

MILE Working Group
Internet-Draft
Intended status: Informational
Expires: November 23, 2017

P. Kampanakis
Cisco Systems
M. Suzuki
NICT
May 22, 2017

IODEF Usage Guidance
draft-ietf-mile-iodef-guidance-10

Abstract

The Incident Object Description Exchange Format v2 [RFC7970] defines a data representation that provides a framework for sharing information commonly exchanged by Computer Security Incident Response Teams (CSIRTs) about computer security incidents. Since the IODEF model includes a wealth of available options that can be used to describe a security incident or issue, it can be challenging for security practitioners to develop tools that can leverage IODEF for incident sharing. This document provides guidelines for IODEF implementers. It also addresses how common security indicators can be represented in IODEF and use-cases of how IODEF is being used. This document aims to make IODEF's adoption by vendors easier and encourage faster and wider adoption of the model by Computer Security Incident Response Teams (CSIRTs) around the world.

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 <http://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 November 23, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

| | |
|---------------------------------------------------------------------------|----|
| 1. Introduction | 2 |
| 2. Terminology | 3 |
| 3. Implementation and Use Strategy | 3 |
| 3.1. Minimal IODEF document | 4 |
| 3.2. Information represented | 4 |
| 3.3. IODEF Classes | 5 |
| 4. Considerations | 6 |
| 4.1. External References | 6 |
| 4.2. Extensions | 6 |
| 4.3. Indicator predicate logic | 7 |
| 4.4. Disclosure level | 7 |
| 5. IODEF Uses | 8 |
| 5.1. Implementations | 8 |
| 5.2. Inter-vendor and Service Provider Exercise | 8 |
| 5.3. Use-cases | 11 |
| 6. Security Considerations | 12 |
| 7. Updates | 12 |
| 8. References | 14 |
| 8.1. Normative References | 14 |
| 8.2. Informative References | 15 |
| Appendix A. Indicator predicate logic examples | 15 |
| Appendix B. Inter-vendor and Service Provider Exercise Examples | 18 |
| B.1. Malware Delivery URL | 18 |
| B.2. DDoS | 19 |
| B.3. Spear-Phishing | 22 |
| B.4. Malware | 26 |
| B.5. IoT Malware | 32 |
| Authors' Addresses | 34 |

1. Introduction

The Incident Object Description Exchange Format v2 [RFC7970] defines a data representation that provides a framework for sharing information commonly exchanged by Computer Security Incident Response Teams (CSIRTs) about computer security incidents. The IODEF data

model consists of multiple classes and data types that are defined in the IODEF XML schema.

The IODEF schema was designed to be able to describe all the possible fields that would be needed in a security incident exchange. Thus, IODEF contains a plethora of data constructs that could potentially make it harder for IODEF implementers to decide which are important. Additionally, in the IODEF schema, there exist multiple fields and classes which do not necessarily need to be used in every possible data exchange. Moreover, some IODEF classes are useful only in rare circumstances. This document tries to address how to avoid these concerns. It also addresses how common security indicators can be represented in IODEF. It points out the most important IODEF classes for an implementer and describe other ones that are not as important. Also, it presents some common challenges for IODEF implementers and how to address them. The end goal of this document is to make IODEF's use by vendors easier and encourage wider adoption of the model by CSIRTs around the world.

Section 3 discusses the recommended classes and how an IODEF implementer should chose the classes to implement. Section 4 presents common considerations a practitioner will come across and how to address them. Section 5 goes over some common uses of IODEF.

2. Terminology

The terminology used in this document follows the one defined in [RFC7970] and [RFC7203].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3. Implementation and Use Strategy

It is important for IODEF implementers to be able to distinguish how the IODEF classes will be used in incident information exchanges. To do that one has to follow a strategy according to which of the various IODEF classes will be implemented. It is also important to know the most common classes that will be used to describe common security incidents or indicators. Thus, this section describes the most important classes and factors an IODEF practitioner should take into consideration before using IODEF, or designing an implementation.

3.1. Minimal IODEF document

IODEF includes one mandatory classes. An IODEF document MUST include at least an Incident class, an xml:lang attribute that defines the supported language an the IODEF version attribute. An Incident MUST contain three minimal mandatory-to-implement classes. An Incident class needs to have a Generation time and IncidentID class and at least one Contact class. The structure of the minimal-style Incident class follows below.

```
+-----+
| Incident |
+-----+
| ENUM purpose | <>-----[ IncidentID      ]
|              | <>-----[ GenerationTime  ]
|              | <>--{1..*}--[ Contact        ]
+-----+
```

Minimal-style Incident class

The minimal Incident class needs to include a purpose attribute and the IncidentID, GenerationTime, and Contact elements.

The Contact class requires the type and role attributes, but no elements are required by the IODEF v2 specification. Nevertheless, at least one of the elements in the Contact class, such as Email class, SHOULD be implemented so that the IODEF document can be practical.

Implementers can refer to Appendix B and Section 7 of [RFC7970] for example IODEF v2 documents.

3.2. Information represented

There is no need for a practitioner to use or implement IODEF classes and fields other than the minimal ones (Section 3.1) and the ones that are necessary for her use-cases. The implementer should carefully look into the schema and decide classes to implement (or not).

For example, if we have Distributed Denial of Service (DDoS) as a potential use-case, then the Flow class and its included information are the most important classes to use. The Flow class describes information related to the attacker hosts and victim hosts, which information could help automated filtering or sink-hole operations.

Another potential use-case is malware command and control (c2). After modern malware infects a device, it usually proceeds to connect

to one or more c2 servers to receive instructions from its master and potentially exfiltrate information. To protect against such activity, it is important to interrupt the c2 communication by filtering the activity. IODEF can describe c2 activities using the Flow and the ServiceName classes.

For use-cases where indicators need to be described, the IndicatorData class its classes will be implemented instead of the EventData class.

In summary, an implementer SHOULD identify her use-cases and find the classes that are necessary to support in IODEF v2. Implementing and parsing all IODEF classes can be cumbersome in some occasions and is not always necessary. Other external schemata can also be used in IODEF to describe incidents or indicators which should be treated accordingly only if the implementer's IODEF use-cases require external schema support.

IODEF supports multiple translations of free-form text in all ML_STRING classes [RFC7970]. That way text can be translated to different languages by using the same translation identifier in the class. Implementers SHOULD be able to parse iodef:MLStringType classes and extract only the information relevant to the language/s of interest.

3.3. IODEF Classes

[RFC7970] contains classes that can describe attack Methods, Events, Incidents, how they were discovered and the Assessment of the repercussions of the incident to the victim. It is important for IODEF users to know the distinction between these classes in order to decide which ones fulfill their use-cases.

