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

S. Grimminck, Ed.

30 April 2026

# Safe and Reversible Sharing of Malicious URLs and Indicators
draft-grimminck-safe-ioc-sharing-10

## 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. The recommended form brackets the URI scheme name so that the string is not syntactically a valid URI per generic URI parsers, and extends the same bracket treatment to colons inside IPv6 literals; recognizable nested indicators within the Path, Query, or Fragment of a URI are obfuscated in place, and legacy scheme-substitution tokens are defined for de-obfuscation interoperability. Safe-IOC strings are a textual rendering convention, not URIs, and are not intended to be processed by generic URI parsers. These conventions aim to improve interoperability among tools and feeds that exchange threat intelligence data.

## 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 November 2026.

Table of Contents



## 1. Introduction

This document is an Informational specification published as an Independent Submission to the RFC series. It is not an Internet Standard and does not represent the consensus of the Internet Engineering Task Force (IETF) or any of its working groups.

The purpose of this document is to define a single transformation that converts a URL, IP address, domain name, or email address into an inert textual rendering, together with the inverse transformation that recovers the original value. Specifying both directions enables interoperability among the systems that produce, transport, or consume these renderings while preventing accidental activation by general-purpose software that handles them in transit.

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 security incident response teams (CSIRTs), 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, machine-readable formats such as STIX [STIX21] / TAXII [TAXII21], and platforms such as MISP [MISP]. 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 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 "." with "[.]" and wrapping the scheme name in brackets so that "http" becomes "[http]". Older conventions substituted characters within the scheme name (e.g., "http" to "hxxp"), but many variant spellings emerged (e.g., "h**p", "hXXp"), hindering reliable parsing and automation. This document defines a canonical form that uses uniform square-bracket wrapping for the scheme, dot, at-sign, and (inside IPv6 literals) colon delimiters, and a strict order of operations so that independent implementations can interoperate. Legacy scheme substitutions ("hxxp", "hxxps") are documented for de-obfuscation interoperability but are NOT RECOMMENDED for new output.

Safe-IOC strings produced by this specification are a textual rendering convention intended for human consumption and for lossy text channels. They are deliberately not valid URIs per [RFC3986] and are not intended to be passed to generic URI parsers, resolvers, or dereferencing libraries.

The canonical "[scheme]" form does not occupy the URI scheme namespace (see Section 4). The legacy tokens "hxxp" and "hxxps" do occupy that syntactic position. The security implications are discussed in Section 11. Implementations MUST NOT treat any obfuscated form as a resolvable URI scheme.

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.

*IP-literal:* The bracketed form of an IPv6 address as it appears in a URI authority component (e.g., "[2001:db8::1]"), per Section 3.2.2 of [RFC3986].

*Nested Indicator:* A URI, email address, or IP address literal that appears inside the Path, Query, or Fragment of another URI (for example, an open-redirect target carried as a query parameter, or an email address embedded in a redirect URL).

## 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 "[http]://example[.]example".

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

* An IPv6 literal left in colon-hexadecimal form (e.g., "[2001:db8::1]") remains parseable by URI libraries and may be auto-linked by document viewers, mail clients, and terminal emulators, producing the same activation risk that "[.]" substitution addresses for IPv4 and domain names.

Such inconsistencies reduce the effectiveness of threat detection and response.

## 4. Canonical Transformation Rule

To prevent double-bracketing (e.g., "[[https]]://example[[.]]example") when a 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 (the tokens "[scheme]", "[.]", "[@]", and "[:]") as opaque and MUST NOT apply transformations to them again; thus, the transformation is idempotent.

Percent-encoding is handled as follows. Percent-encoded delimiters in the Host or Userinfo of an input URI (for example, "%2e" for ".") MAY be decoded to their literal form before Steps 2 and 3 are applied; the resulting literal characters are then bracketed by those steps. The output of the transformation MUST NOT itself contain percent-encoded forms of the bracketed delimiters: an obfuscator that does not decode percent-encoded delimiters in the input MUST leave them as percent-encoded sequences in the output rather than emit

token-like strings such as "[%2e]". Percent-encoded content inside a Path, Query, or Fragment MUST NOT be decoded during obfuscation, so that independent implementations produce the same output for the same input.

