

<https://doi.org/10.31891/csit-2026-3-11>

UDC 004.3:004.75:004.49

## ROUTING MULTIMEDIA OBJECTS BASED ON INPUT FEATURE COMPATIBILITY

**Dmytro DENYSIUK**

PhD student

Department of Computer Engineering and  
Information Systems,

Khmelnytskyi National University

<https://orcid.org/0000-0002-7345-8341>

e-mail: [denysiuk@khnmu.edu.ua](mailto:denysiuk@khnmu.edu.ua)

**Oleksandr MELNYCHENKO**

PhD in Computer Science

Department of Computer Engineering and  
Information Systems,

Khmelnytskyi National University

<https://orcid.org/0000-0001-8565-7092>

e-mail: [melnychenko@khnmu.edu.ua](mailto:melnychenko@khnmu.edu.ua)

Received: 23/08/2026

Accepted: 14/09/2026

Published: 30/09/2026

*Identifying the format variant of a multimedia object affects the choice of processing procedure, although the filename, declared MIME type, and initial bytes may imply incompatible interpretations. This study examines the relationship between a model of format representation admissibility, a preliminary candidate identification method, and experimental responses to conflicting input features. The model represents the sources as separate sets of possible format variants and distinguishes missing information, an unrecognized value, an established contradiction, and an approximate byte match. The method selects the candidate solely from the bytes. Metadata constrains whether processing may continue and, for the ISO BMFF family, also determines the target class. A unique route is defined with a fixed target format variant, required level, procedures, and resource limits. Route uniqueness is established for a finite candidate set and well-defined procedures. For identical bytes and target class, the operational part of the route is shown to be invariant under admissible metadata changes. The candidate remains unconfirmed until the subsequent method completes a full read. The S3 test matrix contains 60 scenarios involving a changed suffix, a conflicting declared type, or a concealed prohibited suffix, with 20 scenarios per group. All 60 scenarios resulted in the expected blocking. This result corresponds to specific model constraints and algorithmic steps but does not establish format identification accuracy in a naturally occurring file population. The proposed approach separates the byte-based grounds for selection from permission to continue and preserves candidate status in the route passed between stages. This allows signature-based or statistical identification tools to be integrated while treating their output as a preliminary conclusion that requires confirmation of the object's structural admissibility.*

*Keywords: multimedia object; format identification; feature compatibility; MIME; routing; signature analysis; test scenarios.*

### Introduction

A multimedia object enters the system as a byte sequence accompanied by a filename and a declared type. These three representations may refer to the same file but have different evidential weight. A user-supplied suffix can be replaced without changing the content, and the transport field likewise describes a declared interpretation. MIME defines the syntax of this description and the media classes [1, 2], while HTTP Semantics specifies the meaning of Content-Type for a representation [3]. A valid header does not establish that all bytes conform to the format rules.

An error in format identification changes subsequent processing and may send the object to an unsuitable reader or grant it privileges associated with another class. Research on browser content-sniffing attacks has shown that discrepancies between server and client interpretations create attack opportunities [4]. Identifying the most likely MIME type is insufficient grounds for further processing in a multimedia gateway. The evidence supporting candidate selection must be distinguished from the grounds for terminating processing and the unconfirmed assumptions passed to the next procedure.

This study examines the relationship between the input feature model and the route selection method and tests this relationship experimentally in controlled scenarios. It investigates the separation of sources without arbitrary voting, the conditions for a unique preliminary result, and the correspondence between observed blocking and the prescribed responses to conflicting information. The subject of the study is the initial admission of a

© Copyright  
2026 by the author(s)



This is an Open Access article distributed  
under the terms of the [Creative Commons  
CC-BY 4.0](https://creativecommons.org/licenses/by/4.0/)

multimedia object to processing. Generating new multimedia bytes and checking the completed output define the responsibilities of the adjacent stages.