An IndicatorData class depicts a threat indicator or observable that could be used to describe a threat that does not necessarily mean that an successful attack happened. For example, we could see an attack happening but it might have been prevented and not have resulted in an incident or security event. On the other hand an EventData class usually describes a security event and can be considered as an incident report of something that took place.

Classes like Discovery, Assessment, Method, and RecoveryTime are used in conjunction with EventData as they related to the incident report described in the EventData. The RelatedActivity class can reference an incident, an indicator or other related threat activity.

While deciding what classes are important for the needed use-cases, IODEF users SHOULD carefully evaluate the necessary classes and how

these are used in order to avoid unnecessary work. For example, if we want to only describe indicators in IODEF, the implementation of Method or Assessment might not be important.

4. Considerations

Implementers need to consider some common, standardized options for their IODEF use strategy.

4.1. External References

The IODEF format includes the Reference class that refers to externally defined information such as a vulnerability, Intrusion Detection System (IDS) alert, malware sample, advisory, or attack technique. To facilitate the exchange of information, the Reference class was extended to the Enumeration Reference Format [RFC7495]. The Enumeration Reference Format specifies a means to use external enumeration specifications (e.g. CVE) that could define an enumeration format, specific enumeration values, or both. As external enumerations can vary greatly, implementers SHOULD only support external enumerations that are expected to describe their specific use-cases.

4.2. Extensions

The IODEF data model ([RFC7970]) is extensible. Many attributes with enumerated values can be extended using the "ext-*" prefix. Additional classes can also be defined by using the AdditionalData and RecordItem classes. An extension to the AdditionalData class for reporting Phishing emails is defined in [RFC5901]. Information about extending IODEF class attributes and enumerated values can be found in Section 5 of [RFC7970].

Additionally, IODEF can import existing schemata by using an extension framework defined in [RFC7203]. The framework enables IODEF users to embed XML data inside an IODEF document using external schemata or structures defined by external specifications. Examples include CVE, CVRF and OVAL. Thus, [RFC7203] enhances the IODEF capabilities without further extending the data model.

IODEF implementers SHOULD NOT consider using their own IODEF extensions unless data cannot be represented using existing standards or importing them in an IODEF document using [RFC7203] is not a suitable option.

4.3. Indicator predicate logic

An IODEF [RFC7970] document can describe incident reports and indicators. The Indicator class can include references to other indicators, observables and more classes that contain details about the indicator. When describing security indicators, it is often common to need to group them together in order to form a group of indicators that constitute a security threat. For example, a botnet might have multiple command and control servers. For that reason, IODEF v2 introduced the IndicatorExpression class that is used to add the indicator predicate logic when grouping more than one indicator or observable.

Implementations MUST be able to parse and apply the Boolean logic offered by an IndicatorExpression in order to evaluate the existence of an indicator. As explained in Section 3.29.5 of [RFC7970] the IndicatorExpression element operator defines the operator applied to all the child element of the IndicatorExpression. If no operator is defined "and" SHOULD be assumed. IndicatorExpressions can also be nested together. Child IndicatorExpressions should be treated as child elements of their parent and they SHOULD be evaluated first before evaluated with the operator of their parent.

Users can refer to Appendix A for example uses of the IndicatorExpressions in an IODEF v2.

4.4. Disclosure level

The information conveyed in IODEF documents SHOULD be treated carefully since the content may be confidential. IODEF has a common attribute, called "restriction", which indicates the disclosure guideline to which the sender expects the recipient to adhere to for the information represented in the class and its children. That way, the sender can express the level of disclosure for each component of an IODEF document. Appropriate external measures could be implemented based on the restriction level. One example is when Real-time Inter-network Defense (RID) [RFC6545] is used to transfer the IODEF documents, it can provide policy guidelines for handling IODEF documents by using the RIDPolicy class.

The enforcement of the disclosure guidelines is out of scope for IODEF. The recipient of the IODEF document needs to follow the guidelines, but these guidelines themselves do not provide any enforcement measures. For that purpose, implementers SHOULD consider appropriate measures, technical or operational.

5. IODEF Uses

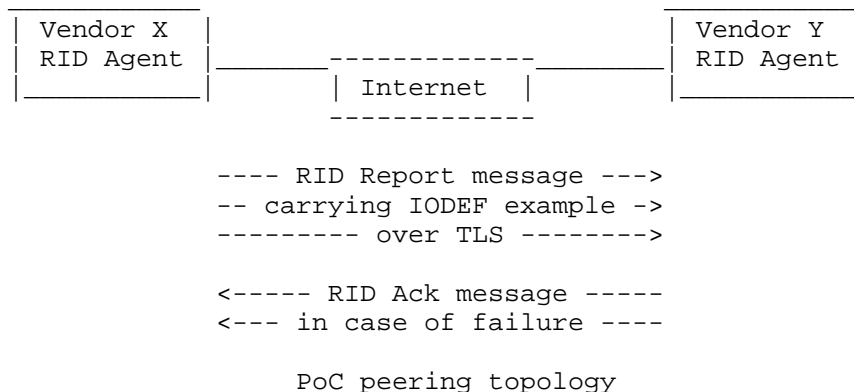
IODEF is currently used by various organizations in order to represent security incidents and share incident and threat information between security operations organizations.

5.1. Implementations

In order to use IODEF, tools like IODEF parsers are necessary. [I-D.ietf-mile-implementreport] describes a set of IODEF implementations and uses by various vendors and Computer Security Incident Response Teams (CERT) organizations. The document does not specify any specific mandatory to implement (MTI) IODEF classes but provides a list of real world uses. Perl and Python modules (XML::IODEF, Iodef::Pb, iodeflib) are some examples. Section 7 also includes practical IODEF use guidelines. Implementers are encouraged to refer to [I-D.ietf-mile-implementreport]. [implementations], on the other hand, includes various vendor incident reporting products that can consume and export in IODEF format.

5.2. Inter-vendor and Service Provider Exercise

As an interoperability exercise, in 2013 a limited number of vendors organized and executed threat indicators exchanges in IODEF. The transport protocol used was RID. The threat information shared included indicators from DDoS attacks; and Malware and Spear-Phishing incidents. The results served as proof-of-concept (PoC) about how seemingly competing entities could use IODEF to exchange sanitized security information. As this was a PoC exercise only example information (no real threats) were shared as part of the exchanges.



The figure above shows how RID interactions took place during the PoC. Participating organizations were running RID Agent software on-

premises. The RID Agents formed peering relationships with other participating organizations. When Entity X had a new incident to exchange it would package it in IODEF and send it to Entity Y over TLS in a RID Report message. In case there was an issue with the message, Entity Y would send an RID Acknowledgement message back to Entity X which included an application level message to describe the issue. Interoperability between RID agents and the standards, Use of [RFC6545] and [RFC6546], were also proven in this exercise.