### 4.1. Step 1: Scheme

Identify the URI scheme and wrap it in square brackets. For example, "http" becomes "[http]", "https" becomes "[https]", "ftp" becomes "[ftp]". The leading "[" character is not permitted in a URI scheme name (Section 3.1 of [RFC3986]), so the result cannot be mistaken for a valid URI by compliant parsers. Because the transformation is purely syntactic wrapping, it extends to any current or future scheme without requiring a per-scheme mapping table. Only the scheme name is wrapped; the scheme delimiter ("://" for hierarchical URIs, ":" for schemes such as mailto: or tel:) is preserved unchanged. The case of the scheme name is preserved verbatim so that the transformation is reversible byte-for-byte; applications that require case-folded schemes can normalize before or after applying this transformation.

The legacy tokens "hxxp" and "hxxps" are in widespread operational use as obfuscated forms of "http" and "https" respectively. Implementations encountering these tokens in existing data SHOULD recognize them as obfuscated indicators during de-obfuscation (see Section 6). These legacy forms are NOT RECOMMENDED for new output. No other legacy substitutions (e.g., "h**p", "fxxps") are defined by this specification.

### 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). IPv4 addresses use dotted-decimal notation that overlaps with domain name syntax, so the "[.]" substitution applies to them naturally. Inside an IPv6 literal, replace every ":" (colon) with "[:]"; if the literal contains an embedded IPv4 address (e.g., "::ffff:192.0.2.1" per Section 2.5.5 of [RFC4291]), each "." in that embedded IPv4 MUST also be replaced with "[.]". The outer square brackets that surround an IP-literal in URI syntax (Section 3.2.2 of [RFC3986]) are part of the URI grammar, not part of the obfuscation,

and MUST be preserved verbatim. Port numbers following the host (e.g., ":8080" or the port following an IP-literal) are not part of the Host subcomponent and MUST NOT be altered.

Zone identifiers (per [RFC4007]) may appear attached to an IPv6 address as the "%<zone-id>" suffix in bare form, or as the "%25<zone-id>" suffix inside a URI IP-literal. The URI-literal encoding was originally defined in [RFC6874], which has since been obsoleted by [RFC9844] without a replacement URI syntax; the "%25<zone-id>" form remains in operational use and is treated here as the de facto encoding. Whichever form is present is preserved verbatim by this transformation.

Bare IPv6 literals without surrounding URI brackets (e.g., "2001:db8::1" appearing on its own line in a report) MUST receive the same colon-bracketing treatment. Implementations SHOULD recognize bare IPv6 by the presence of either a "::" shorthand or the fully expanded eight-group colon-hexadecimal form so that strings such as "host:port" (a single colon) are not misclassified. Intermediate forms (more than one colon but neither a "::" shorthand nor a full eight-group expansion, e.g., a malformed "2001:db8:1:2:3:4:5") are not recognized as IPv6 by this rule and are left unchanged.

For schemes whose content after the scheme delimiter is not a hierarchical authority (e.g., "mailto:" per [RFC6068], or "urn:"), implementations apply Step 2 to every "@" and Step 3 to every "." within the scheme body, up to the first "?" or "#" delimiter.

When a scheme is present, host subcomponent boundaries are determined by the URI syntax defined in [RFC3986]. For bare indicators without a scheme (e.g., a standalone domain or IP address), host identification is implementation-dependent; implementations SHOULD apply reasonable heuristics such as recognizing dotted-decimal IPv4 addresses, colon-hexadecimal IPv6 addresses, and domain-name patterns.

### 4.4. Step 4: Nested Indicators

Implementations MUST NOT apply Steps 2 and 3 wholesale to the Path, Query, or Fragment of the primary URI; doing so would alter semantics (for example, by bracketing dots in file extensions or query values) and is not reversible without ambiguity.

Implementations locate nested indicators by scanning the literal (undecoded) text of the Path, Query, and Fragment for spans matching one of the grammars listed below; each matched span is obfuscated in place by recursively applying Steps 1 through 3. Percent-encoded nested indicators are not matched by this step, and implementations

that wish to surface them SHOULD emit them as separate obfuscated indicators. For the purposes of this step, a nested indicator is one of the following:

* A URI with a scheme (including "mailto:" URIs).