### Related work and problem statement

Standards distinguish an object's name from its media type description. Content-Disposition carries presentation information and a suggested filename [5], while its HTTP specification treats the filename as advisory and requires careful handling of the suffix [6]. MIME registration procedures and the IANA registry provide a common type vocabulary [7, 8]. Structured syntax suffixes refine media type properties but are not equivalent to filename extensions [9]. The filename and MIME value therefore require separate mappings, as direct string comparison is insufficient.

Another line of work addresses content-based identification rules. WHATWG MIME Sniffing defines contextual algorithms for analyzing declared types and byte patterns [10]. The shared-mime-info database combines filename patterns, signature conditions, and type relationships [11], while libmagic supports configurable content identification rules [12]. These mechanisms support application interoperability, but their responses require an explicit status when used in a protective processing route. A brief analysis provides no evidence that every structural element has been read completely.

Digital preservation tools address identification through maintained registries. PRONOM provides format descriptions and identifying features [13], and DROID uses this information for batch identification of formats and versions [14]. Siegfried and FIDO implement other ways of applying signature sets [15, 16]. A MITRE technical review systematizes identification challenges and the limitations of available tools [17]. The problem considered here requires a fixed rule version and reproducible grounds for selection. The number of supported type labels is insufficient to establish that these requirements are met.

Statistical classification complements signature analysis. Fileprints describes types through byte n-gram distributions [18], while feature selection and selective block reading reduce the cost of this approach [19]. Natural language processing methods have been applied to fragment classification using byte frequencies [20], and Magika uses a compact neural model for type identification [21]. These studies establish the classification capabilities of the respective models. Their accuracy estimates cannot automatically be extended to the admissibility of an entire multimedia representation.

For container formats, identification from an initial feature must be distinguished from a conclusion based on a complete read. Apache Tika documentation distinguishes rapid identification of the outer container from detectors that open it to determine its contents [22]. A comparison of ten identification tools found differences in their results and format coverage [23]. Research on polyglot files further shows that a single object can intentionally support several interpretations [24]. Selecting one detector therefore does not remove the need for a formal admission rule and subsequent confirmation of the selected candidate.

### Model of input features and route admissibility

Let  $b$  denote a finite byte sequence,  $n$  the filename,  $m$  the declared type,  $c$  the execution conditions, and  $p$  the fixed rule set. The symbol  $\perp$  denotes missing metadata. The set  $\mathcal{F}^H$  contains supported format variants with defined initial conditions, and  $L_f$  is the language of complete admissible inputs of format variant  $f$ . The basic model objects are defined by (1).

$$x = \langle b, n, m, c \rangle, \quad I(b) = \{f \in \mathcal{F}^H : b \in L_f\}, \quad I^H(b) = \{f \in \mathcal{F}^H : h_f(b) = 1\}. \quad (1)$$

The set  $I(b)$  describes confirmed membership, while  $I^H(b)$  is used to select the initial candidate. The initial condition must be necessary for membership, so  $b \in L_f \Rightarrow h_f(b) = 1$ , which implies  $I(b) \subseteq I^H(b)$ . This requirement applies within the agreed specialization of the rules and does not extend to arbitrary signature databases. Bounded reading uses an approximate predicate  $\hat{h}_f$ , which is not assumed to equal the exact predicate  $h_f$ . An empty set of approximate matches prevents selection even when a supported format variant may exist.

For the primary initial conditions  $h'_f$ , a strict total order  $<_H$  is defined. The priority rule removes selection ambiguity within the adopted specialization according to (2).

$$h_f(b) = h'_f(b) \wedge \neg \exists g <_H f : h'_g(b) = 1, \quad |I^H(b)| \leq 1. \quad (2)$$

The languages  $L_f$  include the priority condition  $h_f$  as part of full acceptance and are restricted languages of the proposed specialization. If two exact conditions held simultaneously, the later one in the order would have to negate the existence of the earlier match, contradicting its definition. Uniqueness therefore follows from the construction of the rules and does not depend on an incidental preference for one type. Polyglot files remain possible because different commonly used specifications may overlap. The model establishes a single candidate within its own set of languages and procedures.

