Network Working Group
Internet-Draft
Intended status: Informational
Expires: 20 September 2026

S. Grimminck, Ed.

19 March 2026

A Standard for Safe and Reversible Sharing of Malicious URLs and Indicators

draft-grimminck-safe-ioc-sharing-05

Abstract

This document defines a consistent and reversible method for sharing potentially malicious indicators of compromise (IOCs), such as URLs, IP addresses, email addresses, and domain names. It introduces a safe obfuscation format to prevent accidental execution or activation when IOCs are displayed or transmitted. These techniques aim to standardize the safe dissemination of threat intelligence data. This specification uses the URI syntax defined in RFC 3986 and follows the key word conventions from RFC 2119.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 20 September 2026.

Copyright Notice

Table of Contents

1. Introduction 2
2. Terminology 3
3. Problem Statement 3
4. Canonical Transformation Rule 3
4.1. Step 1: Scheme 4
4.2. Step 2: Userinfo 4
4.3. Step 3: Host 4
4.4. Step 4: Stop 4
5. Formal ABNF Grammar 4
6. De-obfuscation Techniques 5
6.1. Safety Check for Reversibility 5
7. Example Use Cases 5
8. Security Considerations 6
8.1. Partial Obfuscation 6
8.2. Parser Confusion 6
8.3. De-obfuscation in Non-Executable Contexts 6
8.4. Additional Considerations 7
9. Implementation Guidance 7
10. Edge Cases and Special Handling 7
11. IANA Considerations 7
12. Test Vectors 7
13. References 8
13.1. Normative References 8
13.2. Informative References 9
Author’s Address 9

1. Introduction

The secure sharing of malicious artifacts is vital to threat intelligence, open-source intelligence (OSINT), and incident response efforts. However, sharing raw URLs, IP addresses, and email addresses associated with malware or threat actors poses a risk of accidental activation.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

This document defines a clear and reversible method for obfuscating and de-obfuscating IOCs to support safe sharing across various platforms, formats, and use cases. The requirements language (e.g., "MUST", "SHOULD") follows [RFC2119], and URI syntax adheres to [RFC3986].

2. Terminology

Obfuscating: The process of altering an indicator so that it cannot be accidentally activated or clicked. This was previously referred to as "defanging".

De-obfuscating: The process of restoring an obfuscated indicator to its original, actionable form. This was previously referred to as "refanging".

IOC: Indicator of Compromise - data such as a URL, IP address, domain name, email address, or hash associated with malicious activity.

3. Problem Statement

Inconsistent obfuscation practices hinder the reliable and automated exchange of threat intelligence. For example:

* A URL obfuscated as "h**p://example[.]com" cannot be reliably parsed by tools expecting "hxxp://example[.]com".

* An IP address obfuscated with parentheses (e.g., "192.0.2(.)1") may fail to de-obfuscate in systems expecting "[.]".

Such inconsistencies reduce the effectiveness of threat detection and response.

4. Canonical Transformation Rule

To prevent nested obfuscation (e.g., "hxxps://example[[.]]com") when an LLM or tool processes the same string twice or in the wrong order, implementations MUST apply transformations in the following strict order of operations. Implementations MUST treat already-obfuscated substrings (e.g., "[.]", "[@]") as opaque and MUST NOT apply transformations to them again; thus, the transformation is idempotent. Using encoded characters (such as %2e for ".") SHOULD be avoided to prevent ambiguity.

4.1. Step 1: Scheme

Identify and replace the scheme first. Replace "http" with "hxxp" and "https" with "hxxps". For other schemes (e.g., "ftp"), apply analogous obfuscation (e.g., "fxp").

4.2. Step 2: Userinfo

Identify the "@" symbol in the userinfo subcomponent (per [RFC3986]) and replace it with "[@]". This applies to email addresses and URIs containing userinfo (e.g., "username:password@host").