* A bare email address (an addr-spec per Section 3.4.1 of [RFC5322]).

* A bare IPv4 or IPv6 address literal.

Only spans matching one of the grammars above are obfuscated; dots, "@", and ":" characters that appear elsewhere in the Path, Query, or Fragment MUST be preserved verbatim.

Implementations MAY instead extract each nested indicator and emit it as a separate obfuscated indicator alongside the primary one.

## 5. Formal ABNF Grammar

The following uses Augmented BNF (ABNF) per [RFC5234] to define the tokens used in obfuscated IOC strings. The scheme production matches the syntax in Section 3.1 of [RFC3986]. An implementation MAY use this grammar to validate whether a string is already obfuscated or still requires processing.

```
safe-scheme    = "[" scheme "]"
scheme         = ALPHA *( ALPHA / DIGIT / "+" / "-" / "." )
legacy-scheme  = "hxxp" / "hxxps"
safe-dot       = "[" "." "]"
safe-at        = "[" "@" "]"
safe-colon     = "[" ":" "]"

obf-label      = 1*( ALPHA / DIGIT / "-" )
obf-host       = obf-label 1*( safe-dot obf-label )
obf-local      = 1*( ALPHA / DIGIT / "." / "_" / "%"
                     / "+" / "-" )
obf-email      = obf-local safe-at obf-host
```

Figure 1

The obf-label production permits any character allowed in a DNS label but does not enforce DNS label structure rules; applications that will resolve the recovered host SHOULD additionally enforce the no-leading-or-trailing-hyphen and 63-octet-maximum rules per Section 3.5 of [RFC1035]. The obf-host production matches a host of two or more labels; a single-label host has no dots to bracket and is left unchanged by Step 3. The composition of a full URI (scheme,

authority, path, query, fragment) and of an obfuscated IPv6 literal is not restated here; both are defined algorithmically in Section 4 and rely on the URI grammar of [RFC3986] and the address grammar of [RFC4291].

A compliant implementation MUST recognize a string as obfuscated when it contains any of the tokens safe-scheme, legacy-scheme, safe-dot, safe-at, or safe-colon in the roles described in this document.

## 6. De-obfuscation Techniques

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

* Strip enclosing brackets from "[scheme]" to recover the original scheme name (e.g., "[https]" becomes "https").

* Convert legacy token "hxxps" back to "https".

* Convert legacy token "hxxp" back to "http".

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

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

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

The substitutions listed above are non-overlapping and may be applied in any order, with one exception: for legacy tokens, longer strings 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.

Safe-IOC strings may contain two kinds of square brackets: the URI IP-literal brackets (Section 3.2.2 of [RFC3986]) and the obfuscation tokens "[scheme]", "[.]", "[@]", and "[:]". When an IPv6 URI is obfuscated, adjacent "[" or "]" characters may appear (e.g., "[http]://[[:][:]1]"). These are adjacent tokens, not nested brackets. Parsers SHOULD tokenize left-to-right, attempting to match the obfuscation tokens first and treating any remaining "[" or "]" as a URI IP-literal delimiter.

### 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

## 7. Example Use Cases

Common scenarios include:

* *OSINT Sharing:* A report lists obfuscated URLs (e.g., "[http]://malware[.]example/payload") to prevent accidental clicks.
* *Email Communication:* Security teams share obfuscated IOCs like "attacker[@]example[.]example" or "[mailto]:attacker[@]example[.]com" in email threads.
* *Threat Intelligence Platforms:* Automated ingestion of obfuscated IPs (e.g., "192[.]0[.]2[.]1" or "[http]://[2001[:]db8[:][:]1]:8080/") for blocklist updates.

Safe-IOC strings are intended for human-readable contexts (reports, email, tickets, chat) and for free-text fields in structured formats such as STIX 2.1 [STIX21]; they are not a substitute for the typed observable and indicator objects those formats provide.

## 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):* Apply Step 3 to the IDN as presented, in either A-label (ASCII-Compatible Encoding, "xn--" prefixed) or U-label (native Unicode) form per [RFC5890]; for example, "xn--n3h.example" becomes "xn--n3h[.]example".