The first use-case included sharing of Malware Data Related to an Incident between CSIRTs. After Entity X detected an incident, she would put data about malware found during the incident in a backend system. Entity X then decided to share the incident information with Entity Y about the malware discovered. This could be a human decision or part of an automated process.

Below are the steps followed for the malware information exchange that was taking place:

- (1) Entity X has a sharing agreement with Entity Y, and has already been configured with the IP address of Entity Y's RID Agent
- (2) Entity X's RID Agent connects to Entity Y's RID Agent, and mutual authentication occurs using PKI certificates.
- (3) Entity X pushes out a RID Report message which contains information about N pieces of discovered malware. IODEF is used in RID to describe the
 - (a) Hash of malware files
 - (b) Registry settings changed by the malware
 - (c) C&C Information for the malware
- (4) Entity Y receives RID Report message, sends RID Acknowledgement message
- (5) Entity Y stores the data in a format that makes it possible for the back end to know which source the data came from.

Another use-case was sharing a DDoS attack as explained in the following scenario: Entity X, a Critical Infrastructure and Key Resource (CIKR) company detects that their internet connection is saturated with an abnormal amount of traffic. Further investigation determines that this is an actual DDoS attack. Entity X's CSIT contacts their ISP, Entity Y, and shares information with them about the attack traffic characteristics. Entity X's ISP is being

overwhelmed by the amount of traffic, so it shares attack signatures and IP addresses of the most prolific hosts with its adjacent ISPs.

Below are the steps followed for a DDoS information exchange:

- (1) Entity X has a sharing agreement with Entity Y, and has already been configured with the IP address of Entity Y's RID Agent
- (2) Entity X's RID Agent connects to Entity Y's RID Agent, and mutual authentication occurs using PKI certificates.
- (3) Entity X pushes out a RID Report message which contains information about the DDoS attack. IODEF is used in RID to describe the
 - (a) Start and Detect dates and times
 - (b) IP Addresses of nodes sending DDoS Traffic
 - (c) Sharing and Use Restrictions
 - (d) Traffic characteristics (protocols and ports)
 - (e) HTTP User-Agents used
 - (f) IP Addresses of C&C for a botnet
- (4) Entity Y receives RID Report message, sends RID Acknowledgement message
- (5) Entity Y stores the data in a format that makes it possible for the back end to know which source the data came from.
- (6) Entity Y shares information with other ISP Entities it has an established relationship with.

One more use-case was sharing spear-phishing email information as explained in the following scenario: The board members of several defense contractors receive an email inviting them to attend a conference in San Francisco. The board members are asked to provide their personally identifiable information such as their home address, phone number, corporate email, etc in an attached document which came with the email. The board members are also asked to click on a URL which would allow them to reach the sign up page for the conference. One of the recipients believes the email to be a phishing attempt and forwards the email to their corporate CSIRT for analysis. The CSIRT identifies the email as an attempted spear phishing incident and distributes the indicators to their sharing partners.

Below are the steps followed for a spear-phishing information exchange between CSIRTs that was part of this PoC.

- (1) Entity X has a sharing agreement with Entity Y, and has already been configured with the IP address of Entity Y's RID Agent
- (2) Entity X's RID Agent connects to Entity Y's RID Agent, and mutual authentication occurs using PKI certificates.
- (3) Entity X pushes out a RID Report message which contains information about the spear-phishing email. IODEF is used in RID to describe the
 - (a) Attachment details (file Name, hash, size, malware family)
 - (b) Target description (IP, domain, NSLookup)
 - (c) Email information (From, Subject, header information, date/time, digital signature)
 - (d) Confidence Score
- (4) Entity Y receives RID Report message, sends RID Acknowledgement message
- (5) Entity Y stores the data in a format that makes it possible for the back end to know which source the data came from.

Appendix B includes some of the incident IODEF example information that was exchanged by the organizations' RID Agents as part of this proof-of-concept.

5.3. Use-cases

Other use-cases of IODEF, other than the ones described above, could be:

- (1) ISP notifying a national CERT or organization when it identifies and acts upon an incident and CERTs notifying ISPs when they are aware of incidents.
- (2) Suspected phishing emails could be shared amongst organizations and national agencies. Automation could validate web content that the suspicious emails are pointing to. Identified malicious content linked in a phishing email could then be shared using IODEF. Phishing campaigns could thus be subverted much faster by automating information sharing using IODEF.

- (3) When finding a certificate that should be revoked, a third-party would forward an automated IODEF message to the CA with the full context of the certificate and the CA could act accordingly after checking its validity. Alternatively, in the event of a compromise of the private key of a certificate, a third-party could alert the certificate owner about the compromise using IODEF.

6. Security Considerations

This document does not incur any new security issues, since it only talks about the usage of IODEFv2 defined RFC7970. Nevertheless, readers of this document SHOULD refer to the Security Considerations section of [RFC7970].

7. Updates

[EDNOTE: To delete during last call.]

version -10 updates:

- (1) Fixed nits identified by Adam M.
- (2) Added paragraph about language support in ML_STRING classes.

version -09 updates:

- (1) Made changes according to suggestions in IETF-98.

version -08 updates:

- (1) Updated Appendix IODEFv2 examples.
- (2) Moved Predicate logic examples in appendix.
- (3) Syntax and grammar fixes, clarifications, wording.
- (4) Reorganized IODEF uses section and subsections.

version -07 updates:

- (1) Updated examples in Appendix A to follow IODEFv2.

version -06 updates:

- (1) Updated wording in various sections to make content clearer.

- (2) Updated Predicate Logic section to reflect the latest IndicatorExpression logic in iodef-bis.
- (3) Updated section to describe the difference between events and indicators and their use in IODEF v2.

version -05 updates:

- (1) Changed section title from "Restrictions in IODEF" to "Disclosure level of IODEF" and added some description
- (2) Mixed "Recommended classes to implement" section with "Unnecessary Fields" section into "Minimal IODEF document" section
- (3) Added description to "Decide what IODEF will be used for" section, "Implementations" section, and "Security Considerations" section

version -04 updates:

- (1) Expanded on the Extensions section using Take's suggestion.
- (2) Moved Future use-cases under the Other section.
- (3) CIF and APWG were consolidated in one "Implementation" section
- (4) Added abstract of RFC7495 to the "External References" section
- (5) Added Kathleen's example of malware delivery URL to "Appendix"
- (6) Added a little description to "Recommended classes to implement" section

version -03 updates:

- (1) Added "Updates" section.
- (2) Added details about the flow of information exchanges in "Inter-vendor and Service Provider Exercise" section. Also updated the usecases with more background information.
- (3) Added future use-cases in the "Collective Intelligence Framework" section
- (4) Updated Perl and Python references with the actual module names. Added IODEF implementation reference "implementations".