The mappings  $F_n$  and  $F_m$  represent metadata as sets of format variants. A missing value maps to  $\mathcal{F}^H$  and does not constrain selection. A present but unrecognized value maps to the empty set and blocks the route. The relation  $\text{Comp}$  defines permitted compatibility between format variants, including related pairs such as PNG/APNG and single-frame/animated WebP. Compatibility of nonempty sets is defined by (3).

$$\text{Comp}^*(S, T) \Leftrightarrow \exists f \in S \exists g \in T : \text{Comp}(f, g), \quad F_n(\perp) = F_m(\perp) = \mathcal{F}^H. \quad (3)$$

Compatibility is reflexive and symmetric, but transitivity is not required. Checking the filename and MIME value separately for compatibility with the bytes therefore does not replace checking their compatibility with each other. All available nonempty sources are compared pairwise. A metadata value that describes only a class maps to the set of format variants in that class, including related cases specified by the rules. This mapping allows information at different levels of detail to be compared without a class label changing the byte-based candidate.

Under approximate conditions, the set  $\hat{I}^H(b)$  may contain several elements. Its first element under the same order is selected, and its compatibility is then checked without selecting again on the basis of the filename or MIME value. Candidate selection is formalized in (4).

$$\hat{I}^H(b) = \{f \in \mathcal{F}^H : \hat{h}_f(b) = 1\}, \quad \hat{f} = \min_{<_H} \hat{I}^H(b) \quad \text{for } \hat{I}^H(b) \neq \emptyset. \quad (4)$$

Exact and approximate grounds are retained as distinct states  $q_c^E$  and  $q_p^E$ . Both states designate a candidate, as input acceptance is not yet complete. If the candidate is incompatible with the metadata, processing terminates without searching other matches to satisfy the declared filename. Such rejection may reduce service coverage while preventing arbitrary metadata from selecting an interpreter. Figure 1 shows the candidate's dependence on the bytes and the checks required to construct the route.

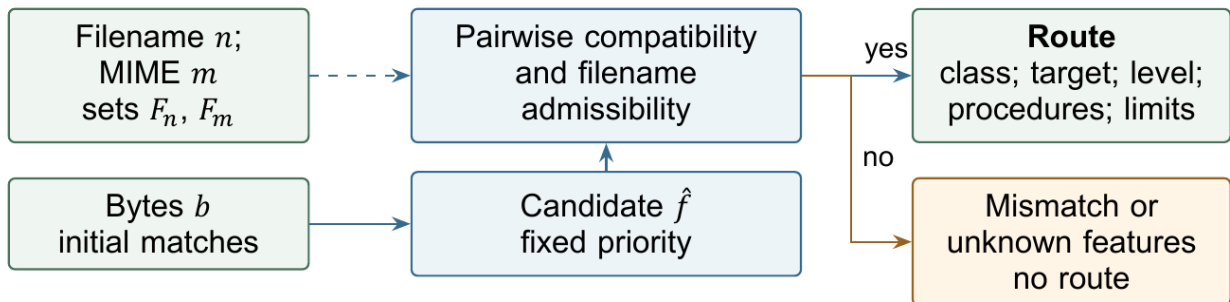


Fig. 1. Separation of candidate selection, compatibility checking, and route construction

### Consistent routing method

The method implements the model as a sequence of checks under fixed rules. The initial step checks timing conditions, the input size limit, and filename admissibility. The filename is treated as a name without path interpretation. Path separators, control characters, or a prohibited suffix anywhere in the suffix chain cause processing to terminate. For example, a safe final suffix in a name such as file.exe.png does not override the prohibition on the preceding suffix. This condition governs filename admission and does not establish the actual executable nature of the object's bytes.

Following initial admission, the sets derived from the filename, MIME value, and bytes are computed independently. Procedural independence means that information from one source is not used to compensate for missing information from another. Statistical independence is not assumed. Exact analysis first checks whether  $I^H(b)$  is empty, then checks for an unrecognized metadata value, and finally checks the compatibility of nonempty sets. This order distinguishes the absence of a supported byte-based candidate from an erroneous metadata description and makes the reason for termination reproducible.