4.3. Step 3: Host

Replace all "." (period) characters in the Host subcomponent with "[.]". This applies to domain names and IPv4 addresses, including standalone values (e.g., "evil.com" or "1.1.1.1" without a scheme). IPv6 addresses enclosed in square brackets (e.g., "[2001:db8::1]") MUST retain their colon-based syntax and brackets; do not alter colons or brackets within the IPv6 literal.

4.4. Step 4: Stop

Do not process the Path, Query, or Fragment components unless they contain nested URIs that require separate obfuscation. Applying transformations beyond the Host in the primary URI may cause incorrect results.

5. Formal ABNF Grammar

The Safe-IOC format is defined using the Augmented BNF (ABNF) notation specified in [RFC5234]. This allows an LLM or tool to generate a parser that can validate whether a string is already obfuscated or needs processing.

```
; Safe-IOC obfuscation symbols
safe-scheme   = "hxxp" / "hxxps"
safe-dot      = "[" "." "]"
safe-at       = "[" "@" "]"

; Additional schemes (e.g., ftp -> fxp)
safe-other-scheme = "fxp" / "fxxps"   ; extensions for ftp, ftps
```

Figure 1

A compliant implementation MUST recognize strings containing safe-scheme, safe-dot, and safe-at as obfuscated. A string that requires obfuscation is one that contains literal "http", "https", "." in host/domain contexts, or "@" in userinfo/email contexts without the Safe-IOC bracketing.

6. De-obfuscation Techniques

Tools designed to ingest obfuscated data SHOULD automatically reverse these transformations in a deterministic manner:

* Convert "hxxp" and "hxxps" back to "http" and "https" respectively.

* Convert "[.]" back to ".".

* Convert "[@]" back to "@".

The order of these replacements does not affect the result. De-obfuscation MUST maintain the original semantics of the data to avoid misinterpretation.

6.1. Safety Check for Reversibility

De-obfuscation MUST only be performed when the output is written to a non-executable buffer (e.g., a variable, string, or file) that cannot be automatically interpreted, executed, or rendered as a clickable link by the system or application. The tool MUST NOT de-obfuscate a string if it is currently being rendered in a "live" environment (e.g., a web browser preview, an active document viewer, or any context where the resulting string could be automatically executed, resolved, or displayed as a clickable link).

De-obfuscation SHOULD only occur in controlled contexts such as:

* Command-line tools with explicit user confirmation

* Isolated analysis environments (sandboxes)

* Backend processing pipelines that do not render output to users

Accidental activation during the de-obfuscation process poses a security risk and MUST be prevented.

7. Example Use Cases

Common scenarios include:

* *OSINT Sharing:* A report lists obfuscated URLs (e.g., "hxxp://malware[.]com/payload") to prevent accidental clicks.

* *Email Communication:* Security teams share obfuscated IOCs like "attacker[@]example[.]com" in email threads.

* *Threat Intelligence Platforms:* Automated ingestion of obfuscated IPs (e.g., "192[.]0[.]2[.]1") for blocklist updates.

8. Security Considerations

While these obfuscation techniques reduce the risk of accidental activation of malicious indicators, obfuscated data SHOULD always be handled with caution.

8.1. Partial Obfuscation

A compliant tool MUST obfuscate both the scheme and the delimiters (periods, at-sign) to be considered Safe-IOC Compliant. Partial obfuscation - for example, replacing only "." with "[.]" while leaving "https" unchanged - creates a false sense of security. A user may incorrectly assume a URL is safe because the period is bracketed, when the scheme remains active and could still trigger automatic linkification or execution in some environments. Implementations MUST NOT produce partially obfuscated output when full obfuscation is intended.

8.2. Parser Confusion

Implementations that parse Safe-IOC strings may become confused by malformed or inconsistently obfuscated input. For example, "hxxps://example.com" (scheme obfuscated but dots not) or "https://example[.]com" (dots obfuscated but scheme not) are not valid Safe-IOC formats. Parsers SHOULD validate that obfuscated strings conform to the canonical transformation rule and the ABNF grammar before de-obfuscation. Rejecting or flagging ambiguous input reduces the risk of misinterpretation.