- (5) Added Predicate logic section
- (6) Updated Logic of watchlist of indicators section to simplify the logic and include examples.
- (7) Renamed externally defined indicators section to Indicator reference and elaborated on the use of indicator-uid and indicator-set-uid attribute use.

version -02 updates:

- (1) Updated the "Logic for watchlist of indications" section to clarify the logic based on community feedback.
- (2) Added "Inter-vendor and Service Provider Exercise" section.
- (3) Added Appendix to include actual use-case IODEF examples.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC5901] Cain, P. and D. Jevans, "Extensions to the IODEF-Document Class for Reporting Phishing", RFC 5901, DOI 10.17487/RFC5901, July 2010, <<http://www.rfc-editor.org/info/rfc5901>>.
- [RFC6545] Moriarty, K., "Real-time Inter-network Defense (RID)", RFC 6545, DOI 10.17487/RFC6545, April 2012, <<http://www.rfc-editor.org/info/rfc6545>>.
- [RFC7203] Takahashi, T., Landfield, K., and Y. Kadobayashi, "An Incident Object Description Exchange Format (IODEF) Extension for Structured Cybersecurity Information", RFC 7203, DOI 10.17487/RFC7203, April 2014, <<http://www.rfc-editor.org/info/rfc7203>>.
- [RFC7495] Montville, A. and D. Black, "Enumeration Reference Format for the Incident Object Description Exchange Format (IODEF)", RFC 7495, DOI 10.17487/RFC7495, March 2015, <<http://www.rfc-editor.org/info/rfc7495>>.

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

8.2. Informative References

[I-D.ietf-mile-implementreport]
Inacio, C. and D. Miyamoto, "MILE Implementation Report", draft-ietf-mile-implementreport-10 (work in progress), November 2016.

[implementations]
"Implementations on IODEF",
<<http://siis.realmv6.org/implementations/>>.

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

Appendix A. Indicator predicate logic examples

In the following example the EventData class evaluates as a Flow of one System with source address being (10.10.10.104 OR 10.10.10.106) AND target address 10.1.1.1.

```
<!-- ...XML code omitted... -->
<IndicatorData>
  <Indicator>
    <IndicatorID name="csirt.example.com" version="1">
      G90823490
    </IndicatorID>
    <Description>C2 domains</Description>
    <IndicatorExpression operator="and">
      <IndicatorExpression operator="or">
        <Observable>
          <System category="source" spoofed="no">
            <Node>
              <Address category="ipv4-addr">
                10.10.10.104
              </Address>
            </Node>
          </System>
        </Observable>
        <Observable>
          <System category="source" spoofed="no">
            <Node>
              <Address category="ipv4-addr">
                10.10.10.106
              </Address>
            </Node>
          </System>
        </Observable>
      </IndicatorExpression>
    </Observable>
    <System category="target" spoofed="no">
      <Node>
        <Address category="ipv4-addr">
          10.1.1.1
        </Address>
      </Node>
    </System>
  </Observable>
</Indicator>
</IndicatorData>
<!-- ...XML code omitted... -->
```

Similarly, the FileData Class can be an observable in an IndicatorExpression. The hash values of two files can be used to match against an indicator using Boolean "or" logic. In the following example the indicator consists of either of the two files with two different hashes.


```
<!-- ...XML code omitted... -->
<IndicatorData>
  <Indicator>
    <IndicatorID name="csirt.example.com" version="1">
      A4399IWQ
    </IndicatorID>
    <Description>File hash watchlist</Description>
    <IndicatorExpression operator="or">
      <Observable>
        <FileData>
          <File>
            <FileName>dummy.txt</FileName>
            <HashData scope="file-contents">
              <Hash>
                <ds:DigestMethod Algorithm=
                  "http://www.w3.org/2001/04/xmlenc#sha256"/>
                <ds:DigestValue>
                  141accec23e7e5157de60853cb1e01bc38042d
                  08f9086040815300b7fe75c184
                </ds:DigestValue>
              </Hash>
            </HashData>
          </File>
        </FileData>
      </Observable>
      <Observable>
        <FileData>
          <File>
            <FileName>dummy2.txt</FileName>
            <HashData scope="file-contents">
              <Hash>
                <ds:DigestMethod Algorithm=
                  "http://www.w3.org/2001/04/xmlenc#sha256"/>
                <ds:DigestValue>
                  141accec23e7e5157de60853cb1e01bc38042d
                  08f9086040815300b7fe75c184
                </ds:DigestValue>
              </Hash>
            </HashData>
          </File>
        </FileData>
      </Observable>
    </IndicatorExpression>
  </Indicator>
</IndicatorData>
<!-- ...XML code omitted... -->
```

Appendix B. Inter-vendor and Service Provider Exercise Examples

Below some of the incident IODEF example information that was exchanged by the vendors as part of this proof-of-concept Inter-vendor and Service Provider Exercise.

B.1. Malware Delivery URL

This example indicates malware and related URL for file delivery.

```

<?xml version="1.0" encoding="UTF-8"?>
<IODEF-Document version="2.00"
  xmlns="urn:ietf:params:xml:ns:iodef-2.0"
  xmlns:iodef="urn:ietf:params:xml:ns:iodef-2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <iodef:Incident purpose="reporting">
    <iodef:IncidentID name="csirt.example.com">
      189801
    </iodef:IncidentID>
    <iodef:ReportTime>2012-12-05T12:20:00+00:00</iodef:ReportTime>
    <iodef:GenerationTime>2012-12-05T12:20:00+00:00</iodef:GenerationTime>
    <iodef:Description>Malware and related indicators</iodef:Description>
    <iodef:Assessment occurrence="potential">
      <iodef:SystemImpact severity="medium" type="breach-privacy">
        <iodef:Description>Malware with C&C
        </iodef:Description>
      </iodef:SystemImpact>
    </iodef:Assessment>
    <iodef:Contact role="creator" type="organization">
      <iodef:ContactName>example.com CSIRT
      </iodef:ContactName>
      <iodef:Email>
        <iodef:EmailTo>contact@csirt.example.com
        </iodef:EmailTo>
      </iodef:Email>
    </iodef:Contact>
    <iodef:EventData>
      <iodef:Flow>
        <iodef:System category="source">
          <iodef:Node>
            <iodef:Address category="ipv4-addr">192.0.2.200
            </iodef:Address>
            <iodef:Address category="site-uri">
              /log-bin/lunch_install.php?aff_id=1&lunch_id=1&maddr=&
action=install
            </iodef:Address>
          </iodef:Node>
          <iodef:NodeRole category="www"/>
        </iodef:System>
      </iodef:Flow>
    </iodef:EventData>
  </iodef:Incident>
</IODEF-Document>

```

B.2. DDoS

The DDoS test exchanged information that described a DDoS including protocols and ports, bad IP addresses and HTTP User-Agent fields.

