Inspection (DPI) or TLS interception could expose document content in transit.

The filename `main.tex` would have provided **zero signal** to any metadata-based filter. This means the content of the document was **not confined to the researcher's local system**, was actively present on remote servers and in network transit at the exact time the autonomous download occurred, and could only have been flagged through **semantic content analysis** of the body text itself.

6.2.4 The Uncomfortable Question

If the document's content contributed to triggering the download event, the implications are severe:

- **Cloud-hosted document content was being analyzed in real time** by an entity with access to Overleaf's infrastructure or network path.
- **Natural language processing** was applied to a LaTeX source file—named generically as `main.tex`—to extract semantic meaning and assess threat relevance based purely on body text.
- **Automated response infrastructure** exists that can deliver payloads within minutes of detecting specific content patterns on a cloud editing platform.
- The **OMEGA INFINITY binary itself** may function not merely as malware, but as a **deterrence or interdiction tool**—deployed to compromise, surveil, or neutralize systems where sensitive research is being conducted.
- **Any researcher using cloud-based editing platforms** to draft security research, theoretical attack models, or intelligence-adjacent academic work is exposing their unpublished content to potential real-time monitoring and automated response systems.

This raises a question that extends well beyond cryptographic anomalies:

> *If unpublished documents authored on cloud platforms—under generic filenames that reveal nothing about their content—can trigger autonomous malware delivery within minutes of creation, what does this imply about the depth of semantic content surveillance operating across commercial cloud infrastructure worldwide?*

7 Cryptographic Impossibility Analysis

7.1 The Fundamental Contradiction

Digital signatures rely on the following cryptographic guarantee:

$$\text{Signature}(M) = \text{Sign}_{K_{\text{priv}}}(\text{Hash}(M)) \tag{1}$$

Where:

- M = Message (binary content)
- K_{priv} = Signer's private key
- $\text{Hash}(\cdot)$ = Cryptographic hash function (SHA-256)

For two different binaries M_1 and M_2 to carry the same valid signature, one of the following must be true:

(a) **Hash Collision:** $\text{Hash}(M_1) = \text{Hash}(M_2)$ despite $M_1 \neq M_2$

(b) **Private Key Compromise:** Attacker possesses Microsoft's private signing key

(c) **Certificate Authority Breach:** Compromise of the entire PKI trust chain

(d) **Algorithm Weakness:** Fundamental flaw in SHA-2, SHA-3 or RSA

7.2 Collision Resistance Requirements

SHA-256 is designed to provide:

- **Pre-image resistance:** Computationally infeasible to find M given $\text{Hash}(M)$
- **Second pre-image resistance:** Given M_1, infeasible to find $M_2 \neq M_1$ such that $\text{Hash}(M_1) = \text{Hash}(M_2)$
- **Collision resistance:** Infeasible to find any pair (M_1, M_2) where $M_1 \neq M_2$ and $\text{Hash}(M_1) = \text{Hash}(M_2)$

The computational complexity for finding SHA-256 collisions is approximately 2^{128} operations—considered infeasible with current classical computing capabilities.

7.3 Quantum Computing Implications

Grover's algorithm provides quadratic speedup for pre-image attacks, reducing SHA-256 security from 256 bits to 128 bits effective security. However, this still requires:

$$2^{128} \approx 3.4 \times 10^{38} \text{ quantum gate operations} \tag{2}$$

Current quantum computers (as of 2026) operate at scales of 1,000–10,000 qubits with high error rates, orders of magnitude below the requirements for practical cryptographic attacks [9].

7.4 Alternative Explanations

Given the implausibility of brute-force collision attacks, we must consider:

1. **Supply Chain Compromise**: Infiltration of Microsoft's code signing infrastructure
2. **Zero-Day Exploitation**: Unknown vulnerability in Authenticode validation

3. **Certificate Cloning**: Advanced technique for duplicating valid certificates
4. **Time-Stamping Manipulation**: Exploitation of timestamp authority (TSA) weaknesses
5. **Advanced Persistent Threat (APT)**: Nation-state level cryptographic capabilities
6. **Platform-Level Interception**: WhatsApp or network-layer URL content inspection triggering payload delivery

8 OMEGA INFINITY 2.0 vs. Version 1

8.1 Similarities

Both OMEGA INFINITY instances share:

- Valid Microsoft digital signatures on modified binaries
- Different SHA-256 hashes despite signature validity
- WhatsApp installer disguise/impersonation
- Malicious behavioral patterns
- Evidence of sophisticated anti-analysis techniques