*Non-Standard URI Schemes:* The "[scheme]" canonical form extends to any URI scheme. For example, "smb://fileserver.example/share" becomes "[smb]://fileserver[.]example/share".

*IPv6 Literals in URIs:* Bracket every ":" inside the IP-literal as "[:]" and every "." in any embedded IPv4 portion as "[.]". Preserve the outer URI brackets, any "%25<zone-id>" suffix, and any following port. For example, "http://[2001:db8::1]:8080" becomes "[http]://[2001[:]db8[:][:]1]:8080", and "[::ffff:192.0.2.1]" becomes "[[:][:]ffff[:]192[.]0[.]2[.]1]". The "[[" and "]]" sequences that appear when an IP-literal begins or ends with "::" are adjacent tokens, not nested brackets.

*Bare IPv6 Literals:* Addresses appearing without URI brackets (e.g., "2001:db8::1" on its own line) are obfuscated by colon-bracketing alone: "2001[:]db8[:][:]1". Implementations SHOULD restrict bare-IPv6 detection to strings that contain "::" or the fully expanded eight-group form so that "host:port" and similar single-colon constructs are not misclassified.

## 10. Test Vectors

The following provides a "golden set" of inputs and expected outputs in JSON form, intended to be consumed directly by automated test harnesses. Each entry has a "label", an "input", an "operation" ("obfuscate" for the forward transformation defined in Section 4, "deobfuscate" for the inverse transformation defined in Section 6), and the "expected" output. Domain names use the "example" reserved space per [RFC2606]; IPv4 addresses use documentation ranges per [RFC5737]; IPv6 addresses use the documentation prefix per [RFC3849].