The IODEF version used for the data representation was based on [RFC7970].

```
<?xml version="1.0" encoding="UTF-8"?>
<IODEF-Document version="2.00"
  xmlns="urn:ietf:params:xml:ns:iodef-2.0"
  xmlns:iodef="urn:ietf:params:xml:ns:iodef-2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <iodef:Incident purpose="reporting" restriction="default">
    <iodef:IncidentID name="csirt.example.com">
      189701
    </iodef:IncidentID>
    <iodef:DetectTime>2013-02-05T01:15:45+00:00</iodef:DetectTime>
    <iodef:StartTime>2013-02-05T00:34:45+00:00</iodef:StartTime>
    <iodef:ReportTime>2013-02-05T01:34:45+00:00</iodef:ReportTime>
    <iodef:GenerationTime>2013-02-05T01:15:45+00:00</iodef:GenerationTime>
    <iodef:Description>DDoS Traffic Seen</iodef:Description>
    <iodef:Assessment occurrence="actual">
      <iodef:SystemImpact severity="medium" type="availability-system">
        <iodef:Description>DDoS Traffic
        </iodef:Description>
      </iodef:SystemImpact>
      <iodef:Confidence rating="high"/>
    </iodef:Assessment>
    <iodef:Contact role="creator" type="organization">
      <iodef:ContactName>Dummy Test</iodef:ContactName>
      <iodef:Email>
        <iodef:EmailTo>contact@dummytest.com
        </iodef:EmailTo>
      </iodef:Email>
    </iodef:Contact>
    <iodef:EventData>
      <iodef:Description>
        Dummy Test sharing with ISP1
      </iodef:Description>
      <iodef:Method>
        <iodef:Reference>
          <iodef:URL>
            http://blog.spiderlabs.com/2011/01/loic-ddos-
            analysis-and-detection.html
          </iodef:URL>
          <iodef:URL>
            http://en.wikipedia.org/wiki/Low_Orbit_Ion_Cannon
          </iodef:URL>
          <iodef:Description>
            Low Orbit Ion Cannon User Agent
          </iodef:Description>
        </iodef:Reference>
      </iodef:Method>
    </iodef:EventData>
  </iodef:Incident>
</IODEF-Document>
```

```
</iodef:Method>
<iodef:Flow>
  <iodef:System category="source" spoofed="no">
    <iodef:Node>
      <iodef:Address category="ipv4-addr">
        192.0.2.104
      </iodef:Address>
    </iodef:Node>
    <iodef:Service ip-protocol="6">
      <iodef:Port>1337</iodef:Port>
    </iodef:Service>
  </iodef:System>
  <iodef:System category="source" spoofed="no">
    <iodef:Node>
      <iodef:Address category="ipv4-addr">
        192.0.2.106
      </iodef:Address>
    </iodef:Node>
    <iodef:Service ip-protocol="6">
      <iodef:Port>1337</iodef:Port>
    </iodef:Service>
  </iodef:System>
  <iodef:System category="source" spoofed="yes">
    <iodef:Node>
      <iodef:Address category="ipv4-net">
        198.51.100.0/24
      </iodef:Address>
    </iodef:Node>
    <iodef:Service ip-protocol="6">
      <iodef:Port>1337</iodef:Port>
    </iodef:Service>
  </iodef:System>
  <iodef:System category="source" spoofed="yes">
    <iodef:Node>
      <iodef:Address category="ipv6-addr">
        2001:db8:dead:beef::1
      </iodef:Address>
    </iodef:Node>
    <iodef:Service ip-protocol="6">
      <iodef:Port>1337</iodef:Port>
    </iodef:Service>
  </iodef:System>
  <iodef:System category="target">
    <iodef:Node>
      <iodef:Address category="ipv4-addr">
        203.0.113.1
      </iodef:Address>
    </iodef:Node>
  </iodef:System>
</iodef:Flow>
```

```

        <iodef:Service ip-protocol="6">
          <iodef:Port>80</iodef:Port>
        </iodef:Service>
      </iodef:System>
      <iodef:System category="sensor">
        <iodef:Node>
        </iodef:Node>
        <iodef:Description>
          Information provided in Flow class instance is from
          Inspection of traffic from network tap
        </iodef:Description>
      </iodef:System>
    </iodef:Flow>
    <iodef:Expectation action="other"/>
  </iodef:EventData>
  <iodef:IndicatorData>
    <iodef:Indicator>
      <iodef:IndicatorID name="csirt.example.com" version="1">
        G83345941
      </iodef:IndicatorID>
      <iodef:Description>
        User-Agent string
      </iodef:Description>
      <iodef:Observable>
        <iodef:BulkObservable type="http-user-agent">
          <iodef:BulkObservableList>
            user-agent="Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10.5; en-US;
rv:1.9.2.12) Gecko/20101026 Firefox/3.6.12">
          </iodef:BulkObservableList>
        </iodef:BulkObservable>
      </iodef:Observable>
    </iodef:Indicator>
  </iodef:IndicatorData>
</iodef:Incident>
</IODEF-Document>

```

B.3. Spear-Phishing

The Spear-Phishing test exchanged information that described a Spear-Phishing email including DNS records and addresses about the sender, malicious attached file information and email data. The IODEF version used for the data representation was based on [RFC7970].

```

<?xml version="1.0" encoding="UTF-8"?>
<IODEF-Document version="2.00"
  xmlns="urn:ietf:params:xml:ns:iodef-2.0"
  xmlns:iodef="urn:ietf:params:xml:ns:iodef-2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

