        Resource-Oriented Lightweight Information Exchange (ROLIE)

Abstract

   This document defines a resource-oriented approach for security
   automation information publication, discovery, and sharing. Using
   this approach, producers may publish, share, and exchange
   representations of software descriptors, security incidents, attack
   indicators, software vulnerabilities, configuration checklists, and
   other security automation information as web-addressable resources.
   Furthermore, consumers and other stakeholders may access and search
   this security information as needed, establishing a rapid and
   on-demand information exchange network for restricted internal use or
   public access repositories. This specification extends the Atom
   Publishing Protocol and Atom Syndication Format to transport and
   share security automation resource representations.

Status of This Memo

   This is an Internet Standards Track document.

   This document is a product of the Internet Engineering Task Force
   (IETF). It represents the consensus of the IETF community. It has
   received public review and has been approved for publication by the
   Internet Engineering Steering Group (IESG). Further information on
   Internet Standards is available in Section 2 of RFC 7841.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   https://www.rfc-editor.org/info/rfc8322.

Table of Contents

1. Introduction

   This document defines a resource-oriented approach to security
   automation information sharing that follows the Representational
   State Transfer (REST) architectural style [REST]. In this approach,
   computer security resources are maintained in web-accessible
   repositories structured as Atom Syndication Format [RFC4287] Feeds.
   Within a given Feed, which may be requested by the consumer,
   representations of specific types of security automation information
   are organized, categorized, and described. Furthermore, all
   collections available to a given user are discoverable, allowing the
   consumer to search all available content they are authorized to view,
   and to locate and request the desired information resources. Through
   the use of granular authentication and access controls, only
   authorized consumers may be permitted the ability to read or write to
   a given Feed.

   The goal of this approach is to increase the communication and
   sharing of security information between providers and consumers that
   can be used to automate security processes (e.g., incident reports,
   vulnerability assessments, configuration checklists, and other
   security automation information). Such sharing allows human

operators and computer systems to leverage this standardized
communication system to gather information that supports the
automation of security processes.

To support new types of security automation information being used as
time goes on, this specification defines a number of extension points
that can be used either privately or globally. These global
extensions are IANA-registered by Resource-Oriented Lightweight
Information Exchange (ROLIE) extension specifications and provide
enhanced interoperability for new use cases and domains. Sections 5
and 6 of this document define the requirements for XML
representations of ROLIE; other equivalent representations (e.g.
JSON) may be described by other documents. An overview of the
extension system is provided in Section 7. Implementers seeking to
provide support for specific security automation information types
should refer to the specification for that domain as described by the
IANA registry found in Section 8.4.

2. Terminology

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.

The previous key words are used in this document to define only the
requirements for implementations of this specification and are not
used for recommendations or requirements for the usage of ROLIE. (In
other words, a programmer of a ROLIE server MUST implement a given
feature, but a user of that ROLIE server needn't use that feature.)

Definitions for some of the common computer-security-related
terminology used in this document can be found in Section 2 of
[RFC7970].

The following term is unique to this specification:

Information Type: A class of security automation information having
   one or more associated data models. Often, such security
   automation information is used in the automation of a security
   process. See Section 7.1.2 for more information.

3. XML-Related Conventions

3.1. XML Namespaces

   This specification uses XML namespaces [W3C.REC-xml-names-20091208]
   to uniquely identify XML element names. It uses the following
   namespace prefix mappings for the indicated namespace URI:

   o "app" is used for the "https://www.w3.org/2007/app" namespace
      defined in [RFC5023].

   o "atom" is used for the "https://www.w3.org/2005/Atom" namespace
      defined in [RFC4287].

   o "rolie" is used for the "urn:ietf:params:xml:ns:rolie:1.0"
      namespace defined in Section 8.1 of this specification.

3.2. RELAX NG Compact Schema

   Some sections of this specification are illustrated with fragments of
   a non-normative RELAX NG Compact Schema [RELAX-NG]. The text of this
   specification provides the definition of conformance. Schema for the
   "https://www.w3.org/2007/app" and "https://www.w3.org/2005/Atom"
   namespaces appear in Appendix B of [RFC5023] and Appendix B of
   [RFC4287], respectively.

   A complete informative RELAX NG Compact Schema for the new elements
   introduced by ROLIE is provided in Appendix A of this document.

4. Background and Motivation

   In order to automate security processes, tools need access to
   sufficient sources of structured security information that can be
   used to drive security processes. Thus, security information sharing
   is one of the core components of automating security processes.
   Vulnerabilities, configurations, software identification, security
   incidents, and patch data are just a few of the classes of
   information that are shared today to enable effective security on a
   wide scale. However, as the scale of defense broadens as networks
   become larger and more complex, and the volume of information to
   process makes humans-in-the-loop difficult to scale, the need for
   automation and machine-to-machine communication becomes increasingly
   critical.

ROLIE seeks to address this need by providing four major information-
sharing benefits:

Extensible information type categories and format agnosticism: ROLIE
   is not bound to any given data format or category of information.
   Instead, information categories are extensible, and Entries
   declare the format of the referenced data. In cases where several
   formats or serializations are available, ROLIE can use link
   relations to communicate how a consumer can access these formats.
   For example, clients may request that a given resource
   representation be returned as XML, JSON, or in some other format
   or serialization. This approach allows the provider to support
   multiple isomorphic formats, allowing the consumer to select the
   most suitable version.

Open and distributed information sharing: Using the Atom Publishing
   Protocol (AtomPub), ROLIE Feeds can easily aggregate Feeds and
   accept information posted to them from other sources. Webs of
   communicating ROLIE servers form ad hoc sharing communities,
   increasing data availability and the ability to correlate linked
   data across sources for participating consumers. ROLIE servers
   needn't be distributed, however, as large ROLIE repositories can
   function as a central collection or federated collections.

Stateless communication model: ROLIE, as a RESTful system, is
   stateless. That is, the server doesn't keep track of client
   sessions but rather uses link relations for state transitions. In
   practice, this means that any consumer can find and share
   information at any organizational level and at any time without
   needing to execute a long series of requests.

Information discovery and navigation: ROLIE provides a number of
   mechanisms to allow clients to programmatically discover and
   navigate collections of information in order to dynamically
   discover new or revised content. Extensible information types and
   other categories provide one way of determining content that is
   desirable. Link elements, each with a target URI and an
   established relationship type, provide a means for ROLIE providers
   to link other information that is relevant to the current Entry
   or Feed.

These benefits result in an information-sharing protocol that is
lightweight, interactive, open, and, most importantly, machine
readable.

The requirements in this specification are broken into two major
sections: extensions to AtomPub [RFC5023] and extensions to the Atom
Syndication Format [RFC4287]. All normative requirements in AtomPub

and Atom Syndication are inherited from their respective
specifications and apply here unless the requirement is explicitly
overridden in this document. In this way, this document may upgrade
the requirement (e.g., make a "SHOULD" a "MUST") but will never
downgrade a given requirement (e.g., make a "MUST" a "SHOULD").

5. ROLIE Requirements for the Atom Publishing Protocol

   This section describes a number of restrictions of, and extensions
   to, AtomPub [RFC5023] that define the use of AtomPub in the context
   of a ROLIE-based solution. The normative requirements in this
   section are generally oriented towards client and server
   implementations. An understanding of the AtomPub specification
   [RFC5023] is helpful to understand the requirements in this section.

5.1. AtomPub Service Documents

   As described in Section 8 of [RFC5023], a Service Document is an
   XML-based document format that allows a client to dynamically
   discover the Collections provided by a publisher. A Service Document
   consists of one or more "app:workspace" elements that may each
   contain a number of "app:collection" elements.