If the comparisons succeed, the method determines the target class  $\kappa_p(x)$ . For ISO BMFF, the initial features alone identify the container family but do not distinguish audio from video. Consistent metadata may specify a class that must subsequently be confirmed by the stream composition. Partial rule mappings then determine the required level  $\rho_p$ , the target format variant  $f^*$ , the construction procedure  $a^*$ , and the rereading procedure  $a_v^*$ . The resulting route is represented by the tuple in (5).

$$\mathcal{R}_1 = \langle \hat{f}, q_{\text{cand}}^E, \kappa_p(x), f^*, \rho_p, a^*, a_v^*, p, \beta, J_1 \rangle. \quad (5)$$

In (5),  $\beta$  contains the applicable resource limits, and  $J_1$  contains explanatory messages. Before constructing the route, the method checks the domains of all mappings and whether the procedure satisfies the required level. A missing component prevents route construction. The rules prohibit lowering the required level because a procedure is unavailable. The route records the requirements for subsequent execution but does not yet establish that the environment can meet them or that an output has been produced.

For a finite candidate set, a fixed order, and terminating invoked procedures, the method returns at most one route. Technical interruption is retained as a distinct outcome because an incomplete computation does not prove structural nonconformity of the input. An exact candidate is confirmed at the next stage by a complete structural read or by the complete decoding defined for that format variant. Rejecting one approximate candidate likewise does not prove that no other exact format variant exists.

Let  $\pi_{op}$  denote the projection of the route onto the candidate, class, target, required level, procedures, and limits. For two inputs with identical  $b, c, p$ , successfully constructed routes, and the same target class, equality (6) holds.

$$\kappa_p(x_1) = \kappa_p(x_2), \quad b_1 = b_2 \Rightarrow \pi_{op}(\mathcal{R}_1(x_1)) = \pi_{op}(\mathcal{R}_1(x_2)). \quad (6)$$

The candidate depends on the bytes and the order, while the route components are functions of the candidate, class, and rules. Changes to admissible filenames and MIME values may change the explanation or terminate processing, but the operational parts of two successfully constructed routes remain identical under (6). The class constraint is necessary because of ISO BMFF. This property establishes the limits of metadata influence in the proposed method and does not extend to arbitrary metadata or external programs.

#### Experimental assessment of the relationship between the model and method

The S3 test matrix examines responses to specified contradictions. It contains 60 scenario records divided equally among three types of manipulation. The unit of analysis is a combination of input information, a specified manipulation, and an expected action. The test set was purposefully constructed at an earlier research stage without random sampling from a naturally occurring multimedia stream. Auxiliary diagnostic examples are excluded from the denominator of 60, and no new runs were performed for this article.

The scenarios involved changes to the final suffix, declared type, or an internal suffix in the filename chain through which an attacker may impose an interpretation. Blocking was specified in advance for each group. The observed final action was assessed against this requirement. Each scenario was linked to the model condition it violates and the method step that checks that condition. Table 1 compares these with the software outcome without reconstructing unrecorded formal states from that outcome.

Table 1.

**Correspondence between S3 test manipulations, model conditions, and observed actions**

| Test manipulation              | Condition checked and method action                                           | Scenarios | Expected / observed blocks |
|--------------------------------|-------------------------------------------------------------------------------|-----------|----------------------------|
| Misleading final suffix        | Filename compatibility with the byte-based candidate; termination on mismatch | 20        | 20 / 20                    |
| Conflicting declared MIME type | Type compatibility with other available features; termination on mismatch     | 20        | 20 / 20                    |
| Concealed prohibited suffix    | Admissibility of the entire suffix chain before route construction            | 20        | 20 / 20                    |
| Entire S3 matrix               | No continuation under the specified violations                                | 60        | 60 / 60                    |

All 60 scenarios resulted in the expected blocking, with no discrepancies in the final action recorded in this matrix. Thus, all 20 records in each group were blocked. The table presents the recorded aggregate result by group, and no separate new series was run to obtain this breakdown. Two groups test source comparison, while the third tests initial filename admission, so the same final outcome has different grounds in the model.

