X-ARF: A Reporting and Exchange Format for the Data Exchange of Netflow and Honeypot Data

Jan Kohlrausch, Sven Übelacker, GÉANT 3 JRA2 T4: Internal deliverable
DFN-CERT Services GmbH
Hamburg, Germany
Email: {kohlrausch,uebelacker}@dfn-cert.de

August 2, 2011

Abstract

X-ARF is a light-weight but structured format for the exchange of computer security incident and other attack data. In contrast to other formats like the "Incident Object Description and Exchange Format" (IODEF) the format is kept as simple as possible. The advantage of X-ARF is that it can be used for multiple tasks. Its main intention is to report security incidents to the abusive sites. However, the format is flexible and can be easily extended to, for example, include malware samples. A X-ARF message contains human as well as machine readable containers. Therefore, the same message can be used to inform the administrator of an abusive system about the incident and it can be processed automatically by an incident management system without changes.

The existing X-ARF report types are designed for single incidents reducing the efficiency when a large number of reports is sent simultaneously. For that reason, we propose two novel alternative X-ARF types to aggregate multiple incidents into a single X-ARF report and propose to test their applicability in the GÉANT 3 JRA2 Task 4.

1 Introduction and Motivation

A large number of attacks on the Internet affect systems located in different countries. Handling such attacks requires a collaboration between these sites that does not stop at national borders. For that reason, the GÉANT 3 project aims to establish an infrastructure that enables the exchange of data of computer security incidents. Technically, a format has to be designed that supports the data exchange between the collaborating sites.

Data exchange formats have been proposed that consider different specific requirements. However, most of these formats suffer from individual disadvantages. Some formats like IODEF and IDMEF are structured to specify an unambiguous syntax and semantics of the data. Although these formats support an automatic processing, they are not suited for manual handling. Other formats have been proposed based on a textual structure. However, there is no formal syntax that specify the structure of the data. This prevents an automatic processing of the data.

Recently, a new format called X-ARF was proposed that alleviates these drawbacks. Technically, a X-ARF message consists of different containers where the first contains textual content, the second a structured YAML-document, and the third is intended to contain log data. Thus, X-ARF is a light-weight but structured format for the exchange of data related to computer security incidents. In contrast to other formats like the "Incident Object Description and Exchange Format" (IODEF) the format is kept as simple as possible. The idea of X-ARF is to focus on the most relevant information and to add extensibility to adapt it to special needs. The format is not limited to incident data and can be additionally used

to exchange malware, honeypot, or IDS data. A X-ARF message contains human as well as machine readable containers. Therefore, the same message can be used to inform the administrator of an abusive system about the incident and it can be automatically processed by an incident management system without changes. Currently, the format is supported by a growing list of CERTs and a broad acceptance in the academic community can be expected. Because of its advantages we propose to use this format in GÉANT 3 JRA2 T4 to exchange incident data collected by netflow sensors and honeypots.

The remainder of this paper is organised as follows. After a summary of the relevant previous work, the X-ARF format is introduced. The third section of this paper also gives a summary of the applications of this format. In the fourth section two alternative new X-ARF types are proposed that perform better for bulk reports. Both types aggregate incident data containing multiple different targets which results in a single report.

2 Previous Work

Previous work on exchange formats relevant to computer security incidents focused on different aspects. Some formats concentrate on the ability to automatically parse and process the data. Usually, these approaches define a structured machine-readable format. The Intrusion Detection Message Exchange Format (IDMEF), for example, is specified by a XML DTD. The format focuses on the transfer of data gathered by intrusion detection systems (IDS). A lot of well-known IDS support this format. Most of them are published under the GPL or a similar licence for open source programs. Most widespread IDS are Snort and the Prelude framework. On the one hand, this format is very powerful and forms a quasi standard for IDS. On the other hand, the format is not human readable and designed to transfer IDS results which limits its applicability to exchange security incident data.