   The general structure of a Service Document is as follows (from
   Section 4.2 of [RFC5023]):

```
    Service
       o- Workspace
       |     |
       |     o- Collection
       |     |     |
       |     |     o- URI, categories, media types
       |     |
       |     o- ...
       |
       o- Workspace
       |      |
       |      o- Collection
       |      |     |
       |      |     o- URI, categories, media types
       |      |
       |      o- ...
       |
       o- ...
```
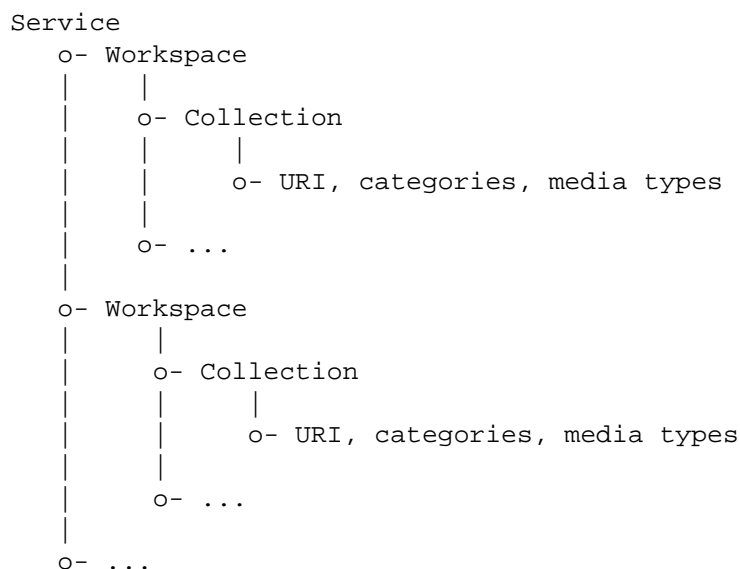
   Note that the Internationalized Resource Identifiers (IRIs) in the
   original diagram have been replaced with URIs.

5.1.1. Use of the "app:workspace" Element

   In AtomPub, a workspace, represented by the "app:workspace" element,
   describes a group of one or more Collections. Building on the
   AtomPub concept of a workspace, in ROLIE a workspace represents an
   aggregation of Collections pertaining to security automation
   information resources. This specification does not restrict the
   number of workspaces that may be in a Service Document or the
   specific Collections to be provided within a given workspace.

   A ROLIE implementation can host Collections containing both public
   and private information Entries. It is suggested that
   implementations segregate Collections into different "app:workspace"
   elements by their client access requirements. With proper naming of
   workspaces, this reduces the amount of trial and error a human user
   would need to utilize to discover accessible Collections.

5.1.2. Use of the "app:collection" Element

   In AtomPub, a Collection in a Service Document, represented by the
   "app:collection" element, provides metadata that can be used to point
   to a specific Atom Feed that contains information Entries that may be
   of interest to a client. The association between a Collection and a
   Feed is provided by the "href" attribute of the "app:collection"
   element. Building on the AtomPub concept of a Collection, in ROLIE a
   Collection represents a pointer to a group of security automation
   information resources pertaining to a given type of security
   automation information. Collections are represented as Atom Feeds as
   per RFC 5023. Requirements specific to Atom Feed are defined in
   Section 6.1.

   ROLIE defines specialized data requirements for Collections, Feeds,
   and Entries containing data related to security automation. The
   difference between a ROLIE Collection and a non-ROLIE Collection
   defined in a Service Document can be determined as follows:

   ROLIE Collection: An app:collection is considered a ROLIE Collection
      when it contains an "app:categories" element that contains only
      one "atom:category" element with a "scheme" attribute value of
      "urn:ietf:params:rolie:category:information-type". Further, this
      category has an appropriate "term" attribute value as defined in
      Section 7.1.1. This ensures that a given Collection corresponds
      to a specific type of security automation information.

   Non-ROLIE Collection: An app:collection is considered a non-ROLIE
      Collection when it does not contain an "atom:category" element
      with a "scheme" attribute value of
      "urn:ietf:params:rolie:category:information-type".

By distinguishing between ROLIE and non-ROLIE Collections in this
way, implementations supporting ROLIE can host Collections pertaining
to security automation information alongside Collections of other
non-ROLIE information within the same AtomPub instance.

The following are additional requirements on the use of the
"app:collection" element for a ROLIE Collection:

o The child "atom:category" elements contained in the
   "app:categories" element MUST be the same set of "atom:category"
   elements used in the Atom Feed resource referenced by the
   "app:collection" element's "href" attribute value. This ensures
   that the category metadata associated with the Collection and the
   associated Feed is discoverable in both of these resources.

o The "app:categories" element in an app:collection MAY include
   additional "atom:category" elements using a scheme other than
   "urn:ietf:params:rolie:category:information-type". This allows
   other category metadata to be included.

5.1.3. Service Document Discovery

The Service Document serves as the "head" of a given ROLIE
repository: from the Service Document, all other repository content
can be discovered. A client will need to determine the URL of this
Service Document to discover the Collections provided by the
repository. The client might determine the URL from a web page,
based on out-of-band communication, or through a "service" link
relation in a Feed or Entry Document that the client has already
retrieved. The latter is a typical scenario if the client learns of
a specific Feed or Entry through an out-of-band mechanism and wishes
to discover additional information provided by the repository.

This document does not provide a fully automated discovery mechanism.
A mechanism may be defined in the future that allows automated
clients to discover the URL to use to retrieve a ROLIE Service
Document representing the head of the ROLIE repository.

5.2. Category Documents

As described in Section 7 of [RFC5023], a Category Document is an
XML-based document format that allows a client to dynamically
discover the categories used within AtomPub Service Documents, Atom
Syndication Feeds, and Entry Documents provided by a publisher. A
Category Document consists of one "app:categories" element that
contains a number of inline "atom:category" elements, or a URI
referencing a Category Document.

5.3. Transport Layer Security

   ROLIE is intended to be handled with Transport Layer Security (TLS).
   TLS version 1.2 MUST be supported. TLS 1.2 SHOULD be implemented
   according to all recommendations and best practices presented in
   [RFC7525].

   It is RECOMMENDED that the most recent published version of TLS be
   supported. If this version is TLS 1.3 [TLS-1.3], it is suggested
   that 0-RTT (Zero Round-Trip Time Resumption) not be used, in order to
   prevent replay attacks. Replay attacks on PUT, POST, or DELETE
   requests can disrupt repository operation by modifying data
   unexpectedly.

   For example, an automated ROLIE repository that updates very
   frequently may receive a PUT request against a given resource a few
   times an hour (or more). An attacker may store an early PUT request,
   and at the end of the resumption window replay the PUT request,
   reverting the resource to an old version. Not only could an attacker
   be doing this replay continuously to cause havoc on the server, but
   the client is completely unaware of the attack taking place.

   Given the potentially sensitive nature of data handled by ROLIE, all
   appropriate precautions should be taken at the transport layer to
   protect forward secrecy and user privacy.

   The server MUST implement certificate-based client authentication.
   This MAY be enabled on a workspace-by-workspace basis.

5.4. User Authentication and Authorization

   Implementations MUST support user authentication. However, a given
   implementation MAY allow user authentication to be disabled on a
   Feed-by-Feed or workspace-by-workspace basis.

   It is recommended that servers participating in an information-
   sharing consortium and supporting interactive user logins by members
   of the consortium support client authentication via a federated
   identity scheme.

   This document does not mandate the use of any specific user
   authorization mechanisms. However, service implementers SHOULD
   support appropriate authorization checking for all resource accesses,
   including individual Atom Entries, Atom Feeds, and Atom Service
   Documents.

5.5. "/" (Forward Slash) Resource URL

   The "/" resource MAY be supported for compatibility with existing
   deployments that are using [RFC6546] ("Transport of Real-time
   Inter-network Defense (RID) Messages over HTTP/TLS"). The following
   requirements apply only to implementations that support both RFC 6546
   and the "/" resource as described above:

   o Consistent with Erratum ID 3267 [Err3267] for [RFC6546], a client
      requesting a GET on the "/" resource SHOULD receive an HTTP status
      code 405 ("Method Not Allowed").

   o An implementation MAY provide full support for [RFC6546] such that
      a POST to the "/" resource containing a recognized RID message is
      handled correctly as a RID request. Alternatively, a client
      requesting a POST to "/" MAY receive an HTTP status code 307
      ("Temporary Redirect"). In this case, the location header in the
      HTTP response will provide the URL of the appropriate RID
      endpoint, and the client may repeat the POST method at the
      indicated location.

   If RFC 6546 is unsupported, then a request for the "/" resource may
   be handled as deemed appropriate by the server.

5.6. HTTP Methods

   Servers MAY accept request methods beyond those specified in this
   document.

   Clients MUST be capable of recognizing and processing any standard
   HTTP status code, as defined in Section 5 of [RFC5023].

6. ROLIE Requirements for the Atom Syndication Format

   This section describes a number of restrictions of, and extensions
   to, the Atom Syndication Format [RFC4287] that define the valid use
   of the format in the context of a ROLIE implementation. An
   understanding of the Atom Syndication Format specification [RFC4287]
   is helpful to understand the requirements in this section.