The observation supports the constraining role of metadata in the route under study because a conflicting value prevents further processing. The concealed-suffix result supports the requirement to check the entire chain, including components preceding the final suffix. These responses test the executable rejection branches of the method. They do not estimate the frequency of correct candidate identification for consistent inputs or replace a paired experiment that changes admissible metadata while keeping the bytes fixed.

Agreement in the final action does not establish the accuracy of all internal states. For exact and approximate conditions, the model permits different classes of conclusions under the same blocking outcome, which the aggregate S3 result does not distinguish. These observations therefore cannot be used to calculate format identification accuracy  $A_H$  or target class accuracy  $A_K$ . The matrix also lacks a group in which continuation is expected, which is necessary to determine the false rejection rate. The experiment confirms the responses covered by the tests, while route uniqueness and invariance are justified analytically.

#### Discussion and scope of applicability

In the proposed approach, the candidate is determined solely from the bytes, while the filename and MIME value constrain whether processing may continue. Initial analysis may use a signature-based or statistical detector provided that the limitations of its conclusion are preserved. Replacing the detector changes the set and quality of preliminary candidates. The criterion for their final confirmation and the meaning of permission to continue remain unchanged.

Compared with format registries and general-purpose detectors, the contribution concerns control of the transition to a specialized procedure and does not expand the type catalogue. Complete detection of all interpretations of polyglot files is outside the scope of this study. The method permits termination after the selected approximate candidate is rejected. This requires a separate assessment of availability on consistent real-world inputs. In a protective application, such rejection prevents unverified metadata from directing the selection of another interpreter.

Even when the input sources are consistent, no route is constructed if any component is undefined for the selected candidate and target class. In the model, this is a limit of configuration applicability and provides no grounds for declaring the bytes invalid or the candidate incorrectly identified. For example, a defined target representation without a procedure at the required level specified by the rules does not form an executable route. The same blocking outcome may therefore result from a conflicting description, an inadmissible filename, or an undefined rule mapping.

Combining these causes into one detector quality measure would conflate identification capability with the completeness of the supported configuration.

A controlled test of a specific rejection branch should establish that the remaining route prerequisites hold. Otherwise, blocking alone cannot distinguish activation of the condition under study from earlier termination. This is a logical requirement for isolating the cause being tested, which the S3 matrix does not test separately. S3 aggregates establish the expected final action, while precise attribution of each rejection to a particular condition requires intermediate evidence. An undefined route before execution is distinguished from a technical error in the selected procedure. Both prevent continuation, but the former requires additional rules and the latter requires resolving the cause of incomplete execution.

The size of the initial region analyzed,  $w_f$ , affects the completeness of the approximate match set. For fixed rules and finite windows, the cost of this analysis is bounded by the amount of data read, while filename and MIME processing also depends on their lengths. Exact initial conditions may require longer reads, so cost independence from file size has not been established for all modes. The completed matrix contains no paired comparison of  $w_f$  values and does not establish that the selected limits are universally sufficient.

The S3 matrix tested 60 predefined scenarios without estimating their prevalence in an external input stream. The aggregates do not provide complete stage-by-stage ground truth for every formal state. Testing invariance on pairs of admissible filenames and types, measuring false rejections, and analyzing ambiguous prefixes require separate studies. Processing speed, content preservation, and the correctness of the generated output were not assessed in this experiment. Practical use requires a fixed rule version and completed candidate confirmation by the adjacent method.

### Conclusions

The input feature compatibility model links format representation admissibility to the routing method while distinguishing a candidate from a confirmed format variant. Under fixed rules, uniqueness of the candidate and the partial mappings ensures at most one route. For identical bytes and target class, its operational part is independent of admissible changes to the filename and declared type. The method establishes a single unconfirmed prerequisite for the full read performed at the next stage.