8.3. De-obfuscation in Non-Executable Contexts

As stated in Section 6, de-obfuscation MUST only occur when the result is placed in a non-executable buffer. A non-executable buffer is one that cannot be automatically interpreted by the system (e.g., as a URI to fetch, a command to run, or a link to display). Writing de-obfuscated output into a live document, rich-text editor, or browser address bar before explicit user action creates an unacceptable risk of accidental activation.

8.4. Additional Considerations

* Implementations that do not follow the canonical transformation rule (e.g., by not treating "[.]" and "[@]" as opaque) MAY produce nested or non-reversible output when obfuscation is applied repeatedly. Compliant implementations avoid this by design.

* Obfuscated URLs in PDFs may still be rendered as hyperlinks; use plain-text formatting.

* Systems processing obfuscated indicators MUST treat them as potentially harmful data, applying sandboxing or isolated environments for analysis.

* Credentials (e.g., _username:password_) SHOULD NOT be shared, even in obfuscated form, due to inherent security risks.

9. Implementation Guidance

Software designed to parse threat intelligence feeds should explicitly support these obfuscation and de-obfuscation standards. Implementations SHOULD verify correct obfuscation and de-obfuscation through unit tests and validation scripts using the test vectors in Section 12.

10. Edge Cases and Special Handling

Internationalized Domain Names (IDNs): Obfuscate punycode domains similarly (e.g., "xn--n3h[.]example[.]com").

Non-Standard URI Schemes: For schemes like "ftp", apply analogous obfuscation (e.g., "fxp://example[.]com").

IPv6 Literals in URIs: Do not alter colon characters (":") or brackets ("[", "]") in IPv6 addresses. For example, "[2001:db8::1]" MUST remain unchanged. Only scheme names or domain elements surrounding them should be obfuscated.

11. IANA Considerations

This document has no IANA actions.

12. Test Vectors

The following provides a "golden set" of inputs and expected outputs. Implementations SHOULD use these vectors to ensure correct behavior and to avoid under-obfuscation (e.g., missing email addresses) or over-obfuscation (e.g., obfuscating IPv6 colons).

* Standard URL: https://bad.com -> hxxps://bad[.]com

* URL with path: https://evil.example.com/path -> hxxps://evil[.]example[.]com/path

* Deep-link URL: https://bad.com/path/to/page?q=1#frag -> hxxps://bad[.]com/path/to/page?q=1#frag

* HTTP URL: http://attacker.com -> hxxp://attacker[.]com

* FTP URL: ftp://files.example.com/ -> fxp://files[.]example[.]com/

* IPv4 address: 1.1.1.1 -> 1[.]1[.]1[.]1

* IPv4 in URL: http://192.0.2.1 -> hxxp://192[.]0[.]2[.]1

* IPv6 in URL: http://[2001:db8::1]:8080 -> hxxp://[2001:db8::1]:8080

* IPv4-mapped IPv6: http://[::ffff:192.0.2.1] -> hxxp://[::ffff:192.0.2.1]

* Email address: phish@target.com -> phish[@]target[.]com

* Punycode domain: xn--n3h.example.com -> xn--n3h[.]example[.]com

* URL with userinfo: http://user:pass@attacker.com -> hxxp://user:pass[@]attacker[.]com

Note: The IPv6 rows demonstrate that colons and brackets within the IPv6 literal MUST NOT be altered, including IPv4-mapped IPv6 (::ffff:192.0.2.1). The deep-link row shows that Path, Query, and Fragment (per Step 4) are not processed. The Punycode row shows that IDN labels in punycode form receive the same "[.]" treatment as regular domain labels.

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008, <https://www.rfc-editor.org/rfc/rfc5234>.

13.2. Informative References

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

Author’s Address

Stefan Grimminck (editor)
Email: ietf@stefangrimminck.nl