6.1. Use of the "atom:feed" Element

   As described in Section 4.1.1 of [RFC4287], an Atom Feed is an
   XML-based document format that describes a list of related
   information items. The Atom Feeds provided by a ROLIE service are
   listed in the service's Service Document through one or more
   "app:collection" elements. Each Feed Document, represented using the
   "atom:feed" element, contains a listing of zero or more Entries.

When applied to the problem domain of security automation information
sharing, an Atom Feed may be used to represent any meaningful
collection of security automation information resources. Each Entry
in a Feed represents an individual resource (e.g., a specific
checklist, a software vulnerability record). Additional Feeds can be
used to represent other collections of security automation resources.

As discussed in Section 5.1.2, ROLIE defines specialized data
requirements for Feeds containing data related to security
automation. The difference between a ROLIE Feed and a non-ROLIE Feed
can be determined as follows:

ROLIE Feed: For an "atom:feed" to be considered a ROLIE Feed, the
   "atom:feed" MUST contain only one child "atom:category" element
   with a "scheme" attribute value of
   "urn:ietf:params:rolie:category:information-type". This category
   MUST have an appropriate "term" attribute value as defined in
   Section 7.1.1. This ensures that a given Feed corresponds to a
   specific type of security automation information.

Non-ROLIE Feed: For an "atom:feed" to be considered a non-ROLIE
   Feed, the "atom:feed" MUST NOT contain an "atom:category" element
   with a "scheme" attribute value of
   "urn:ietf:params:rolie:category:information-type".

By distinguishing between ROLIE and non-ROLIE Feeds in this way,
implementations supporting ROLIE can host Feeds pertaining to
security automation information alongside Feeds of other non-ROLIE
information within the same AtomPub instance. This is parallel to
the handling of Collections as discussed earlier in this
specification (Section 5.1.2).

   The following Atom Feed definition represents a stricter definition
   of the "atom:feed" element defined in [RFC4287] when used as a
   ROLIE Feed. Any element not specified here inherits its definition
   and requirements from [RFC4287].

```
   atomFeed =
      element atom:feed {
         atomCommonAttributes,
         (atomAuthor*
          & atomCategory+
          & atomContributor*
          & atomGenerator?
          & atomIcon?
          & atomId
          & atomLink+
          & atomLogo?
          & atomRights?
          & atomSubtitle?
          & atomTitle
          & atomUpdated
          & extensionElement*),
         atomEntry*
      }
```

   The following subsections contain requirements for a ROLIE Feed.

6.1.1. Use of the "atom:category" Element

   An "atom:feed" can contain one or more "atom:category" elements. In
   Atom, the naming scheme and the semantic meaning of the terms used to
   identify an Atom category are application defined.

   The following are additional requirements on the use of the
   "atom:category" element when used in a ROLIE Feed:

   o All member Entries in the Feed MUST represent security automation
      information records of the provided information type category.

   o The "atom:feed" MAY include additional "atom:category" elements
      using a scheme other than
      "urn:ietf:params:rolie:category:information-type". This allows
      other category metadata to be included.

6.1.2. Use of the "atom:link" Element

   Link relations defined by the "atom:link" element are used to
   represent state transitions using a stateless approach. In Atom, a
   type of link relationship can be defined using the "rel" attribute.

   A ROLIE Feed MUST contain one or more "atom:link" elements with
   rel="service" and an "href" attribute whose value is a URI that
   points to an Atom Service Document associated with the Feed. If a
   client accesses a Feed without first accessing the service's Service
   Document, a link with the "service" relationship provides a means to
   discover additional security automation information. The "service"
   link relationship is defined in the IANA "Link Relations" registry at
   <https://www.iana.org/assignments/link-relations/>.

   A Feed can contain an arbitrary number of Entries. In some cases, a
   complete Feed may consist of a large number of Entries.
   Additionally, as new and updated Entries are ordered at the beginning
   of a Feed, a client may only be interested in retrieving the first N
   Entries in a Feed to process only the Entries that have changed since
   the last retrieval of the Feed. As a practical matter, a large set
   of Entries will likely need to be divided into more manageable
   portions, or pages. Based on Section 3 of [RFC5005], link elements
   SHOULD be included in all Feeds to support paging using the following
   link relation types:

   o "first" - Indicates that the "href" attribute value of the link
      identifies a resource URI for the furthest preceding page of
      the Feed.

   o "last" - Indicates that the "href" attribute value of the link
      identifies a resource URI for the furthest following page of
      the Feed.

   o "previous" - Indicates that the "href" attribute value of the link
      identifies a resource URI for the immediately preceding page of
      the Feed.

   o "next" - Indicates that the "href" attribute value of the link
      identifies a resource URI for the immediately following page of
      the Feed.

For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="https://www.w3.org/2005/Atom">
    <id>b7f65304-b63b-4246-88e2-c104049c5fd7</id>
    <title>Paged Feed</title>
    <link rel="self" href="https://example.org/feedA?page=5"/>
    <link rel="first" href="https://example.org/feedA?page=1"/>
    <link rel="prev" href="https://example.org/feedA?page=4"/>
    <link rel="next" href="https://example.org/feedA?page=6"/>
    <link rel="last" href="https://example.org/feedA?page=10"/>
    <updated>2012-05-04T18:13:51.0Z</updated>

    <!-- remainder of the Feed's elements -->
</feed>
```

                      Example Paged Feed

   A reference to a historical Feed may need to be stable, and/or a Feed
   may need to be divided into a series of defined epochs.
   Implementations SHOULD support the mechanisms described in Section 4
   of [RFC5005] to provide link-based state transitions for maintaining
   the archiving of Feeds.

   A Feed MAY include additional link relationships not specified in
   this document. If a client encounters an unknown link relationship
   type, the client MUST ignore the unrecognized link and continue
   processing as if the unrecognized link element did not appear. The
   definition of new link relations that provide additional state
   transition extensions is discussed in Section 7.3.

6.1.3. Use of the "atom:updated" Element

   The "atom:updated" element identifies the date and time that a Feed
   was last updated.

   The "atom:updated" element MUST be populated with the current time at
   the instant the Feed was last updated by adding, updating, or
   deleting an Entry, or by changing any metadata for the Feed.

6.2. Use of the "atom:entry" Element

   Each Entry in an Atom Feed, represented by the "atom:entry" element,
   describes a single referenced information record, along with
   descriptive information about its format, media type, and other
   publication metadata. The following "atom:entry" schema definition
   represents a stricter representation of the "atom:entry" element
   defined in [RFC4287] for use in a ROLIE-based Atom Feed as defined in
   Section 6.1.1.

```
   atomEntry =
     element atom:entry {
       atomCommonAttributes,
       (atomAuthor*
       & atomCategory*
       & atomContent
       & atomContributor*
       & atomId
       & atomLink*
       & atomPublished?
       & atomRights?
       & atomSource?
       & atomSummary?
       & atomTitle
       & atomUpdated
       & rolieFormat?
       & rolieProperty*
       & extensionElement*)
   }
```

   The notable changes from [RFC4287] are the addition of "rolieFormat"
   and "rolieProperty" elements. Also, the "atomContent" element is
   restricted to the atomOutOfLineContent formulation and is now
   REQUIRED.

   The following subsections contain requirements for Entries in a
   ROLIE Feed.

6.2.1. Use of the "atom:content" Element

   An "atom:content" element associates its containing Entry with a
   content resource identified by the "src" attribute.

   There MUST be exactly one "atom:content" element in the Entry. The
   "atom:content" element MUST adhere to this definition, which is a
   stricter representation of the "atom:content" element defined in
   [RFC4287]:

```
     atomContent =
       element atom:content {
         atomCommonAttributes,
         attribute type { atomMediaType },
         attribute src { atomUri },
         empty
     }
```

   This restricts atomContent in ROLIE to the atomOutOfLineContent
   formulation presented in [RFC4287].

   The "type" attribute MUST identify the serialization type of the
   content -- for example, "application/xml" or "application/json". A
   prefixed media type MAY be used to reflect a specific model used with
   a given serialization approach (e.g., "application/rdf+xml"). The
   "src" attribute MUST be a URI that can be dereferenced to retrieve
   the related content data.

6.2.2. Use of the "atom:link" Element

   Link relations can be included in an Entry to represent state
   transitions to and from the Entry, as well as to provide links to
   related information.

   If there is a need to provide the same information in different data
   models and/or serialization formats, separate Entry instances can be
   included in the same Feed or a different Feed. Such an alternate
   content representation can be indicated using an "atom:link" having a
   "rel" attribute with the value "alternate".

   A Feed MAY include additional link relationships not specified in
   this document. If a client encounters an unknown link relationship
   type, the client MUST ignore the unrecognized link and continue
   processing as if the unrecognized link element did not appear. The
   definition of new link relations that provide additional state
   transition extensions is discussed in Section 7.3.

6.2.3. Use of the "rolie:format" Element

   As mentioned in Sections 1 and 4, a key goal of this specification is
   to allow a consumer to review a set of published security automation
   information resources and then identify and retrieve any resources of
   interest. The format of the data is a key criteria to consider when
   deciding what information to retrieve. For a given type of security
   automation information, it is expected that a number of different
   formats may be used to represent this information. To support this
   use case, both the serialization format and the specific data model
   expressed in that format must be known by the consumer.

   In the Atom Syndication Format, a media type can be defined using the
   "type" attribute of the "atom:content" element of an "atom:entry".
   The media type can be fully descriptive of the format of the linked
   document, such as "application/atom+xml". In some cases, however, a
   format-specific media type may not be defined. An example might be
   when "application/xml" is used because there is no defined specific
   media type for the content. In such a case, the exact data model of
   the content cannot be known without first retrieving the content.

   In cases where a specific media type does not exist, the
   "rolie:format" element is used to describe the data model used to
   express the information referenced in the "atom:content" element.
   The "rolie:format" element also allows a schema to be identified that
   can be used when parsing the content to verify or better understand
   the structure of the content.

   When it appears, the "rolie:format" element MUST adhere to this
   definition:

```
     rolieFormat =
       element rolie:format {
         atomCommonAttributes,
         attribute ns { atomUri },
         attribute version { text } ?,
         attribute schema-location { atomUri } ?,
         attribute schema-type { atomMediaType } ?,
         empty
     }
```

   The "rolie:format" element MUST provide a "ns" attribute that
   identifies the data model of the resource referenced by the
   "atom:content" element. For example, the namespace used may be an
   XML namespace URI or an identifier that represents a serialized JSON
   model. The URI used for the "ns" attribute MUST be absolute. The
   resource identified by the URI need not be resolvable.

   The "rolie:format" element MAY provide a "version" attribute that
   identifies the version of the format used for the related
   "atom:content" element.

   The "rolie:format" element MAY provide a "schema-location" attribute,
   which is a URI that identifies a schema resource that can be used to
   validate the related "atom:content" element.

   The "rolie:format" element MAY provide a "schema-type" attribute,
   which is a media type (as described in [RFC2045]) identifying the
   format of the schema resource identified by the "schema-location"
   attribute.

   The following nominal example shows how these attributes describe the
   format of the content:

```
<rolie:format ns="urn:ietf:params:xml:ns:iodef-2.0"
  version="2.0"
  schema-location=
    "https://www.iana.org/assignments/xml-registry/schema/iodef-2.0.xsd"
  schema-type="text/xml"/>
```

   The previous element provides an indication that the content of the
   given Entry is using the Incident Object Description Exchange Format
   (IODEF) v2 format.

6.2.4. Use of the "rolie:property" Element

   An "atom:category" element provides a way to associate a name/value
   pair of categorical information using the "scheme" and "term"
   attributes to represent the name and using the "label" attribute to
   represent the value. When used in this way, an "atom:category"
   allows a specific label to be selected from a finite set of possible
   label values that can be used to further classify a given Entry or
   Feed. Within ROLIE, there may be a need to associate additional
   metadata with an Entry. In such a case, the use of an
   "atom:category" is not practical to represent name/value data for
   which the allowed values are unbounded. Instead, ROLIE introduces a
   new "rolie:property" element that can represent non-categorical
   metadata as name/value pairs. Examples include content-specific
   identifiers, naming data, and other properties that allow for
   unbounded values.

   There MAY be zero or more "rolie:property" elements in an
   "atom:entry".

The element MUST adhere to this definition:

```
rolieProperty =
  element rolie:property {
    atomCommonAttributes,
    attribute name { atomUri },
    attribute value { text },
    empty
}
```

The "name" attribute provides a URI that identifies the namespace and
name of the property as a URI.

The "value" attribute is text that provides a value for the property
identified by the "name" attribute.

For example, the nominal element <rolie:property
name="urn:ietf:params:rolie:property:content-id" value="12345"/>
would expose an IODEF ID value contained in a given Entry's content.
The name used in the example also demonstrates the use of a
registered ROLIE property extension, which is described in
Section 7.4.

Implementations MAY use locally defined and namespaced elements in an
Entry in order to provide additional information. Clients that do
not recognize a property with an unregistered "name" attribute MUST
ignore the "rolie:property" element; that is, the client MUST NOT
fail parsing content that contains an unrecognized property.

6.2.5. Requirements for a Standalone Entry

   If an Entry is ever shared as a standalone resource, separate from
   its containing Feed, then the following additional requirements
   apply:

   o The Entry MUST have an "atom:link" element with rel="collection"
      and href="[URI of the containing Collection]". This allows the
      Feed or Feeds of which the Entry is a member to be discovered,
      along with the related information the Feed may contain. In the
      case where the Entry has multiple containing Feeds, the Entry MUST
      have one "atom:link" for each related Feed.

   o The Entry MUST declare the information type of the content
      resource referenced by the Entry (see Section 7.1.2).

7. Available Extension Points Provided by ROLIE

   This specification does not require particular information types or
   data formats; rather, ROLIE is intended to be extended by additional
   specifications that define the use of new categories and link
   relations. The primary point of extension is through the definition
   of new information type category terms. Additional specifications
   can register new information type category terms with IANA that serve
   as the main characterizing feature of a ROLIE Collection/Feed or
   resource/Entry. These additional specifications defining new
   information type terms can describe additional requirements for
   including specific categories and link relations, as well as the use
   of specific data formats supporting a given information type term.

7.1. The Category Extension Point

   The "atom:category" element, defined in Section 4.2.2 of [RFC4287],
   provides a mechanism to provide additional categorization information
   for a content resource in ROLIE. The ability to define new
   categories is one of the core extension points provided by Atom. A
   Category Document, defined in Section 7 of [RFC5023], provides a
   mechanism for an Atom implementation to make discoverable the
   "atom:category" terms and associated allowed values.

   ROLIE further defines the use of the existing Atom extension category
   mechanism by allowing ROLIE-specific category extensions to be
   registered with IANA. The
   "urn:ietf:params:rolie:category:information-type" category scheme,
   which has special meaning for implementations of ROLIE, has been
   assigned (see Section 8.3). This allows category scheme namespaces
   to be managed in a more consistent way, allowing for greater
   interoperability between content producers and consumers.

   Any "atom:category" element whose "scheme" attribute uses an
   unregistered scheme MUST be considered "Private Use" as defined in
   [RFC8126]. Implementations encountering such a category MUST parse
   the content without error but MAY otherwise ignore the element.

   The use of the "atom:category" element is discussed in the following
   subsections.

7.1.1. General Use of the "atom:category" Element

   The "atom:category" element can be used for characterizing a ROLIE
   resource. An "atom:category" element has a "term" attribute that
   indicates the assigned category value and a "scheme" attribute that
   provides an identifier for the category type. The "scheme" provides
   a means to describe how a set of category terms should be used and
   provides a namespace that can be used to differentiate terms that are
   provided by multiple organizations and that have different semantic
   meaning.

   To further differentiate category types used in ROLIE, an IANA
   subregistry has been established for ROLIE protocol parameters to
   support the registration of new category "scheme" attribute values by
   ROLIE extension specifications. The use of this extension point is
   discussed in Section 8.3, using the "name" field with a type
   parameter of "category" to indicate a category extension.

7.1.2. Identification of Security Automation Information Types

   A ROLIE-specific extension point is provided through the
   "atom:category" element's "scheme" attribute value
   "urn:ietf:params:rolie:category:information-type". This value is a
   Uniform Resource Name (URN) [RFC8141] that is registered with IANA as
   described in Section 8.3. When used as the "scheme" attribute in
   this way, the "term" attribute is expected to be a registered value
   as defined in Section 8.4. Through this mechanism, a given security
   automation information type can be used to:

   1. identify that an "app:collection" element in a Service Document
       points to an Atom Feed that contains Entries pertaining to a
       specific type of security automation information (see
       Section 5.1.2),

   2. identify that an "atom:feed" element in an Atom Feed contains
       Entries pertaining to a specific type of security automation
       information (see Section 6.1.1), or

   3. identify the information type of a standalone resource (see
       Section 6.2.5).

For example, the notional security automation information type
"incident" would be identified as follows:

```
<atom:category
    scheme="urn:ietf:params:rolie:category:information-type"
    term="incident"/>
```

A security automation information type represents a class of
information that represents the same or similar information model
[RFC3444]. Note that this document does not register any information
types but offers the following as examples of potential information
types:

indicator: Computing device- or network-related "observable features
   and phenomenon that aid in the forensic or proactive detection of
   malicious activity and associated metadata" (from [RFC7970]).

incident: Information pertaining to or derived from security
   incidents.

vulnerability reports: Information identifying and describing a
   vulnerability in hardware or software.

configuration checklists: Content that can be used to assess the
   configuration settings related to installed software.

software tags: Metadata used to identify and characterize
   installable software.

This is a short list to inspire new engineering of information type
extensions that support the automation of security processes.

This document does not specify any information types. Instead,
information types in ROLIE are expected to be registered in extension
documents that describe one or more new information types. This
allows the information types used by ROLIE implementations to grow
over time to support new security automation use cases. These
extension documents may also enhance ROLIE Service, Category, Feed,
and Entry Documents by defining link relations, other categories, and
Format data model extensions to address the representational needs of
these specific information types. New information types are added to
ROLIE through registrations to the IANA "ROLIE Information Types"
registry defined in Section 8.4.

7.2. The "rolie:format" Extension Point

   Security automation data pertaining to a given information type may
   be expressed using a number of supported formats. As described in
   Section 6.2.3, the "rolie:format" element is used to describe the
   specific data model used to represent the resource referenced by a
   given "atom:entry". The structure provided by the "rolie:format"
   element provides a mechanism for extension within the "atom:entry"
   model. ROLIE extensions MAY further restrict which data models are
   allowed to be used for a given information type.

   By declaring the data model used for a given resource, a consumer can
   choose to download or ignore the resource, or look for alternate
   formats. This saves the consumer from downloading and parsing
   resources that the consumer is not interested in or resources
   expressed in formats that are not supported by the consumer.

7.3. The Link Relation Extension Point

   This document uses several link relations defined in the IANA
   "Link Relation Types" registry at
   <https://www.iana.org/assignments/link-relations/>. Additional link
   relations can be registered in this registry to allow new
   relationships to be represented in ROLIE according to Section 4.2.7.2
   of [RFC4287]. Based on the preceding reference, if the link relation
   is too specific or limited in its intended use, an absolute URI can
   be used in lieu of registering a new simple name with IANA.

7.4. The "rolie:property" Extension Point

   As discussed previously in Section 6.2.3, many formats contain unique
   identifying and characterizing properties that are vital for sharing
   information. In order to provide a global reference for these
   properties, this document establishes an IANA registry that allows
   ROLIE extensions to register named properties using the "name" field
   with a type parameter of "property" to indicate a property extension;
   see Section 8.3. Implementations SHOULD prefer the use of registered
   properties over implementation-specific properties when possible.

   ROLIE extensions are expected to register new properties and use
   existing properties to provide valuable identifying and
   characterizing information for a given information type and/or
   format.

Any "rolie:property" element whose "name" attribute has
"urn:ietf:params:rolie:property:local" as a prefix MUST be considered
"Private Use" as defined in [RFC8126]. Implementations encountering
such a property MUST parse the content without error but MAY
otherwise ignore the element.

This document also registers a number of general-use properties that
can be used to expose content information in any ROLIE use case. The
following are descriptions of how to use these registered properties:

urn:ietf:params:rolie:property:content-author-name
   The "value" attribute of this property is a text representation
   indicating the individual or organization that authored the
   content referenced by the "src" attribute of the Entry's
   "atom:content" element. This author may differ from the
   "atom:author" element when the author of the content and the
   author of the Entry are different people or entities.

urn:ietf:params:rolie:property:content-id
   The "value" attribute of this property is a text representation of
   an identifier pertaining to or extracted from the content
   referenced by the "src" attribute of the Entry's "atom:content"
   element. For example, if the "atom:entry"'s "atom:content"
   element links to an IODEF document, the "content-id" value would
   be an identifier of that IODEF document.

urn:ietf:params:rolie:property:content-published-date
   The "value" attribute of this property is a text representation
   indicating the original publication date of the content referenced
   by the "src" attribute of the Entry's "atom:content" element.
   This date may differ from the published date of the ROLIE Entry
   because publication of the content and publication of the ROLIE
   Entry represent different events. The date MUST be formatted as
   specified in [RFC3339].

urn:ietf:params:rolie:property:content-updated-date
   The "value" attribute of this property is a text representation
   indicating the date that the content, referenced by the "src"
   attribute of the Entry's "atom:content" element, was last updated.
   This date may differ from the updated date of the ROLIE Entry
   because updates made to the content and to the ROLIE Entry are
   different events. The date MUST be formatted as specified in
   [RFC3339].

8. IANA Considerations

   This document has a number of IANA considerations, as described in
   the following subsections.

8.1. XML Namespaces and Schema URNs

   This document uses URNs to describe XML namespaces and XML schemas
   conforming to the registry mechanism described in [RFC3688].

   ROLIE XML Namespace: The ROLIE namespace (rolie-1.0) has been
      registered in the "ns" registry.

      URI: urn:ietf:params:xml:ns:rolie-1.0

      Registrant Contact: IESG

      XML: None. Namespace URIs do not represent an XML specification.

   ROLIE XML Schema: The ROLIE schema (rolie-1.0) has been registered
      in the "schema" registry.

      URI: urn:ietf:params:xml:schema:rolie-1.0

      Registrant Contact: IESG

      XML: See Appendix A of this document.

8.2. ROLIE URN Sub-namespace

   IANA has added an entry to the "IETF URN Sub-namespace for Registered
   Protocol Parameter Identifiers" registry located at
   <https://www.iana.org/assignments/params/> as per [RFC3553].

   The entry is as follows:

      Registered Parameter Identifier: rolie

      Specification: This document

      Repository: ROLIE URN Parameters. See Section 8.3.

      Index value: See Section 8.4.

8.3. ROLIE URN Parameters

   A new top-level registry has been created, titled "Resource-Oriented
   Lightweight Information Exchange (ROLIE) URN Parameters".

   Registration in the "ROLIE URN Parameters" subregistry is via the
   Specification Required policy [RFC8126]. Registration requests must
   be sent to both the MILE Working Group mailing list (mile@ietf.org)
   and IANA. IANA will forward registration requests to the Designated
   Expert.

   Each entry in this subregistry must record the following fields:

   Name: A URN segment that adheres to the pattern {type}:{label}. The
      keywords are defined as follows:

      {type}:   The parameter type. The allowed values are "category"
                or "property". "category" denotes a category extension
                as discussed in Section 7.1. "property" denotes a
                property extension as discussed in Section 7.4.

      {label}: A required US-ASCII string that conforms to the URN
                syntax requirements (see [RFC8141]). This string must
                be unique within the namespace defined by the {type}
                keyword. The "local" label for both the "category" and
                "property" types has been reserved for private use.

   Extension URI: The identifier to use within ROLIE, which is the full
      URN using the form "urn:ietf:params:rolie:{name}", where {name} is
      the "name" field of this registration.

   Reference: A static link to the specification and section where the
      definition of the parameter can be found.

   Subregistry: An optional field that links to an IANA subregistry for
      this parameter. If the {type} is "category", the subregistry must