```

```
<iodef:Incident purpose="reporting">
  <iodef:IncidentID name="csirt.example.com">
    189601
  </iodef:IncidentID>
  <iodef:DetectTime>2013-01-04T08:06:12+00:00</iodef:DetectTime>
  <iodef:StartTime>2013-01-04T08:01:34+00:00</iodef:StartTime>
  <iodef:EndTime>2013-01-04T08:31:27+00:00</iodef:EndTime>
  <iodef:ReportTime>2013-01-04T09:15:45+00:00</iodef:ReportTime>
  <iodef:GenerationTime>2013-01-04T09:15:45+00:00</iodef:GenerationTime>
  <iodef:Description>
    Zeus Spear Phishing E-mail with Malware Attachment
  </iodef:Description>
  <iodef:Assessment occurrence="potential">
    <iodef:SystemImpact severity="medium" type="takeover-system">
      <iodef:Description>
        Malware with Command and Control Server and System Changes
      </iodef:Description>
    </iodef:SystemImpact>
  </iodef:Assessment>
  <iodef:Contact role="creator" type="organization">
    <iodef:ContactName>example.com CSIRT</iodef:ContactName>
    <iodef:Email>
      <iodef:EmailTo>contact@csirt.example.com</iodef:EmailTo>
    </iodef:Email>
  </iodef:Contact>
  <iodef:EventData>
    <iodef:Description>
      Targeting Defense Contractors,
      specifically board members attending Dummy Con
    </iodef:Description>
    <iodef:Method>
      <iodef:Reference observable-id="ref-1234">
        <iodef:Description>Zeus</iodef:Description>
      </iodef:Reference>
    </iodef:Method>
    <iodef:Flow>
      <iodef:System category="source">
        <iodef:Node>
          <iodef:Address category="site-uri">
            http://www.zeusevil.example.com
          </iodef:Address>
          <iodef:Address category="ipv4-addr">
            192.0.2.166
          </iodef:Address>
          <iodef:Address category="asn">
            65535
          </iodef:Address>
          <iodef:Address category="ext-value">
```

```

        ext-category="as-name">
        EXAMPLE-AS - University of Example"
    </iodef:Address>
    <iodef:Address category="ext-value"
        ext-category="as-prefix">
        192.0.2.0/24
    </iodef:Address>
</iodef:Node>
    <iodef:NodeRole category="malware-distribution"/>
</iodef:System>
</iodef:Flow>
<iodef:Flow>
    <iodef:System category="source">
        <iodef:Node>
            <iodef:DomainData>
                <Name>maill.evildave.example.com</Name>
            </iodef:DomainData>
            <iodef:Address category="ipv4-addr">
                198.51.100.6
            </iodef:Address>
            <iodef:Address category="asn">
                65534
            </iodef:Address>
            <iodef:Address category="ext-value"
                ext-category="as-name">
                EXAMPLE-AS - University of Example
            </iodef:Address>
            <iodef:DomainData>
                <iodef:Name>evildave.example.com</iodef:Name>
                <iodef:DateDomainWasChecked>2013-01-04T09:10:24+00:00
            </iodef:DateDomainWasChecked>
                <!-- <iodef:RelatedDNS RecordType="MX"> -->
                <iodef:RelatedDNS dtype="string">
                    evildave.example.com MX prefernce = 10, mail exchanger
                    = maill.evildave.example.com
                </iodef:RelatedDNS>
                <iodef:RelatedDNS dtype="string">
                    maill.evildave.example.com
                    internet address = 198.51.100.6
                </iodef:RelatedDNS>
                <iodef:RelatedDNS dtype="string">
                    zuesevil.example.com. IN TXT \"v=spf1 a mx -all\"
                </iodef:RelatedDNS>
            </iodef:DomainData>
        </iodef:Node>
        <iodef:NodeRole category="mail">
            <iodef:Description>
                Sending phishing mails

```



```
</iodef:Description>
</iodef:NodeRole>
<iodef:Service>
  <iodef:EmailData>
    <iodef:EmailFrom>
      emaildave@evildave.example.com
    </iodef:EmailFrom>
    <iodef:EmailSubject>
      Join us at Dummy Con
    </iodef:EmailSubject>
    <iodef:EmailX-Mailer>
      StormRider 4.0
    </iodef:EmailX-Mailer>
  </iodef:EmailData>
</iodef:Service>
</iodef:System>
<iodef:System category="target">
  <iodef:Node>
    <iodef:Address category="ipv4-addr">
      203.0.113.2
    </iodef:Address>
  </iodef:Node>
</iodef:System>
</iodef:Flow>
<iodef:Expectation action="other"/>
<iodef:Record>
  <iodef:RecordData>
    <iodef:FileData observable-id="fd-1234">
      <iodef:File>
        <iodef:FileName>
          Dummy Con Sign Up Sheet.txt
        </iodef:FileName>
        <iodef:FileSize>
          152
        </iodef:FileSize>
        <iodef:HashData scope="file-contents">
          <iodef:Hash>
            <ds:DigestMethod
              Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
            <ds:DigestValue>
              141accec23e7e5157de60853cb1e01bc38042d
              08f9086040815300b7fe75c184
            </ds:DigestValue>
          </iodef:Hash>
        </iodef:HashData>
      </iodef:File>
    </iodef:FileData>
  </iodef:RecordData>
```

```

    <iodef:RecordData>
      <iodef:CertificateData>
        <iodef:Certificate>
          <ds:X509Data>
            <ds:X509IssuerSerial>
              <ds:X509IssuerName>FakeCA
            </ds:X509IssuerName>
            <ds:X509SerialNumber>
              57482937101
            </ds:X509SerialNumber>
          </ds:X509IssuerSerial>
          <ds:X509SubjectName>EvilDaveExample
          </ds:X509SubjectName>
        </ds:X509Data>
      </iodef:Certificate>
    </iodef:CertificateData>
  </iodef:RecordData>
</iodef:Record>
</iodef:EventData>
</iodef:Incident>
</IODEF-Document>

```

B.4. Malware

In this test, malware information was exchanged using RID and IODEF. The information included file hashes, registry setting changes and the C&C servers the malware uses.

```

<?xml version="1.0" encoding="UTF-8"?>
<IODEF-Document version="2.00"
  xmlns="urn:ietf:params:xml:ns:iodef-2.0"
  xmlns:iodef="urn:ietf:params:xml:ns:iodef-2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <iodef:Incident purpose="reporting">
    <iodef:IncidentID name="csirt.example.com">
      189234
    </iodef:IncidentID>
    <iodef:ReportTime>2013-03-07T16:14:56.757+05:30</iodef:ReportTime>
    <iodef:GenerationTime>2013-03-07T16:14:56.757+05:30</iodef:GenerationTime>
    <iodef:Description>
      Malware and related indicators identified
    </iodef:Description>
    <iodef:Assessment occurrence="potential">
      <iodef:SystemImpact severity="medium" type="breach-proprietary">
        <iodef:Description>
          Malware with Command and Control Server and System Changes
        </iodef:Description>