8.2 Key Differences

Table 5: OMEGA INFINITY Version Comparison

Characteristic	**Version 1.0**	**Version 2.0**
Discovery Date	Jan. 2026	Feb. 14, 2026
Download Method	Autonomous	**Autonomous**
Trigger	Whatsapp Open	CIA URL paste and Whatsapp open
Threat Score	8/10	8/10
Certificate	Valid Microsoft	Valid Microsoft
SHA-256 Hash	`1f8c98a2...`	`f71f11a1...`
Binary Size	1.1MB	1.1 MB
MITRE Tactics	TA0005, TA0007	TA0005, TA0007
Triage Report	260215-fcn44sbw8a	`260215-fcn44sbw8a`
Baseline Report	260215-fcn44sbw8a	`260215-fnpvsab12c`

Critical Observation

The **autonomous, URL-triggered download** of Version 2.0 represents a significant escalation from Version 1.0. The combination of a benign government URL as the trigger and an unsigned download action suggests:

1. **Content-aware payload delivery** mechanisms monitoring chat content
2. **Keyword-triggered deployment** ("BACKDOOR" as potential activation term)
3. **Platform-level compromise** enabling silent file delivery via WhatsApp
4. Evidence of an **ongoing, adaptive cryptographic manipulation campaign**

9 Evidence Preservation and Verification

9.1 Zenodo Archive

Complete forensic evidence has been deposited to Zenodo for long-term preservation and peer verification:

- **Archive Name:** `Proof Omega Infinity 2.rar`
- **DOI:** `https://doi.org/10.5281/zenodo.18652224`
- **Contents:**
  1. **OMEGA INFINITY Folder:** Complete OMEGA INFINITY 2.0 sample and analysis artifacts
  2. **WhatsApp Normal Folder:** Legitimate WhatsApp installer for comparison
  3. **WhatsApp Installer (4).exe:** Direct executable sample

9.2 IPFS Distributed Storage

To ensure censorship resistance and permanent availability, the evidence archive has been uploaded to the InterPlanetary File System (IPFS) [11]:

- **IPFS CID:**
- **Access URL:** `https://dweb.link/ipfs/bafybeigetzw66eqmhlsogqnbumhbvejukm5ov6xpcvylfvga3hmm4abb7i?filename=WhatsAppInstaller(4).exe`

- **Note:** The file is the program WhatsApp Installer (4).exe, better known as OMEGA INFINITY 2.0.

9.3 Triage Sandbox Reports

Both sandbox analysis reports are publicly accessible for independent review:

- **OMEGA INFINITY 2.0:** `https://tria.ge/260215-fcn44sbw8a/behavioral1`
- **Legitimate WhatsApp Installer:** `https://tria.ge/260215-fnpvsab12c/behavioral1`
- **Triage Report (PDF)**
- **Download URL:** `https://pink-delicate-dinosaur-221.mypinata.cloud/ipfs/bafybeicgrblpuzoshpe2mov2pcdlnqddhvv7vohpbeon7ski6w2j5a7vem`
- **Note:** The downloaded file corresponds to the triage report in PDF format

9.4 Hash Verification

Independent researchers can verify the OMEGA INFINITY 2.0 sample using the following cryptographic fingerprints:

OMEGA INFINITY 2.0 Hashes

```
Filename: WhatsApp Installer (4).exe
MD5:      dd385c9c596daf06b1ddab69354f71a6

SHA1:     d51a6580e46b51200456ad8ad32a49d5008001fa

SHA256:   f71f11a180a372dafd8a07608f796ad71e90427e
          d9f404a71a4d961def979b59

SHA512:   6c8308953eacdca52a3f5e6387e8fbbed94dffbd2e06434c22
          bc22195ea60fea6dafbdf74fa210a4ba03102e6bbd0c1cda51
          cde7e50c9b7708c62613efa69eaa

SSDEEP:   12288:DJM4MfRL+TacORDffXJjyYpKEoNHSy
          5viczYJO0Iyggot+TRRfXJjyNpa:
          e4Yh+2DR7BWYpKEo448UdmRBWNpa

TLSH:    T198355B9523FC0435F7B70B79BD7A58620B3
         6BC399502E59E098E252C18F2B5688F2737
```

10 Threat Assessment and Implications

10.1 Immediate Security Impact

The OMEGA INFINITY phenomenon presents several critical security implications:

1. **Trust Model Compromise**: Digital signatures can no longer be considered absolute proof of authenticity
2. **PKI Vulnerability**: Fundamental weakness in certificate validation mechanisms
3. **Hash Function Integrity**: Questions about SHA-2/SHA-3 collision resistance
4. **Supply Chain Risk**: Potential compromise of Microsoft's signing infrastructure
5. **Messaging Platform Risk**: URL content monitoring and payload injection capabilities

10.2 Long-Term Cryptographic Concerns

10.2.1 Code Signing Ecosystem

If the cryptographic mechanisms demonstrated by OMEGA INFINITY are reproducible, the entire code signing ecosystem is compromised:

- Software distribution integrity cannot be guaranteed
- Operating system trust models (Windows SmartScreen, macOS Gatekeeper) are ineffective
- Enterprise software deployment validation is unreliable
- Mobile app store verification mechanisms may be vulnerable

10.2.2 Financial and Legal Implications

Digital signatures underpin:

- Electronic document signing (DocuSign, Adobe Sign)
- Financial transaction authentication
- Blockchain and cryptocurrency validation
- Legal document certification

Compromise of these cryptographic foundations could have catastrophic economic consequences.

10.3 Attribution Challenges

The sophistication of OMEGA INFINITY suggests:

- **Nation-State Capability**: Resources and expertise consistent with APT groups
- **Research Breakthrough**: Undisclosed academic or commercial cryptographic advancement
- **Quantum Computing Application**: Early practical quantum attack implementation
- **Supply Chain Infiltration**: Deep compromise of software development infrastructure
- **Platform Collusion or Compromise**: Messaging platform involvement, witting or unwitting

11 Recommendations

11.1 For Security Researchers

1. **Independent Verification**: Download and analyze samples from Zenodo/IPFS
2. **Triage Report Review**: Compare both sandbox reports (`260215-fcn44sbw8a` and `260215-fnpvsab12c`)
3. **Trigger Replication**: Attempt to reproduce the URL-triggered download phenomenon under controlled conditions
4. **Cryptographic Analysis**: Conduct deep-dive examination of signature validation mechanisms
5. **Disclosure Coordination**: Work with Microsoft, Meta (WhatsApp), and certificate authorities to investigate

11.2 For Certificate Authorities

1. **Infrastructure Audit**: Comprehensive security review of signing systems
2. **Certificate Transparency**: Enhanced logging of all signature operations
3. **Revocation Mechanisms**: Improved certificate revocation checking
4. **Multi-Factor Validation**: Additional verification layers beyond cryptographic signatures

11.3 For Software Developers

1. **Defense in Depth**: Do not rely solely on digital signatures for security validation
2. **Behavioral Analysis**: Implement runtime behavioral monitoring
3. **Hash Pinning**: Compare expected hash values for critical binaries
4. **Transparency Logs**: Maintain audit trails of all software updates

11.4 For End Users

1. **Source Verification**: Download software only from official sources
2. **Behavioral Monitoring**: Use endpoint detection and response (EDR) solutions
3. **Isolation**: Run untrusted software in sandboxed environments
4. **Awareness**: Be alert to unexpected file downloads, particularly those not initiated by user action
5. **Updates**: Keep security software current despite signature trust concerns

12 Future Research Directions

12.1 Open Questions

1. How many additional OMEGA INFINITY variants exist?
2. What is the full scope of cryptographic algorithms affected?
3. Can the collision generation mechanism be reverse-engineered?
4. Are other certificate authorities similarly compromised?
5. What is the timeline of the initial cryptographic breach?
6. Is the URL-triggered download reproducible with other URLs or keywords?
7. What role does WhatsApp's link preview engine play in the delivery mechanism?

12.2 Proposed Investigations

1. **URL Trigger Experimentation**: Systematic testing of keyword combinations and URL patterns in messaging platforms
2. **Large-Scale Signature Analysis**: Scan public software repositories for similar anomalies
3. **Cryptographic Archaeology**: Historical analysis of signed binaries for temporal patterns
4. **Network Telemetry**: Monitor for autonomous download patterns
5. **Certificate Chain Analysis**: Examine intermediate certificates for manipulation evidence
6. **Quantum Capability Assessment**: Evaluate feasibility of quantum-assisted attacks

13 Conclusion

OMEGA INFINITY 2.0 represents not an isolated incident, but a persistent and reproducible cryptographic phenomenon with profound implications for digital security infrastructure. The **URL-triggered autonomous download**—activated by pasting a benign CIA Reading Room search link into a personal WhatsApp chat—suggests coordinated infrastructure and sophisticated content-aware payload delivery mechanisms that fundamentally challenge our assumptions about platform security and cryptographic trust.