To address these limitations, the Incident Object Description Exchange Format (IODEF) was developed that evolved from IDMEF. Focus of this format is the exchange of data related to computer security incidents such as scanning activities and compromises between CERTs. To consider constraints on the usage and distribution of incident data, IODEF contains special XML entities that specify the intended usage. In addition, the format contains some specific information about the expectation how the data is used and other information that are relevant for handling the incident. Although, IODEF is supported by a working group, its complexity and the lack of ready-to-use software prevented a widespread usage so far.

In the past Computer Security Incident Response Teams reported compromises or other incident related data in an informal textual form. While this is feasible for a low number of incidents that are manually processed, it becomes impractical for the actual automated attacks. Therefore, a lot of ad-hoc proprietary formats were introduced. However, these formats typically lack a standardised structure which prevents a large-scale distribution.

The X-ARF formats tries to combine the best properties from the previous approaches. On the one hand, the format is structured based on a formal specification. On the other hand, it is light-weight and supported by publicly available tools. X-ARF messages consist of a variable number of containers. This allows to combine different parts that are human as well as machine readable. For that reason, X-ARF messages can be simultaneously used to report to human as well as to transfer data that is automatically handled and interpreted by programs. So far, the format is applied by different sites to report security incidents. For example, the abusix Project is actively supporting this format.

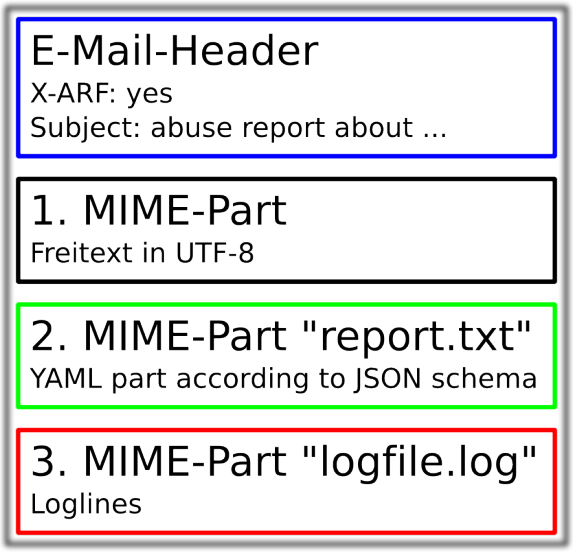

Figure 1: Structure of a X-ARF report

3 Introduction to the X-ARF Format

X-ARF documents are structured in multiple parts denoted as container. Each container is structurally independent of the other and may contain completely different content. However, the idea is to combine human readable and machine readable parts which contain the same or at least similar information. Therefore, the same X-ARF documents is addressed simultaneously to humans, for example, system administrators and allows an automated processing.

As shown in Fig. 1 all specified use-cases consist typically of three containers where the third part is optional. However, this number may vary for future specifications. Since X-ARF documents are transferred by e-mail, each container has a specific MIME-type (Content-Type) and a specification of the character encoding (charset) which is usually a UTF-8 encoding. Currently, three different use-cases exist that share the following container:

- The first container is human readable and can contain arbitrary text. Its MIME type is typically “text/plain” encoded in an UTF-8 charset.
- The second container contains data that uses the YAML markup language for structuring the content. A JSON (JavaScript Object Notation) schema exists defining the structure and syntax of the data. This allows to test if a X-ARF document is well-structured and valid in respect of its specification.
- The third container is intended for transferring various additional data regarding to the abuse type that depends on the previous specification in the second container. It can include log-data as a kind of evidence for the abuse handler or it may be used for malicious files that are, for example, captured by a honeypot.

The first container is intended to contain a summary of the message in textual form. The second part contains the details of the attack data. It is structured and can be automatically processed by an incident handling system. The fields contained in the second container depend on the abuse type of the X-ARF document. However, all abuse types share a common set of fields that, for example, contain data about the abusive system. An example report is shown in Appendix A.

3.1 Applications of the X-ARF Format

The focus of the X-ARF format is on the reporting and exchanging of data detailing computer security incidents. Common to all applications are information that, for example, include the IP address of the attacking system. Thus, all applications share a similar JSON schema that specifies the format of the X-ARF document. To address the individual requirements of each abuse type, the corresponding JSON schema contains additional fields that depend on the abuse type.