      contain a "name" field whose registered values MUST be US-ASCII.
      The list of names are the allowed values of the "term" attribute
      in the "atom:category" element (see Section 7.1.2).

This repository has the following initial values:

Name	Extension URI	Reference (This Document)	Subregistry
category: information-type	urn:ietf:params:rolie: category: information-type	Section 8.4	See Section 8.4
property: content-author-name	urn:ietf:params:rolie: property:content-author-name	Section 7.4	None
property: content-id	urn:ietf:params:rolie: property:content-id	Section 7.4	None
property: content-published-date	urn:ietf:params:rolie: property:content-published-date	Section 7.4	None
property: content-updated-date	urn:ietf:params:rolie: property:content-updated-date	Section 7.4	None

8.4. ROLIE Information Types Registry

   A new subregistry has been created to store ROLIE information type
   values.

      Name of Registry: "ROLIE Information Types"

      Location of Registry:
         <https://www.iana.org/assignments/rolie/>

      Fields to record in the registry:

         Name: The full name of the security resource information type
         as a string from the printable ASCII character set [RFC20] with
         individual embedded spaces allowed. This value must be unique
         in the context of this table. The ABNF [RFC5234] syntax for
         this field is:

            1*VCHAR *(SP 1*VCHAR)

         Index: An IANA-assigned positive integer that identifies the
         registration. The first entry added to this registry uses the
         value 1, and this value is incremented for each subsequent
         entry added to the registry.

         Reference: A list of one or more URIs [RFC3986] from which the
         registered specification can be obtained. The registered
         specification MUST be readily and publicly available from that
         URI. The URI SHOULD be a stable reference.

      Allocation Policy: Specification Required, as per [RFC8126]

9. Security Considerations

   This document defines a resource-oriented approach for lightweight
   information exchange using HTTP over TLS, the Atom Syndication
   Format, and AtomPub. As such, implementers must understand the
   security considerations described in those specifications. All that
   follows is guidance; instructions that are more specific are out of
   scope for this document.

   To protect the confidentiality of a given resource provided by a
   ROLIE implementation, requests for retrieval of the resource need to
   be authenticated to prevent unauthorized users from accessing the
   resource (see Section 5.4). It can also be useful to log and audit
   access to sensitive resources to verify that proper access controls
   remain in place over time.

Access control to information published using ROLIE should use
mechanisms that are appropriate to the sensitivity of the
information. Primitive authentication mechanisms like HTTP Basic
Authentication [RFC7617] are rarely appropriate for sensitive
information. A number of authentication schemes are defined in the
"HTTP Authentication Schemes" registry at
<https://www.iana.org/assignments/http-authschemes/>. Of these, HTTP
Origin-Bound Authentication (HOBA) [RFC7486] and SCRAM-SHA-256
[RFC7804] ("SCRAM" stands for "Salted Challenge Response
Authentication Mechanism") provide improved security properties over
HTTP Basic [RFC7617]and Digest [RFC7616] authentication schemes.
However, sharing communities that are engaged in sensitive
collaborative analysis and/or operational response for indicators and
incidents targeting high-value information systems should adopt a
suitably stronger user authentication solution, such as a risk-based
or multi-factor approach.

Collaborating consortiums may benefit from the adoption of a
federated identity solution, such as those based upon OAuth [RFC6749]
with the JSON Web Token (JWT) [RFC7797], or SAML-core [SAML-core]
("SAML" stands for "Security Assertion Markup Language"), SAML-bind
[SAML-bind], and SAML-prof [SAML-prof] for web-based authentication
and cross-organizational single sign-on. Dependency on a trusted
third-party identity provider implies that appropriate care must be
exercised to sufficiently secure the identity provider. Any attacks
on the federated identity system would present a risk to the
consortium, as a relying party. Potential mitigations include
deployment of a federation-aware identity provider that is under the
control of the information-sharing consortium, with suitably
stringent technical and management controls.

Authorization of resource representations is the responsibility of
the source system, i.e., based on the authenticated user identity
associated with an HTTP(S) request. The required authorization
policies that are to be enforced must therefore be managed by the
security administrators of the source system. Various authorization
architectures would be suitable for this purpose, such as Role-Based
Access Control (RBAC) <https://csrc.nist.gov/projects/
role-based-access-control> and/or Attribute-Based Access Control
(ABAC), as embodied in the eXtensible Access Control Markup Language
(XACML) [XACML]. In particular, implementers adopting XACML may
benefit from the capability to represent their authorization policies
in a standardized, interoperable format. Note that implementers are
free to choose any suitable authorization mechanism that is capable
of fulfilling the policy enforcement requirements relevant to their
consortium and/or organization.

   Additional security requirements such as enforcing message-level
   security at the destination system could supplement the security
   enforcements performed at the source system; however, these
   destination-provided policy enforcements are out of scope for this
   specification. Implementers requiring this capability should
   consider leveraging, for example, the <RIDPolicy> element in the RID
   schema. Refer to Section 9 of [RFC6545] for more information.
   Additionally, the underlying serialization approach used in the
   representation (e.g., XML, JSON) can offer encryption and message
   authentication capabilities. For example, XML Digital Signatures
   (XMLDSIG) [RFC3275] for XML, as well as JSON Web Encryption [RFC7516]
   and JSON Web Signature [RFC7515] for JSON, can provide such
   mechanisms.

   When security policies relevant to the source system are to be
   enforced at both the source and destination systems, implementers
   must take care to avoid unintended interactions of the separately
   enforced policies. Potential risks will include unintended denial of
   service and/or unintended information leakage. These problems may be
   mitigated by avoiding any dependence upon enforcements performed at
   the destination system. When distributed enforcement is unavoidable,
   the usage of a standard language (e.g., XACML) for the expression of
   authorization policies will enable the source and destination systems
   to better coordinate and align their respective policy expressions.

   A service discovery mechanism is not explicitly specified in this
   document, but there are several approaches available for
   implementers. When selecting this mechanism, implementations need to
   ensure that their choice provides a means for authenticating the
   server. DNS SRV records [RFC2782] are a possible solution to the
   discovery problem described in Section 5.1.3.

10. Privacy Considerations

   The optional "author" field may provide an identification privacy
   issue if populated without the author's consent. This information
   may become public if posted to a public Feed. When aggregating or
   sharing Entries from other Feeds or when programmatically generating
   ROLIE Entries from some data source, special care should be taken to
   ensure that the author's personal information is not shared without
   the author's consent.

   When using AtomPub to POST Entries to a Feed, attackers may use
   correlating techniques to profile the user. The request time can be
   compared to the generated "updated" field of the Entry in order to
   build out information about a given user. This correlation attempt
   can be mitigated by not using HTTP requests to POST Entries when
   profiling is a risk and instead using backend control of the Feeds.

   Adoption of the information-sharing approach described in this
   document will enable users to more easily perform correlations across
   separate, and potentially unrelated, cybersecurity information
   providers. A client may succeed in assembling a data set that would
   not have been permitted within the context of the authorization
   policies of either provider when considered individually. Thus,
   providers may face a risk of an attacker obtaining an access that
   constitutes an undetected separation of duties (SOD) violation. It
   is important to note that this risk is not unique to this
   specification, and a similar potential for abuse exists with any
   other cybersecurity information-sharing protocol. However, the wide
   availability of tools for HTTP clients and Atom Feed handling implies
   that the resources and technical skills required for a successful
   exploit may be less than it was previously. This risk can be best