```

```

    </iodef:SystemImpact>
  </iodef:Assessment>
  <iodef:Contact role="creator" type="organization">
    <iodef:ContactName>example.com CSIRT</iodef:ContactName>
    <iodef:Email>
      <iodef:EmailTo>contact@csirt.example.com</iodef:EmailTo>
    </iodef:Email>
  </iodef:Contact>
  <iodef:EventData>
    <iodef:Method>
      <iodef:Reference>
        <iodef:URL>
          http://www.threatexpert.example.com/report.aspx?
            md5=e2710ceb088dacdc03678db250742b7
        </iodef:URL>
        <iodef:Description>Zeus</iodef:Description>
      </iodef:Reference>
    </iodef:Method>
    <iodef:Flow>
      <iodef:System category="source">
        <iodef:Node>
          <iodef:Address category="ipv4-addr" observable-id="addr-c2-91011-001"
">
            203.0.113.200
          </iodef:Address>
          <iodef:Address category="site-uri" observable-id="addr-c2-91011-002"
>
            http://zeus.556677889900.example.com/log-bin/
            lunch_install.php?aff_id=1&amp;amp;
            lunch_id=1&amp;amp;maddr=&amp;amp;
            action=install
          </iodef:Address>
        </iodef:Node>
        <iodef:NodeRole category="c2-server"/>
      </iodef:System>
    </iodef:Flow>
    <iodef:Record>
      <iodef:RecordData>
        <iodef:FileData observable-id="file-91011-001">
          <iodef:File>
            <iodef:HashData scope="file-contents">
              <iodef:Hash>
                <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#s
hal"/>
                <ds:DigestValue>
                  MHg2NzUxQTIlMzQ4M0E2N0Q4NkUwRjg0NzYwRjYxRjEwQkJDQzJFREZG
                </ds:DigestValue>
              </iodef:Hash>
            </iodef:HashData>
          </iodef:File>
        </iodef:File>

```

```

    <iodef:HashData scope="file-contents">
      <iodef:Hash>
        <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#m
d5"/>
        <ds:DigestValue>
          MHgyRTg4ODA5ODBENjI0NDdFOTc5MEFGQTg5NTEzRjBBNA==
        </ds:DigestValue>
      </iodef:Hash>
    </iodef:HashData>
  </iodef:File>
</iodef:FileData>
<iodef:WindowsRegistryKeysModified observable-id="regkey-91011-001">
  <iodef:Key registryaction="add-value">
    <iodef:KeyName>
      HKLM\Software\Microsoft\Windows\
      CurrentVersion\Run\tamg
    </iodef:KeyName>
    <iodef:Value>
      ?\?\?%System%\wins\mc.exe\?\??
    </iodef:Value>
  </iodef:Key>
  <iodef:Key registryaction="modify-value">
    <iodef:KeyName>HKLM\Software\Microsoft\
      Windows\CurrentVersion\Run\dgo
    </iodef:KeyName>
    <iodef:Value>"\" \"%Windir%\Resources\
      Themes\Luna\km.exe\?\?"
    </iodef:Value>
  </iodef:Key>
</iodef:WindowsRegistryKeysModified>
</iodef:RecordData>
</iodef:Record>
</iodef:EventData>
<iodef:EventData>
  <iodef:Method>
    <iodef:Reference>
      <iodef:URL>
        http://www.threatexpert.example.com/report.aspx?
        md5=c3c528c939f9b176c883ae0ce5df0001
      </iodef:URL>
      <iodef:Description>Cridex</iodef:Description>
    </iodef:Reference>
  </iodef:Method>
</iodef:Flow>
  <iodef:System category="source">
    <iodef:Node>
      <iodef:Address category="ipv4-addr" observable-id="addr-c2-91011-003
">
        203.0.113.100
      </iodef:Address>

```

```

    </iodef:Node>
    <iodef:NodeRole category="c2-server"/>
    <iodef:Service ip-protocol="6">
      <iodef:Port>8080</iodef:Port>
    </iodef:Service>
  </iodef:System>
</iodef:Flow>
<iodef:Record>
  <iodef:RecordData>
    <iodef:FileData observable-id="file-91011-002">
      <iodef:File>
        <iodef:HashData scope="file-contents">
          <iodef:Hash>
            <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha1"/>
              <ds:DigestValue>
                MHg3MjYzRkUwRDNBMDk1RDU5QzhFMEM4OTVBOUM1ODVFMzQzRTcxNDFD
              </ds:DigestValue>
            </iodef:Hash>
          </iodef:HashData>
        </iodef:File>
      </iodef:FileData>
      <iodef:FileData observable-id="file-91011-003">
        <iodef:File>
          <iodef:HashData scope="file-contents">
            <iodef:Hash>
              <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#md5"/>
                <ds:DigestValue>
                  MHg0M0NEODUwRkNEQURFNDMzMEE1QkVBNkYxNkVFOTcxQw==
                </ds:DigestValue>
              </iodef:Hash>
            </iodef:HashData>
          </iodef:File>
        </iodef:FileData>
        <iodef:WindowsRegistryKeysModified observable-id="regkey-91011-002">
          <iodef:Key registryaction="add-value">
            <iodef:KeyName>
              HKLM\Software\Microsoft\Windows\
              CurrentVersion\Run\KB00121600.exe
            </iodef:KeyName>
            <iodef:Value>
              \?\\%AppData%\KB00121600.exe\?\?
            </iodef:Value>
          </iodef:Key>
        </iodef:WindowsRegistryKeysModified>
      </iodef:RecordData>
    </iodef:Record>
  </iodef:EventData>
</iodef:IndicatorData>

```

```

<iodef:Indicator>
  <iodef:IndicatorID name="csirt.example.com" version="1">
    ind-91011
  </iodef:IndicatorID>
  <iodef:Description>
    evil c2 server, file hash, and registry key
  </iodef:Description>
  <iodef:IndicatorExpression operator="or">
    <iodef:IndicatorExpression operator="or">
      <iodef:Observable>
        <iodef:Address category="site-uri" observable-id="addr-grst">
          http://foo.example.com:12345/evil/cc.php
        </iodef:Address>
      </iodef:Observable>
      <iodef:Observable>
        <iodef:Address category="ipv4-addr" observable-id="addr-stuv">
          192.0.2.1
        </iodef:Address>
      </iodef:Observable>
      <iodef:Observable>
        <iodef:Address category="ipv4-addr" observable-id="addr-tuvw">
          198.51.100.1
        </iodef:Address>
      </iodef:Observable>
      <iodef:Observable>
        <iodef:Address category="ipv6-addr" observable-id="addr-uvwx">
          2001:db8:dead:beef::1
        </iodef:Address>
      </iodef:Observable>
      <iodef:ObservableReference uid-ref="addr-c2-91011-001"/>
      <iodef:ObservableReference uid-ref="addr-c2-91011-002"/>
      <iodef:ObservableReference uid-ref="addr-c2-91011-003"/>
    </iodef:IndicatorExpression>
    <iodef:IndicatorExpression operator="and">
      <iodef:Observable>
        <iodef:FileData observable-id="file-91011-000">
          <iodef:File>
            <iodef:HashData scope="file-contents">
              <iodef:Hash>
                <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmle
nc#sha256"/>
                <ds:DigestValue>
                  141accecc23e7e5157de60853cb1e01bc38042d08f9086040815300b7
fe75c184
                </ds:DigestValue>
              </iodef:Hash>
            </iodef:HashData>
          </iodef:File>
        </iodef:FileData>
      </iodef:Observable>
    </iodef:IndicatorExpression>
  </iodef:IndicatorExpression>
</iodef:Indicator>