The vectors fall into four groups. The forward URI rows (standard-url through scheme-case-preserved) exercise Steps 1 through 3 on full URIs and on bare indicators, including IPv4, IPv6 in every form covered by Section 4.3, zone identifiers, userinfo, and ports. The Step 4 rows (nested-url-in-query, nested-email-in-query) exercise in-place obfuscation of nested indicators. The idempotency rows confirm that applying the transformation to already-obfuscated input produces no change. The deobfuscate rows exercise the inverse transformation defined in Section 6, including the legacy "hxxps" recognition rule.

```
[
  {"label": "standard-url", "operation": "obfuscate",
   "input": "https://bad.example",
   "expected": "[https]://bad[.]example"},
  {"label": "url-with-path", "operation": "obfuscate",
   "input": "https://evil.example/path",
   "expected": "[https]://evil[.]example/path"},
  {"label": "deep-link-url", "operation": "obfuscate",
   "input": "https://bad.example/path/to/page?q=1#frag",
   "expected": "[https]://bad[.]example/path/to/page?q=1#frag"},
  {"label": "http-url", "operation": "obfuscate",
   "input": "http://attacker.example",
   "expected": "[http]://attacker[.]example"},
  {"label": "ftp-url", "operation": "obfuscate",
   "input": "ftp://files.example/",
   "expected": "[ftp]://files[.]example/"},
  {"label": "mailto", "operation": "obfuscate",
   "input": "mailto:user@example.com",
   "expected": "[mailto]:user[@]example[.]com"},
  {"label": "ipv4-address", "operation": "obfuscate",
   "input": "198.51.100.1",
   "expected": "198[.]51[.]100[.]1"},
  {"label": "ipv4-in-url", "operation": "obfuscate",
   "input": "http://192.0.2.1",
   "expected": "[http]://192[.]0[.]2[.]1"},
  {"label": "ipv6-in-url", "operation": "obfuscate",
   "input": "http://[2001:db8::1]:8080",
   "expected": "[http]://[2001[:]db8[:][:]1]:8080"},
  {"label": "ipv6-full-form", "operation": "obfuscate",
   "input": "http://[2001:db8:0:0:0:0:0:1]/",
   "expected": "[http]://[2001[:]db8[:]0[:]0[:]0[:]0[:]0[:]0[:]1]/"},
  {"label": "ipv4-mapped-ipv6", "operation": "obfuscate",
   "input": "http://[::ffff:192.0.2.1]",
   "expected": "[http]://[[:][:]ffff[:]192[.]0[.]2[.]1]"},
  {"label": "ipv6-with-zone-id", "operation": "obfuscate",
   "input": "http://[2001:db8::1%25eth0]/",
   "expected": "[http]://[2001[:]db8[:][:]1%25eth0]/"},
  {"label": "bare-ipv6", "operation": "obfuscate",
   "input": "2001:db8::1",
   "expected": "2001[:]db8[:][:]1"},
  {"label": "bracketed-bare-ipv6", "operation": "obfuscate",
   "input": "[2001:db8::1]",
   "expected": "[2001[:]db8[:][:]1]"},
  {"label": "bare-ipv6-with-zone", "operation": "obfuscate",
   "input": "2001:db8::1%eth0",
   "expected": "2001[:]db8[:][:]1%eth0"},
  {"label": "email-address", "operation": "obfuscate",
   "input": "phish@target.example",
```

```
     "expected": "phish[@]target[.]example"},
    {"label": "punycode-domain", "operation": "obfuscate",
     "input": "xn--n3h.example",
     "expected": "xn--n3h[.]example"},
    {"label": "url-with-userinfo", "operation": "obfuscate",
     "input": "http://user:pass@attacker.example",
     "expected": "[http]://user:pass[@]attacker[.]example"},
    {"label": "bare-domain-with-port", "operation": "obfuscate",
     "input": "evil.example:443",
     "expected": "evil[.]example:443"},
    {"label": "scheme-case-preserved", "operation": "obfuscate",
     "input": "HTTPS://bad.example",
     "expected": "[HTTPS]://bad[.]example"},
    {"label": "nested-url-in-query", "operation": "obfuscate",
     "input":
       "http://example.com/r?url=http://evil.example",
     "expected":
       "[http]://example[.]com/r?url=[http]://evil[.]example"},
    {"label": "nested-email-in-query", "operation": "obfuscate",
     "input":
       "http://example.com/?contact=abuse@evil.example",
     "expected":
       "[http]://example[.]com/?contact=abuse[@]evil[.]example"},
    {"label": "idempotency-check", "operation": "obfuscate",
     "input": "[https]://bad[.]example",
     "expected": "[https]://bad[.]example"},
    {"label": "ipv6-idempotency", "operation": "obfuscate",
     "input": "[http]://[2001[:]db8[:][:]1]:8080",
     "expected": "[http]://[2001[:]db8[:][:]1]:8080"},
    {"label": "legacy-deobfuscation", "operation": "deobfuscate",
     "input": "hxxps://bad[.]example",
     "expected": "https://bad.example"},
    {"label": "legacy-deobfuscation-bare", "operation": "deobfuscate",
     "input": "hxxps://bad.example",
     "expected": "https://bad.example"},
    {"label": "case-preserved-roundtrip", "operation": "deobfuscate",
     "input": "[HTTPS]://bad[.]example",
     "expected": "HTTPS://bad.example"},
    {"label": "ipv6-uri-deobfuscation", "operation": "deobfuscate",
     "input": "[http]://[2001[:]db8[:][:]1]:8080",
     "expected": "http://[2001:db8::1]:8080"},
    {"label": "mailto-deobfuscation", "operation": "deobfuscate",
     "input": "[mailto]:user[@]example[.]com",
     "expected": "mailto:user@example.com"}
  ]
```

Figure 2

## 11. Security Considerations

While these obfuscation techniques reduce the risk of accidental activation, systems processing obfuscated indicators SHOULD apply the safeguards described in this section.

### 11.1. Relationship to the URI Scheme Registry

The canonical "[scheme]" form does not occupy the URI scheme namespace (see Section 4). Software that encounters bracketed schemes MUST NOT attempt to resolve or dereference them.

The legacy tokens "hxxp" and "hxxps" do occupy the syntactic position of a URI scheme. Provisional entries for both names appear in the "Uniform Resource Identifier (URI) Schemes" registry per [RFC7595]. Implementations should consider surrounding context when disambiguating these tokens.

### 11.2. Partial Obfuscation

When a scheme is present, a compliant tool MUST obfuscate both the scheme (Step 1) and the host-level delimiters (Steps 2-3). Partial obfuscation (for example, replacing only "." with "[.]" while leaving "https" unchanged) creates a false sense of security: the scheme remains active and could still trigger automatic linkification or execution. Implementations MUST NOT produce partially obfuscated output when full obfuscation is intended. For bare indicators without a scheme (e.g., a standalone IP address or email address), host-level and userinfo obfuscation (Steps 2-3) alone constitutes valid Safe-IOC output.