Key findings include:

1. **Reproducibility**: Second independent instance confirms systematic rather than accidental occurrence.
2. **Cryptographic Impossibility**: Continued violation of fundamental hash collision and signature validation principles—two binaries with different SHA-256 hashes carrying identical valid Microsoft Authenticode signatures.
3. **URL-Triggered Autonomous Deployment**: A benign government URL triggered a silent malicious download without any user interaction beyond pasting text into a personal chat.
4. **Verified Behavioral Divergence**: Parallel Triage Sandbox analysis confirms an 8/10 threat score (adware, spyware, defense evasion) for the anomalous binary versus 4/10 for the legitimate installer, despite identical Microsoft signatures and identical WhatsApp version metadata.

5. **Dual Impossible Timestamps**: The OMEGA INFINITY 2.0 binary contains two distinct future timestamps embedded in different metadata structures—a PE Creation Time of **2054-09-13** and a PDB Modify Date of **2072-02-08**—ruling out simple clock misconfiguration and indicating deliberate, field-specific manipulation of compilation metadata.

6. **Official Distribution Infrastructure Compromise**: The legitimate WhatsApp installer downloaded directly from `https://www.whatsapp.com/download` shares the **exact same impossible PE Creation Time (2054-09-13 07:53:09 UTC)** as the confirmed malicious OMEGA INFINITY 2.0 binary. Both are the same WhatsApp version, share identical SSDEEP fuzzy hashes, and carry identical valid Microsoft signatures. This demonstrates that the compromise does not originate at the endpoint—it originates at or above the level of WhatsApp's official build and distribution pipeline, potentially affecting over **2 billion users worldwide**.

7. **Content-Aware Surveillance Indicators**: The temporal correlation between authoring a theoretical NSA attack document on Overleaf (under the generic filename `main.tex`) and the autonomous malware delivery minutes later suggests real-time semantic content monitoring of cloud-hosted documents and compound behavioral trigger activation.

8. **Persistent, Adaptive Campaign**: The evolution from Version 1.0 (January 2026, triggered by opening WhatsApp) to Version 2.0 (February 2026, triggered by a CIA URL paste combined with theoretical attack document authoring) demonstrates an **ongoing, adaptive campaign** with increasingly sophisticated trigger mechanisms.

The convergence of these findings leads to an unavoidable conclusion:

Summary of Implications

- **Digital signatures are no longer trustworthy.** Valid Microsoft Authenticode signatures can exist on malicious binaries with completely different cryptographic hashes from the legitimate original.
- **Official software distribution channels are compromised.** The WhatsApp installer served from `whatsapp.com/download` shares build-origin markers with confirmed malware.
- **Hash function integrity is in question.** SHA-2 and SHA-3 collision resistance guarantees are being violated in practice, either through algorithmic breakthroughs or infrastructure-level bypass.
- **Autonomous payload delivery infrastructure exists.** Malicious binaries can be silently deployed to endpoints based on URL content pasted into messaging platforms, without any user-initiated download action.

- **Cloud-hosted document content may be monitored in real time.** Unpublished academic documents authored on cloud platforms under generic filenames can temporally correlate with autonomous malware delivery events.
- **The software supply chain is compromised at the source.** The infection does not begin at the endpoint. It begins at the build pipeline.

The cryptographic community faces a critical juncture: either our fundamental understanding of hash function security is incomplete, or we are witnessing evidence of capabilities far beyond the publicly acknowledged state-of-the-art. The shared impossible timestamp between an officially distributed binary and a confirmed malicious sample eliminates the possibility of an isolated, third-party attack—this is a compromise that touches the root of software trust infrastructure.

Complete evidence has been preserved in multiple redundant archives (Zenodo, IPFS) and all analysis reports (Triage Sandbox, VirusTotal, Threat.Rip) are publicly accessible to enable independent verification and continued investigation by the global security research community.

The OMEGA INFINITY phenomenon demands immediate, coordinated investigation by cryptographers, security researchers, platform providers, certificate authorities, and Meta/WhatsApp worldwide. The officially distributed WhatsApp installer and a confirmed malicious binary share the same impossible origin. The compromise is not at the edge. It is at the center.