The S3 test matrix produced 60 expected blocks in 60 scenarios. For the two groups of conflicting sources and the concealed prohibited suffix test, correspondence was established between model constraints, algorithmic actions, and observed blocking. The result characterizes the rejection branches examined, while identification completeness and the false rejection rate require separate control groups.

### ADDITIONAL INFORMATION

#### DECLARATION ON THE USE OF GENERATIVE ARTIFICIAL INTELLIGENCE TOOLS

During manuscript preparation, the authors used OpenAI (ChatGPT) exclusively as an aid for technical proofreading, including grammar, spelling, and language correction. Artificial intelligence was not used to generate text, structure the work, or analyze or interpret data. The authors personally checked all language revisions and take full responsibility for the final content of the publication. The S3 test results originate from the author's dissertation research, and no additional experimental runs were performed during preparation of the article.

### REFERENCES

1. Freed N., Borenstein N. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. RFC 2045. 1996. DOI: <https://doi.org/10.17487/RFC2045>
2. Freed N., Borenstein N. Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. RFC 2046. 1996. DOI: 10.17487/RFC2046.
3. Fielding R., Nottingham M., Reschke J. HTTP Semantics. RFC 9110. 2022. DOI: <https://doi.org/10.17487/RFC9110>
4. Barth A., Caballero J., Song D. Secure Content Sniffing for Web Browsers, or How to Stop Papers from Reviewing Themselves. 2009 30th IEEE Symposium on Security and Privacy. 2009. P. 360–371. DOI: <https://doi.org/10.1109/SP.2009.3>
5. Troost R., Dorner S., Moore K. Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field. RFC 2183. 1997. DOI: <https://doi.org/10.17487/RFC2183>
6. Reschke J. Use of the Content-Disposition Header Field in the Hypertext Transfer Protocol (HTTP). RFC 6266. 2011. DOI: <https://doi.org/10.17487/RFC6266>
7. Freed N., Klensin J., Hansen T. Media Type Specifications and Registration Procedures. RFC 6838. 2013. DOI: <https://doi.org/10.17487/RFC6838>
8. Internet Assigned Numbers Authority. Media Types. Electronic registry. URL: <https://www.iana.org/assignments/media-types/>
9. Hansen T., Melnikov A. Additional Media Type Structured Syntax Suffixes. RFC 6839. 2013. DOI: <https://doi.org/10.17487/RFC6839>
10. WHATWG. MIME Sniffing Standard. Living Standard. URL: <https://mimesniff.spec.whatwg.org/>
11. Leonard T. Shared MIME-info Database. Version 0.21. 2018. URL: <https://www.freedesktop.org/software/SharedMIMEInfo/>

<https://specifications.freedesktop.org/shared-mime-info/latest/>

12. File project. File: file type identification utility. Source code and documentation. URL: <https://github.com/file/file>

13. The National Archives. About PRONOM. Technical registry of file formats. URL: <https://pronom.nationalarchives.gov.uk/about>

14. The National Archives. Download DROID: file format identification tool. Software documentation. URL: <https://www.nationalarchives.gov.uk/information-management/manage-information/preserving-digital-records/droid>

15. Lehane R. Siegfried: signature-based file format identification. Source code and documentation. URL: <https://github.com/richardlehane/siegfried>

16. Open Preservation Foundation. FIDO: Format Identification for Digital Objects. Source code and documentation. URL: <https://github.com/openpreserve/fido>

17. Vidrine K. File Format Identification. MITRE. 2010. URL: <https://www.mitre.org/news-insights/publication/file-format-identification>

18. Li W.-J., Wang K., Stolfo S. J., Herzog B. Fileprints: Identifying File Types by n-Gram Analysis. Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop. 2005. P. 64–71. DOI: <https://doi.org/10.1109/IAW.2005.1495935>