   mitigated through appropriate vetting of the client at the time of
   account provisioning. In addition, any increase in the risk of this
   type of abuse should be offset by the corresponding increase in
   effectiveness that this specification affords to the defenders.

   Overall, privacy concerns in ROLIE can be mitigated by following
   security considerations and by the careful use of the optional
   personally identifying elements (e.g., author) provided by Atom
   Syndication and ROLIE.

11. References

11.1. Normative References

   [RELAX-NG] Clark, J., Ed., "RELAX NG Compact Syntax", November 2002,
              <https://www.oasis-open.org/committees/relax-ng/
              compact-20021121.html>.

   [RFC20]   Cerf, V., "ASCII format for network interchange", STD 80,
              RFC 20, DOI 10.17487/RFC0020, October 1969,
              <https://www.rfc-editor.org/info/rfc20>.

   [RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail
              Extensions (MIME) Part One: Format of Internet Message
              Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996,
              <https://www.rfc-editor.org/info/rfc2045>.

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119,
              DOI 10.17487/RFC2119, March 1997,
              <https://www.rfc-editor.org/info/rfc2119>.

   [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet:
              Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002,
              <https://www.rfc-editor.org/info/rfc3339>.

   [RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An
              IETF URN Sub-namespace for Registered Protocol
              Parameters", BCP 73, RFC 3553, DOI 10.17487/RFC3553,
              June 2003, <https://www.rfc-editor.org/info/rfc3553>.

   [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688,
              DOI 10.17487/RFC3688, January 2004,
              <https://www.rfc-editor.org/info/rfc3688>.

   [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
              Resource Identifier (URI): Generic Syntax", STD 66,
              RFC 3986, DOI 10.17487/RFC3986, January 2005,
              <https://www.rfc-editor.org/info/rfc3986>.

   [RFC4287] Nottingham, M., Ed., and R. Sayre, Ed., "The Atom
              Syndication Format", RFC 4287, DOI 10.17487/RFC4287,
              December 2005, <https://www.rfc-editor.org/info/rfc4287>.

   [RFC5005] Nottingham, M., "Feed Paging and Archiving", RFC 5005,
              DOI 10.17487/RFC5005, September 2007,
              <https://www.rfc-editor.org/info/rfc5005>.

   [RFC5023] Gregorio, J., Ed., and B. de hOra, Ed., "The Atom
              Publishing Protocol", RFC 5023, DOI 10.17487/RFC5023,
              October 2007, <https://www.rfc-editor.org/info/rfc5023>.

   [RFC5234] Crocker, D., Ed., and P. Overell, "Augmented BNF for
              Syntax Specifications: ABNF", STD 68, RFC 5234,
              DOI 10.17487/RFC5234, January 2008,
              <https://www.rfc-editor.org/info/rfc5234>.

   [RFC6546] Trammell, B., "Transport of Real-time Inter-network
              Defense (RID) Messages over HTTP/TLS", RFC 6546,
              DOI 10.17487/RFC6546, April 2012,
              <https://www.rfc-editor.org/info/rfc6546>.

   [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre,
              "Recommendations for Secure Use of Transport Layer
              Security (TLS) and Datagram Transport Layer Security
              (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525,
              May 2015, <https://www.rfc-editor.org/info/rfc7525>.

   [RFC7970] Danyliw, R., "The Incident Object Description Exchange
              Format Version 2", RFC 7970, DOI 10.17487/RFC7970,
              November 2016, <https://www.rfc-editor.org/info/rfc7970>.

   [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for
              Writing an IANA Considerations Section in RFCs", BCP 26,
              RFC 8126, DOI 10.17487/RFC8126, June 2017,
              <https://www.rfc-editor.org/info/rfc8126>.

   [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in
              RFC 2119 Key Words", BCP 14, RFC 8174,
              DOI 10.17487/RFC8174, May 2017,
              <https://www.rfc-editor.org/info/rfc8174>.

   [W3C.REC-xml-names-20091208]
              Bray, T., Hollander, D., Layman, A., Tobin, R., and H.
              Thompson, "Namespaces in XML 1.0 (Third Edition)", World
              Wide Web Consortium Recommendation REC-xml-names-20091208,
              December 2009, <https://www.w3.org/TR/2009/
              REC-xml-names-20091208>.

11.2. Informative References

   [Err3267] RFC Errata, Erratum ID 3267, RFC 6546,
              <https://www.rfc-editor.org/errata/eid3267>.

   [REST]   Fielding, R., "Architectural Styles and the Design of
              Network-based Software Architectures", 2000,
              <http://www.ics.uci.edu/~fielding/pubs/
              dissertation/top.htm>.

   [RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for
              specifying the location of services (DNS SRV)", RFC 2782,
              DOI 10.17487/RFC2782, February 2000,
              <https://www.rfc-editor.org/info/rfc2782>.

   [RFC3275] Eastlake 3rd, D., Reagle, J., and D. Solo, "(Extensible
              Markup Language) XML-Signature Syntax and Processing",
              RFC 3275, DOI 10.17487/RFC3275, March 2002,
              <https://www.rfc-editor.org/info/rfc3275>.

   [RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between
              Information Models and Data Models", RFC 3444,
              DOI 10.17487/RFC3444, January 2003,
              <https://www.rfc-editor.org/info/rfc3444>.

   [RFC6545] Moriarty, K., "Real-time Inter-network Defense (RID)",
              RFC 6545, DOI 10.17487/RFC6545, April 2012,
              <https://www.rfc-editor.org/info/rfc6545>.

   [RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework",
              RFC 6749, DOI 10.17487/RFC6749, October 2012,
              <https://www.rfc-editor.org/info/rfc6749>.

   [RFC7486] Farrell, S., Hoffman, P., and M. Thomas, "HTTP
              Origin-Bound Authentication (HOBA)", RFC 7486,
              DOI 10.17487/RFC7486, March 2015,
              <https://www.rfc-editor.org/info/rfc7486>.

   [RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web
              Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515,
              May 2015, <https://www.rfc-editor.org/info/rfc7515>.

   [RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)",
              RFC 7516, DOI 10.17487/RFC7516, May 2015,
              <https://www.rfc-editor.org/info/rfc7516>.

   [RFC7616] Shekh-Yusef, R., Ed., Ahrens, D., and S. Bremer, "HTTP
              Digest Access Authentication", RFC 7616,
              DOI 10.17487/RFC7616, September 2015,
              <https://www.rfc-editor.org/info/rfc7616>.

   [RFC7617] Reschke, J., "The 'Basic' HTTP Authentication Scheme",
              RFC 7617, DOI 10.17487/RFC7617, September 2015,
              <https://www.rfc-editor.org/info/rfc7617>.

   [RFC7797] Jones, M., "JSON Web Signature (JWS) Unencoded Payload
              Option", RFC 7797, DOI 10.17487/RFC7797, February 2016,
              <https://www.rfc-editor.org/info/rfc7797>.

   [RFC7804] Melnikov, A., "Salted Challenge Response HTTP
              Authentication Mechanism", RFC 7804, DOI 10.17487/RFC7804,
              March 2016, <https://www.rfc-editor.org/info/rfc7804>.

   [RFC8141] Saint-Andre, P. and J. Klensin, "Uniform Resource Names
              (URNs)", RFC 8141, DOI 10.17487/RFC8141, April 2017,
              <https://www.rfc-editor.org/info/rfc8141>.

   [SAML-bind]
              Cantor, S., Hirsch, F., Kemp, J., Philpott, R., and E.
              Maler, "Bindings for the OASIS Security Assertion Markup
              Language (SAML) V2.0", OASIS Standard saml-bindings-
              2.0-os, March 2005, <http://docs.oasis-open.org/
              security/saml/v2.0/saml-bindings-2.0-os.pdf>.

   [SAML-core]
              Cantor, S., Kemp, J., Philpott, R., and E. Maler,
              "Assertions and Protocols for the OASIS Security Assertion
              Markup Language (SAML) V2.0", OASIS Standard saml-core-
              2.0-os, March 2005, <http://docs.oasis-open.org/
              security/saml/v2.0/saml-core-2.0-os.pdf>.

   [SAML-prof]
              Hughes, J., Cantor, S., Hodges, J., Hirsch, F., Mishra,
              P., Philpott, R., and E. Maler, "Profiles for the OASIS
              Security Assertion Markup Language (SAML) V2.0", OASIS
              Standard saml-profiles-2.0-os, March 2005,
              <http://docs.oasis-open.org/security/saml/v2.0/
              saml-profiles-2.0-os.pdf>.

   [TLS-1.3] Rescorla, E., "The Transport Layer Security (TLS) Protocol
              Version 1.3", Work in Progress, draft-ietf-tls-tls13-23,
              January 2018.

   [XACML]   Rissanen, E., "eXtensible Access Control Markup Language
              (XACML) Version 3.0 Plus Errata 01", July 2017,
              <http://docs.oasis-open.org/xacml/3.0/
              xacml-3.0-core-spec-en.pdf>.

Appendix A. RELAX NG Compact Schema for ROLIE

   This appendix is informative.

   The RELAX NG schema below defines the "rolie:format" element.