```

```

    <iodef:Observable>
      <iodef:WindowsRegistryKeysModified observable-id="regkey-91011-000
">
        <iodef:Key registryaction="add-key"
          observable-id="regkey-vwxy">
          <iodef:KeyName>
            HKLM\SYSTEM\CurrentControlSet\
            Services\.Net CLR
          </iodef:KeyName>
        </iodef:Key>
        <iodef:Key registryaction="add-key"
          observable-id="regkey-wxyz">
          <iodef:KeyName>
            HKLM\SYSTEM\CurrentControlSet\
            Services\.Net CLR\Parameters
          </iodef:KeyName>
          <iodef:Value>
            \"\"%AppData%\KB00121600.exe\"\"
          </iodef:Value>
        </iodef:Key>
        <iodef:Key registryaction="add-value"
          observable-id="regkey-xyza">
          <iodef:KeyName>
            HKLM\SYSTEM\CurrentControlSet\Services\
            .Net CLR\Parameters\ServiceDll
          </iodef:KeyName>
          <iodef:Value>C:\bad.exe</iodef:Value>
        </iodef:Key>
        <iodef:Key registryaction="modify-value"
          observable-id="regkey-zabc">
          <iodef:KeyName>
            HKLM\SYSTEM\CurrentControlSet\
            Services\.Net CLR\Parameters\Bar
          </iodef:KeyName>
          <iodef:Value>Baz</iodef:Value>
        </iodef:Key>
      </iodef:WindowsRegistryKeysModified>
    </iodef:Observable>
  </iodef:IndicatorExpression>
  <iodef:IndicatorExpression operator="or">
    <iodef:IndicatorExpression operator="and">
      <iodef:ObservableReference uid-ref="file-91011-001"/>
      <iodef:ObservableReference uid-ref="regkey-91011-001"/>
    </iodef:IndicatorExpression>
    <iodef:IndicatorExpression operator="and">
      <iodef:IndicatorExpression operator="or">
        <iodef:ObservableReference uid-ref="file-91011-002"/>
        <iodef:ObservableReference uid-ref="file-91011-003"/>
      </iodef:IndicatorExpression>
    </iodef:IndicatorExpression>
  </iodef:IndicatorExpression>

```

```

        <iodef:ObservableReference uid-ref="regkey-91011-002"/>
      </iodef:IndicatorExpression>
    </iodef:IndicatorExpression>
  </iodef:IndicatorExpression>
</iodef:Indicator>
</iodef:IndicatorData>
</iodef:Incident>
</IODEF-Document>

```

B.5. IoT Malware

The IoT Malware test exchanged information that described a bad IP address of IoT malware and its scanned ports. This example information is extracted from alert messages of a Darknet monitoring system referred in [I-D.ietf-mile-implementreport]. The IODEF version used for the data representation was based on [RFC7970].

```

<?xml version="1.0" encoding="UTF-8"?>
<IODEF-Document version="2.00"
  xmlns="urn:ietf:params:xml:ns:iodef-2.0"
  xmlns:iodef="urn:ietf:params:xml:ns:iodef-2.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <iodef:Incident purpose="reporting">
    <iodef:IncidentID name="csirt.example.com">
      189802
    </iodef:IncidentID>
    <iodef:ReportTime>2017-03-01T01:15:00+09:00</iodef:ReportTime>
    <iodef:GenerationTime>2017-03-01T01:15:00+09:00</iodef:GenerationTime>
    <iodef:Description>IoT Malware and related indicators</iodef:Description>
    <iodef:Assessment occurrence="potential">
      <iodef:SystemImpact severity="medium" type="takeover-system">
        <iodef:Description>IoT Malware is scanning other hosts
        </iodef:Description>
      </iodef:SystemImpact>
    </iodef:Assessment>
    <iodef:Contact role="creator" type="organization">
      <iodef:ContactName>example.com CSIRT
      </iodef:ContactName>
      <iodef:Email>
        <iodef:EmailTo>contact@csirt.example.com
        </iodef:EmailTo>
      </iodef:Email>
    </iodef:Contact>
    <iodef:EventData>
      <iodef:Discovery source="nids">
        <iodef:Description>
          Detected by darknet monitoring
        </iodef:Description>

```



```
</iodef:Discovery>
<iodef:Flow>
  <iodef:System category="source">
    <iodef:Node>
      <iodef:Address category="ipv4-addr">
        192.0.2.210
      </iodef:Address>
    </iodef:Node>
    <iodef:NodeRole category="camera"/>
    <iodef:Service ip-protocol="6">
      <iodef:Port>23</iodef:Port>
    </iodef:Service>
    <iodef:OperatingSystem>
      <iodef:Description>
        Example Surveillance Camera OS 2.1.1
      </iodef:Description>
    </iodef:OperatingSystem>
  </iodef:System>
</iodef:Flow>
<iodef:EventData>
  <iodef:Flow>
    <iodef:System category="target">
      <iodef:Node>
        <iodef:Address category="ipv4-addr">
          198.51.100.1
        </iodef:Address>
      </iodef:Node>
      <iodef:NodeRole category="honeypot"/>
      <iodef:Service ip-protocol="6">
        <iodef:Port>23</iodef:Port>
      </iodef:Service>
    </iodef:System>
  </iodef:Flow>
</iodef:EventData>
<iodef:EventData>
  <iodef:Flow>
    <iodef:System category="target">
      <iodef:Node>
        <iodef:Address category="ipv4-addr">
          198.51.100.94
        </iodef:Address>
      </iodef:Node>
      <iodef:NodeRole category="honeypot"/>
      <iodef:Service ip-protocol="6">
        <iodef:Port>23</iodef:Port>
      </iodef:Service>
    </iodef:System>
  </iodef:Flow>
</iodef:EventData>
```

```
</iodef:EventData>
<iodef:EventData>
  <iodef:Flow>
    <iodef:System category="target">
      <iodef:Node>
        <iodef:Address category="ipv4-addr">
          198.51.100.237
        </iodef:Address>
      </iodef:Node>
      <iodef:NodeRole category="honeypot"/>
      <iodef:Service ip-protocol="6">
        <iodef:Port>2323</iodef:Port>
      </iodef:Service>
    </iodef:System>
  </iodef:Flow>
</iodef:EventData>
</iodef:EventData>
</iodef:Incident>
</IODEF-Document>
```

Authors' Addresses

Panos Kampanakis
Cisco Systems

Email: pkampana@cisco.com

Mio Suzuki
NICT
4-2-1, Nukui-Kitamachi
Koganei, Tokyo 184-8795
JP

Email: mio@nict.go.jp