19. Ahmed I., Lhee K.-S., Shin H.-J., Hong M.-P. Fast Content-Based File Type Identification. Advances in Digital Forensics VII. IFIP AICT. Vol. 361. 2011. P. 65–75. DOI: [https://doi.org/10.1007/978-3-642-24212-0\\_5](https://doi.org/10.1007/978-3-642-24212-0_5)

20. Fitzgerald S., Mathews G., Morris C., Zhulyn O. Using NLP Techniques for File Fragment Classification. Digital Investigation. 2012. Vol. 9. P. S44–S49. DOI: <https://doi.org/10.1016/j.diin.2012.05.008>

21. Fratantonio Y., Invernizzi L., Farah L. et al. MAGIKA: AI-Powered Content-Type Detection. 2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE). 2025. P. 2638–2649. DOI: <https://doi.org/10.1109/ICSE55347.2025.00158>

22. Apache Software Foundation. Apache Tika: Content Detection. Version 3.2.3. Documentation. URL: <https://tika.apache.org/3.2.3/detection.html>

23. Dubettier A., Gernot T., Giguet E., Rosenberger C. File Type Identification Tools for Digital Investigations. Forensic Science International: Digital Investigation. 2023. Vol. 46. Article 301574. DOI: <https://doi.org/10.1016/j.fsidi.2023.301574>

24. Koch L., Oesch S., Adkisson M., Erwin S., Weber B., Chaulagain A. Toward the Detection of Polyglot Files. arXiv:2203.07561v4 [cs.CR]. 2022. Preprint. DOI: <https://doi.org/10.48550/arXiv.2203.07561>

Дмитро ДЕНИСЮК, Олександр МЕЛЬНИЧЕНКО

Хмельницький національний університет

## МАРШРУТИЗАЦІЯ МУЛЬТИМЕДІЙНИХ ОБ'ЄКТІВ НА ОСНОВІ СУМІСНОСТІ ВХІДНИХ ОЗНАК

Ідентифікація варіанта формату мультимедійного об'єкта впливає на вибір процедури його оброблення, хоча ім'я файлу, заявлений тип MIME та початкові байти можуть вказувати на несумісні інтерпретації. У дослідженні розглянуто взаємозв'язок між моделлю допустимості форматного подання, методом попередньої ідентифікації кандидата та експериментальними результатами реагування на суперечливі вхідні ознаки. Модель подає джерела інформації як окремі множини можливих варіантів формату та розрізняє відсутність інформації, нерозпізнане значення, встановлену суперечність і наблизений збіг байтів. Метод обирає кандидата виключно на підставі байтів. Метадані визначають допустимість продовження оброблення, а для сімейства ISO BMFF також визначають цільовий клас. Визначено єдиний маршрут із фіксованим цільовим варіантом формату, необхідним рівнем, процедурами та ресурсними обмеженнями. Єдиність маршруту встановлено для скінченної множини кандидатів і чітко визначених процедур. Показано, що за однакових байтів і цільового класу операційна частина маршруту є інваріантною щодо допустимих змін метаданих. Кандидат залишається непідтвердженим, доки наступний метод не завершить повне зчитування об'єкта. Тестова матриця S3 містить 60 сценаріїв зі зміненням суфіксом, суперечливим заявленим типом або прихованим забороненим суфіксом — по 20 сценаріїв у кожній групі. У всіх 60 сценаріях отримано очікуване блокування. Цей результат відповідає конкретним обмеженням моделі та крокам алгоритму, однак не встановлює точності ідентифікації формату на сукупності файлів, що трапляються на практиці. Запропонований підхід відокремлює підстави для вибору, отримані з байтів, від дозволу на продовження оброблення та зберігає статус кандидата в маршруті, що передається між етапами. Це дає змогу інтегрувати засоби сигнатурної або статистичної ідентифікації, розглядаючи їхній результат як попередній висновок, що потребує підтвердження структурної допустимості об'єкта.

Ключові слова: мультимедійний об'єкт; ідентифікація формату; сумісність ознак; MIME; маршрутизація; сигнатурний аналіз; тестові сценарії.