Network Working Group
Internet-Draft
Intended status: Informational
Expires: 1 October 2026

S. Grimminck, Ed.

30 March 2026

A Standard for Safe and Reversible Sharing of Malicious URLs and Indicators
draft-grimminck-safe-ioc-sharing-06

Abstract

This document codifies a consistent and reversible convention used in the threat intelligence and security communities for sharing potentially malicious indicators of compromise (IOCs), such as URLs, IP addresses, email addresses, and domain names. It describes a safe obfuscation format that reduces the risk of accidental execution or activation when IOCs are displayed or transmitted. These conventions aim to improve interoperability among tools and feeds that exchange threat intelligence data. This specification references 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 1 October 2026.

Copyright Notice

Table of Contents

1. Introduction

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

Participants who routinely share indicators of compromise (IOCs) include security operations center (SOC) analysts, computer emergency response teams (CERTs), OSINT researchers, incident responders, and vendors of threat intelligence platforms and feeds. IOCs appear in email threads, instant-messaging channels, ticketing systems, PDF and

HTML reports, blog posts, paste sites, and machine-readable formats such as STIX/TAXII. Both human readers and automated pipelines consume this material.

When a raw URI such as "https://malicious-host.example/path" is embedded in those channels, many systems automatically detect it and render it as a clickable or otherwise actionable link. An analyst may then activate the resource unintentionally: navigating to an attacker-controlled URI can reveal the analyst's IP address and organizational affiliation, trigger delivery of malware, or alert the threat actor that a particular indicator is under active investigation. Some mail and web infrastructure pre-fetches or resolves links for scanning or preview purposes, producing the same exposure without any deliberate user action. PDF viewers and rich-text editors may turn strings that resemble URIs into hyperlinks even when the author intended plain text.

A longstanding, well-established practice in the security community is to alter IOCs so that they remain human-readable but are not treated as live URIs by typical software: for example, replacing "http" with "hxxp" and "." with "[.]". Many variant spellings exist (e.g., "h**p", different bracketing conventions for dots). That inconsistency hinders reliable parsing, exchange, and automation. This document codifies the most widely adopted spellings and a canonical order of operations so that independent implementations can interoperate. It does not introduce new URI schemes; the transformed scheme prefixes are intentionally invalid as registered URI schemes and exist only to inhibit automatic resolution.

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.

2. Terminology

Obfuscating: The process of altering an indicator so that it cannot be accidentally activated or clicked. The goal is to prevent automatic execution or resolution, not to conceal the content from human readers; the original indicator remains visually recognizable.

De-obfuscating: The process of restoring an obfuscated indicator to its original, actionable form.

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[.]example" cannot be reliably parsed by tools expecting "hxxp://example[.]example".

* 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[[.]]example") 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", "ftps"), apply the analogous community convention by substituting vowels in the scheme name with "x" in the same manner (e.g., "ftp" to "fxp", "ftps" to "fxxps").

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.example" or "198.51.100.1" 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 following uses Augmented BNF (ABNF) per [RFC5234] to illustrate tokens that commonly appear in obfuscated IOC strings. The rules are not an exhaustive registry of every scheme observed in the field; they document the widely used HTTP(S) and FTP(S) forms and the bracketed delimiters. Other schemes MAY be obfuscated using the same vowel-to-"x" substitution pattern described in Step 1 of Section 4. An implementation MAY use this grammar to help validate whether a string is already obfuscated or still requires processing.

```
; Illustrative Safe-IOC tokens (not exhaustive)
safe-scheme   = "hxxp" / "hxxps"
safe-dot      = "[" "." "]"
safe-at       = "[" "@" "]"

; Other commonly observed scheme tokens (ftp, ftps)
safe-other-scheme = "fxp" / "fxxps"
```

Figure 1

A compliant implementation MUST recognize strings containing safe-scheme, safe-dot, and safe-at as obfuscated when those tokens appear in the roles described in this document. 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 "fxp" and "fxxps" back to "ftp" and "ftps" respectively, and apply the inverse of the scheme substitution for any other obfuscated scheme encountered.

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

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

Longer scheme tokens such as "hxxps" and "fxxps" MUST be reversed before shorter prefixes that are substrings of them (e.g., reverse "hxxps" before "hxxp"). 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[.]example/payload") to prevent accidental clicks.

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

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

8. Implementation Guidance

Software designed to parse threat intelligence feeds should explicitly support these obfuscation and de-obfuscation conventions. Implementations SHOULD verify correct behavior through unit tests and validation scripts using the test vectors in Section 10.

9. Edge Cases and Special Handling

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

Non-Standard URI Schemes: For schemes such as "ftp", apply the same substitution pattern (e.g., "fxp://files[.]example/").

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.

10. Test Vectors

The following provides a "golden set" of inputs and expected outputs. Domain names use the "example" reserved space per [RFC2606]; IPv4 addresses use documentation ranges per [RFC5737]. 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.example -> hxxps://bad[.]example
* URL with path: https://evil.example/path -> hxxps://evil[.]example/path
* Deep-link URL: https://bad.example/path/to/page?q=1#frag -> hxxps://bad[.]example/path/to/page?q=1#frag
* HTTP URL: http://attacker.example -> hxxp://attacker[.]example
* FTP URL: ftp://files.example/ -> fxp://files[.]example/
* IPv4 address: 198.51.100.1 -> 198[.]51[.]100[.]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.example -> phish[@]target[.]example
* Punycode domain: xn--n3h.example.example -> xn--n3h[.]example[.]example

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

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.

11. Security Considerations

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

11.1. Relationship to the URI Scheme Registry

The strings produced by substituting characters in URI scheme names (e.g., "hxxp", "hxxps", "fxp") are not registered URI schemes and MUST NOT be treated as valid or resolvable URI schemes by generic URI parsers. Their sole purpose is operational: they are deliberately invalid so that common software does not fetch or open them as live URIs.

These spellings reflect longstanding, widespread de facto use in threat intelligence sharing, security mailing lists, and related tools. If a future assignment in the IANA URI Schemes registry were to collide with one of these strings, implementations would need to disambiguate by context (IOC obfuscation versus a registered scheme). The risk of such collision is limited by the well-established role of these strings as IOC markers in operational practice.

11.2. 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.

11.3. Parser Confusion

Implementations that parse Safe-IOC strings may become confused by malformed or inconsistently obfuscated input. For example, "hxxps://example.example" (scheme obfuscated but dots not) or "https://example[.]example" (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 illustrative ABNF before de-obfuscation. Rejecting or flagging ambiguous input reduces the risk of misinterpretation.

11.4. 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.

11.5. 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.

12. IANA Considerations

This document does not request IANA registration of any URI schemes or other protocol parameters.

The character sequences described in this document as obfuscated scheme prefixes (including but not limited to "hxxp", "hxxps", "fxp", and "fxxps") are documented here as existing operational convention in threat intelligence sharing. They are not URI schemes and are not assigned through this specification. IANA and the community are

advised that these strings are widely used in that context when evaluating future URI scheme registration requests that might collide with them.

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., Ed. 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

[RFC2606] Eastlake, D. and A. Panitz, "Reserved Top Level DNS Names", BCP 32, RFC 2606, June 1999, <https://www.rfc-editor.org/rfc/rfc2606>.

[RFC5737] Arkko, J., Cotton, M., and L. Vegoda, "IPv4 Address Blocks Reserved for Documentation", RFC 5737, January 2010, <https://www.rfc-editor.org/rfc/rfc5737>.

[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