Acknowledgments

The author thanks the anonymous malware analysis sandbox operators and the open security research community for providing infrastructure enabling this investigation. And finally, for יהוה to create OMEGA INFINITY 2.0. Likewise, thank you יהוה for creating OMEGA INFINITY 2.0 to destroy the Great Babel, and all cryptography, Amen.

Data Availability Statement

All forensic evidence and analysis artifacts are publicly available:

- **Zenodo:** `Proof Omega Infinity 2.rar` **IPFS:** `https://tria.ge/260215-fcn44sbw8a/behavioral1`
- **Triage (OMEGA INFINITY 2.0):** `https://tria.ge/260215-fcn44sbw8a/behavioral1`
- **Triage (Legitimate Installer):** `https://tria.ge/260215-fnpvsab12c/behavioral1`
- **Original OMEGA INFINITY v1:** DOI `10.5281/zenodo.18234712`

Conflict of Interest

The author declares no financial or commercial relationships that could be construed as potential conflicts of interest.

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* (Version 1). Zenodo. `https://doi.org/10.5281/zenodo.18234712`

[2] Triage Malware Analysis. (2026). *Automated Malware Analysis Report for WhatsApp Installer (4).exe — OMEGA INFINITY 2.0*. Sample ID: 260215-fcn44sbw8a. Retrieved from `https://tria.ge/260215-fcn44sbw8a/behavioral1`

[3] Triage Malware Analysis. (2026). *Automated Malware Analysis Report for WhatsApp Installer (Legitimate)*. Sample ID: 260215-fnpvsab12c. Retrieved from `https://tria.ge/260215-fnpvsab12c/behavioral1`

[4] Microsoft Corporation. (2024). *Authenticode Digital Signatures*. Microsoft Security Documentation. Retrieved from `https://docs.microsoft.com/en-us/windows-hardware/drivers/install/authenticode`

[5] National Institute of Standards and Technology. (2015). *Secure Hash Standard (SHS)*. FIPS PUB 180-4. U.S. Department of Commerce.

[6] Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, 212–219.

[7] Stevens, M., Bursztein, E., Karpman, P., Albertini, A., & Markov, Y. (2017). The first collision for full SHA-1. *Advances in Cryptology – CRYPTO 2017*, 570–596.

[8] MITRE Corporation. (2025). *MITRE ATT&CK® Framework — Enterprise Matrix V16*. Retrieved from `https://attack.mitre.org/`

[9] Bernstein, D. J. (2009). Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete? *Workshop Record of SHARCS*, 9, 105.

[10] Rescorla, E. (2018). *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. Internet Engineering Task Force.

[11] Benet, J. (2024). *IPFS — Content Addressed, Versioned, P2P File System*. Protocol Labs Technical Report.

[12] U.S. Central Intelligence Agency. (2024). *Electronic Reading Room — Freedom of Information Act*. Retrieved from `https://www.cia.gov/readingroom/`

Appendix: Technical Specifications

A. Analysis Environment

- Platform: Windows 11 21H2 x64
- Sandbox: Triage (`win11-20260130-en`)
- Analysis Date: 2026-02-15 05:01 UTC
- Sample Submission: 2026-02-15 04:43 UTC

B. Sample Metadata

- Target: WhatsApp Installer (4).exe
- File Size: 1,155,688 bytes (1.1 MB)
- File Type: PE32 executable (GUI) Intel 80386, for MS Windows
- Threat Score: 8/10
- Classification: Adware, Defense Evasion, Discovery, Spyware

C. Trigger URL (Decoded)

- **Host:** `www.cia.gov`
- **Path:** `/readingroom/search/site/BACKDOOR`
- **Filter:** Release date between 2024-01-01 and 2025-01-01
- **Content:** Publicly declassified FOIA documents
- **Classification:** Benign, publicly accessible government resource

D. Triage Sandbox Report References

- OMEGA INFINITY 2.0: `https://tria.ge/260215-fcn44sbw8a/behavioral1`
- Legitimate WhatsApp Installer: `https://tria.ge/260215-fnpvsab12c/behavioral1`

E. Research Repository DOI References

This is to avoid censorship by organizations such as the NSA.

- Zenodo DOI: `https://doi.org/10.5281/zenodo.18652224`
- OSF DOI: `https://doi.org/10.17605/OSF.IO/7JW58`