### 11.3. Parser Confusion

Implementations that parse Safe-IOC strings may become confused by malformed or inconsistently obfuscated input. For example, "[https]://example.example" (scheme obfuscated but dots not) or "https://example[.]example" (dots obfuscated but scheme not) are not canonical Safe-IOC output and SHOULD be treated as partially obfuscated: such strings retain activation risk and implementations SHOULD either complete the obfuscation before display or flag the input as non-conforming.

### 11.4. De-obfuscation in Non-Executable Contexts

The non-executable-buffer requirement on de-obfuscation is stated normatively in Section 6.1. A non-executable buffer is one that cannot be automatically interpreted by the system (for example, 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.

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

* Systems processing obfuscated indicators MUST treat them as untrusted 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.

* Analysts accessing de-obfuscated indicators for investigation SHOULD do so within protected environments such as containers or dedicated analysis VMs to limit exposure to exploits.

* If a de-obfuscated indicator must be forwarded to a system that is not prepared to handle unsafe input, it SHOULD be re-obfuscated before being passed on.

## 12. IANA Considerations

This document has no IANA actions.

[RFC Editor: Please remove this section before publication.]

## 13. References

### 13.1. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987, <https://www.rfc-editor.org/rfc/rfc1035>.

[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>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006, <https://www.rfc-editor.org/rfc/rfc4291>.

[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>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, October 2008, <https://www.rfc-editor.org/rfc/rfc5322>.

[RFC6068] Duerst, M., Masinter, L., and J. Zawinski, "The 'mailto' URI Scheme", RFC 6068, October 2010, <https://www.rfc-editor.org/rfc/rfc6068>.

[RFC7595] Thaler, D., Ed., Hansen, T., and T. Hardie, "Guidelines and Registration Procedures for URI Schemes", BCP 35, RFC 7595, June 2015, <https://www.rfc-editor.org/rfc/rfc7595>.

[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>.

## 13.2. Informative References

[RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and B. Zill, "IPv6 Scoped Address Architecture", RFC 4007, March 2005, <https://www.rfc-editor.org/rfc/rfc4007>.

[RFC6874] Carpenter, B., Cheshire, S., and R. Hinden, "Representing IPv6 Zone Identifiers in Address Literals and Uniform Resource Identifiers", RFC 6874, February 2013, <https://www.rfc-editor.org/rfc/rfc6874>.

[RFC9844] Carpenter, B. and R. Hinden, "Entering IPv6 Zone Identifiers in User Interfaces", RFC 9844, August 2025, <https://www.rfc-editor.org/rfc/rfc9844>.

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

[RFC3849] Huston, G., Lord, A., and P. Smith, "IPv6 Address Prefix Reserved for Documentation", RFC 3849, July 2004, <https://www.rfc-editor.org/rfc/rfc3849>.

[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>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010, <https://www.rfc-editor.org/rfc/rfc5890>.

[STIX21] Jordan, B., Ed., Piazza, R., Ed., and T. Darley, Ed., "STIX Version 2.1", OASIS Standard stix-v2.1-os, 10 June 2021, <https://docs.oasis-open.org/cti/stix/v2.1/os/stix-v2.1-os.html>.

[TAXII21] Jordan, B., Ed. and D. Varner, Ed., "TAXII Version 2.1", OASIS Standard taxii-v2.1-os, 10 June 2021, <https://docs.oasis-open.org/cti/taxii/v2.1/os/taxii-v2.1-os.html>.

[MISP] MISP Project, "MISP - Malware Information Sharing Platform and Threat Sharing", <https://www.misp-project.org/>.

## Acknowledgements

The author thanks Frank Denis, Martin J. D端rst, Ted Hardie, Graham Klyne, Eliot Lear, Barry Leiba, and Kathleen Moriarty for their review and feedback during the development of this document. The author also thanks the participants of the IETF SecDispatch and practical-cybersecurity discussions for early input on this work.

## Author’s Address

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