```
   # -*- rnc -*-
   # RELAX NG Compact Syntax Grammar for the rolie ns

   namespace rolie = "urn:ietf:params:xml:ns:rolie-1.0"

   # import the ATOM Syndication RELAX NG Compact Syntax Grammar
   include "atomsynd.rnc"

   # rolie:format
   rolieFormat =
     element rolie:format {
       atomCommonAttributes,
       attribute ns { atomUri },
       attribute version { text } ?,
       attribute schema-location { atomUri } ?,
       attribute schema-type { atomMediaType } ?,
       empty
     }

   # rolie:property

   rolieProperty =
     element rolie:property {
       atomCommonAttributes,
       attribute name { atomUri },
       attribute value { text },
       empty
     }
```

Appendix B. Examples of Use

B.1. Service Discovery

   This appendix provides a non-normative example of a client doing
   service discovery.

   An Atom Service Document enables a client to dynamically discover
   what Feeds a particular publisher makes available. Thus, a provider
   uses an Atom Service Document to enable authorized clients to
   determine what specific information the provider makes available to

the community. The Service Document should be made accessible from
an easily found location, such as a link from the producer's
home page.

A client may format an HTTP GET request to retrieve the Service
Document from the specified location:

```
GET /rolie/servicedocument
Host: www.example.org
Accept: application/atomsvc+xml
```

Notice the use of the HTTP Accept: request header, indicating the
MIME type for Atom service discovery. The response to this GET
request will be an XML document that contains information on the
specific Collections that are provided.

Example HTTP GET response:

```
HTTP/1.1 200 OK
Date: Fri, 24 Aug 2016 17:09:11 GMT
Content-Length: 570
Content-Type: application/atomsvc+xml;charset="utf-8"

<?xml version="1.0" encoding="UTF-8"?>
<service xmlns="https://www.w3.org/2007/app"
    xmlns:atom="https://www.w3.org/2005/Atom">
  <workspace>
    <atom:title type="text">Vulnerabilities</atom:title>
    <collection href="https://example.org/provider/vulns">
      <atom:title type="text">Vulnerabilities Feed</atom:title>
      <categories fixed="yes">
        <atom:category
            scheme="urn:ietf:params:rolie:category:information-type"
            term="vulnerability"/>
      </categories>
    </collection>
  </workspace>
</service>
```

This simple Service Document example shows that the server provides
one workspace, named "Vulnerabilities". Within that workspace, the
server makes one Collection available.

A server may also offer a number of different Collections, each
containing different types of security automation information. In
the following example, a number of different Collections are
provided, each with its own category and authorization scope. This
categorization will help the clients to decide which Collections will
meet their needs.

```
HTTP/1.1 200 OK
Date: Fri, 24 Aug 2016 17:10:11 GMT
Content-Length: 1912
Content-Type: application/atomsvc+xml;charset="utf-8"

<?xml version="1.0" encoding='utf-8'?>
<service xmlns="https://www.w3.org/2007/app"
    xmlns:atom="https://www.w3.org/2005/Atom">
  <workspace>
    <atom:title>Public Security Information Sharing</atom:title>
    <collection
        href="https://example.org/provider/public/vulns">
      <atom:title>Public Vulnerabilities</atom:title>
      <atom:link rel="service"
        href="https://example.org/rolie/servicedocument"/>
      <categories fixed="yes">
        <atom:category
            scheme="urn:ietf:params:rolie:category:information-type"
            term="vulnerability"/>
      </categories>
    </collection>
  </workspace>
  <workspace>
    <atom:title>Private Consortium Sharing</atom:title>
    <collection
        href="https://example.org/provider/private/incidents">
      <atom:title>Incidents</atom:title>
      <atom:link rel="service"
        href="https://example.org/rolie/servicedocument"/>
      <categories fixed="yes">
        <atom:category
            scheme="urn:ietf:params:rolie:category:information-type"
            term="incident"/>
      </categories>
    </collection>
  </workspace>
</service>
```

In this example, the provider is making available a total of two
Collections, organized into two different workspaces. The first
workspace contains a Collection consisting of publicly available
software vulnerabilities. The second workspace provides an incident
Collection for use by a private sharing consortium. An appropriately
authenticated and authorized client may then proceed to make HTTP
requests for these Collections. The publicly provided vulnerability
information may be accessible with or without authentication.
However, users accessing the Collection restricted to authorized
members of a private sharing consortium are expected to authenticate
before access is allowed.

B.2. Feed Retrieval

This appendix provides a non-normative example of a client retrieving
a vulnerability Feed.

Having discovered the available Collections that share security
information, a client who is a member of the general public may be
interested in receiving the Collection of public vulnerabilities,
expressed as Common Vulnerabilities and Exposures (CVEs). The client
may retrieve the Feed for this Collection by performing an HTTP GET
operation on the URL indicated by the Collection's "href" attribute.

Example HTTP GET request for a Feed:

```
GET /provider/public/vulns
Host: www.example.org
Accept: application/atom+xml
```

The corresponding HTTP response would be an XML document containing
the vulnerability Feed:

Example HTTP GET response for a Feed:

```
HTTP/1.1 200 OK
Date: Fri, 24 Aug 2016 17:20:11 GMT
Content-Length: 2882
Content-Type: application/atom+xml;charset="utf-8"

<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="https://www.w3.org/2005/Atom"
    xmlns:rolie="urn:ietf:params:xml:ns:rolie-1.0"
    xml:lang="en-US">
  <id>2a7e265a-39bc-43f2-b711-b8fd9264b5c9</id>
  <title type="text">
      Atom-formatted representation of
      a Feed of XML vulnerability documents
  </title>
  <category
      scheme="urn:ietf:params:rolie:category:information-type"
      term="vulnerability"/>
  <updated>2016-05-04T18:13:51.0Z</updated>
  <link rel="self"
      href="https://example.org/provider/public/vulns"/>
  <link rel="service"
      href="https://example.org/rolie/servicedocument"/>
  <entry>
    <rolie:format ns="urn:ietf:params:xml:ns:exampleformat"/>
    <id>dd786dba-88e6-440b-9158-b8fae67ef67c</id>
    <title>Sample Vulnerability</title>
    <published>2015-08-04T18:13:51.0Z</published>
    <updated>2015-08-05T18:13:51.0Z</updated>
    <summary>A vulnerability issue identified by CVE-...</summary>
    <content type="application/xml"
        src="https://example.org/provider/vulns/123456/data"/>
  </entry>
  <entry>
      <!-- ...another entry... -->
  </entry>
</feed>
```

This Feed Document has two Atom Entries, one of which has been
elided. The first Entry illustrates an "atom:entry" element that
provides a summary of essential details about one particular
vulnerability. Based upon this summary information and the provided

   category information, a client may choose to do an HTTP GET request
   on the content "src" attribute to retrieve the full details of the
   vulnerability.

B.3. Entry Retrieval

   This appendix provides a non-normative example of a client retrieving
   a vulnerability as an Atom Entry.

   Having retrieved the Feed of interest, the client may then decide,
   based on the description and/or category information, that one of the
   Entries in the Feed is of further interest. The client may retrieve
   this vulnerability Entry by performing an HTTP GET operation on the
   URL indicated by the "src" attribute of the "atom:content" element.

   Example HTTP GET request for an Entry:

     GET /provider/public/vulns/123456
     Host: www.example.org
     Accept: application/atom+xml;type=entry

   The corresponding HTTP response would be an XML document containing
   the Atom Entry for the vulnerability record:

   Example HTTP GET response for an Entry:

     HTTP/1.1 200 OK
     Date: Fri, 24 Aug 2016 17:30:11 GMT
     Content-Length: 713
     Content-Type: application/atom+xml;type=entry;charset="utf-8"

     <?xml version="1.0" encoding="UTF-8"?>
     <entry xmlns="https://www.w3.org/2005/Atom"
         xmlns:rolie="urn:ietf:params:xml:ns:rolie-1.0"
         xml:lang="en-US">
       <id>f63aafa9-4082-48a3-9ce6-97a2d69d4a9b</id>
       <title>Sample Vulnerability</title>
       <published>2015-08-04T18:13:51.0Z</published>
       <updated>2015-08-05T18:13:51.0Z</updated>
       <category
           scheme="urn:ietf:params:rolie:category:information-type"
           term="vulnerability"/>
       <summary>A vulnerability issue identified by CVE-...</summary>
       <rolie:format ns="urn:ietf:params:xml:ns:exampleformat"/>
       <content type="application/xml"
           src="https://example.org/provider/vulns/123456/data">
       </content>
     </entry>

   The example response above shows an XML document referenced by the
   "src" attribute of the "atom:content" element. The client may
   retrieve the document using this URL.

Acknowledgements

   The authors gratefully acknowledge the valuable contributions of Tom
   Maguire, Kathleen Moriarty, and Vijayanand Bharadwaj. These
   individuals provided detailed review comments on earlier draft
   versions of this document and made many suggestions that have helped
   to improve this document.

   The authors would also like to thank the MILE Working Group, the SACM
   Working Group, and countless other people from both within the IETF
   community and outside of it for their excellent review and effort
   towards constructing this document.

Authors' Addresses

   John P. Field
   Pivotal Software, Inc.
   625 Avenue of the Americas
   New York, New York 10011
   United States of America

   Phone: (646)792-5770
   Email: jfield@pivotal.io

   Stephen A. Banghart
   National Institute of Standards and Technology
   100 Bureau Drive
   Gaithersburg, Maryland 20877
   United States of America

   Phone: (301)975-4288
   Email: stephen.banghart@nist.gov

   David Waltermire
   National Institute of Standards and Technology
   100 Bureau Drive
   Gaithersburg, Maryland 20877
   United States of America

   Email: david.waltermire@nist.gov