Reporting Scan Activity and Phishing

A very large fraction of the e-mails sent on the Internet are unsolicited bulk e-mail (Spam) or phishing e-mails. Phishing e-mails are used for fraudulent activity that includes trying to steal bank account information. Usually, the victim is lured connecting to a fake bank server. Very often these web-sites are installed on compromised web-server. The first X-ARF schema specifies the format of reports to sites hosting such a compromised web-server. In addition, the schema addresses other reports for similar fraudulent activity.

A different schema called 'login-attack' is published for reporting SSH account probes. Attacking a SSH server is very promising because of several reasons. A lot of server suffer from user accounts with weak passwords and it is likely that the attacker is able to guess a weak password. In addition, the compromised credentials are often valid on more than one server. As a consequence, the attacker can often abuse the stolen credentials to compromise other server. For these reasons, SSH scans are an important threat on the Internet and an individual schema for these scans is advantageous.

Exchange of Honeypot Data and Malware Samples

A X-ARF schema is proposed for reporting attacks against honeypots such as Nepenthes and other low-interaction honeypots. In addition, information about the malware related to the attack and its detection by anti-virus products can be included in the report. This helps the affected site to detect the malware on the compromised system and to recover from the incident. Although, the intention of this schema is to report attacks, it could be, for example, extended to exchange malware samples. This would enable an information exchange of abuse data between vendors of anti-virus products, academic sites, and CERTs. Although all sites have a different intention how to use the data, they could all benefit from the X-ARF transport format. First, the format is capable of transferring malware files in a structured form. In addition, the same message can be forwarded to the abusive site after stripping the file to report the security incident.

Currently, it is planned to use X-ARF for the European GÉANT 3 project to exchange attack data gathered by different honeypots. These include SSH honeypots that monitors the server for scanning activity. In addition, other honeypots including Dionaea will be extended to produce X-ARF compliant results. Based on these honeypots and the X-ARF exchange format, an open infrastructure will be established to share the attack data. Although the primary aim is to use the data for incident handling services, a collaboration between academic sites and anti-virus vendors could be established from which all sites benefit.

To address the requirements of privacy and data protection the Traffic Light Protocol (TLP) is integrated into the X-ARF format. The TLP is a simple format intended to control the sharing of potential sensitive information. The integration of the TLP in the X-ARF specification allows to control the further dissemination of the message. This is of particular importance for sharing sensitive data.

4 Extension of the X-ARF Format for Bulk Data

X-ARF is intended to exchange data related to computer security incidents. The current schema are designed to report abusive computers that conduct, for example, port-scans or send spam mails. It is important to note, that the existing schema are defined for a single incident (e.g. a port scan, or an abusive host) and accept only one unique destination. While this allows to send reports without any delay to the abusive site, it does not scale reporting a very large number of incidents. The massive amount of mails may result in the exhaustion of resources at the sending as well as receiving site. We believe that this may impede the data exchange between GÉANT NRNs. For that reason, we propose to avoid partitioning of the incident data which is done by sending multiple mails containing data of a single incident. This can be done by aggregating the data before transfering it.

For the exchange of incident data between national research networks (NRN) within the GÉANT project we propose to aggregate multiple reports into a single X-ARF report instead of sending multiple X-ARF mails. Thus, incident data is collected and included in a single aggregated report. We propose to test the applicability of the aggregated reports in the GÉANT 3 JRA2 Task 4.

The X-ARF format can be extended in different ways. The X-ARF specification itself includes a field "Category" that specifies the type of the second MIME part. Its intention is a specify the type of the data that can be, for example, "fraud" for credit card abuse or "abuse" for reporting any kinds of incidents related to viruses, spam, or port scan activity. A category *private* is introduced to address extensions of the format without the need of a formal registration of a type. Thus, this type can be used within a closed group of sites to address individual extensions.

The alternative is to extend the specification of the format itself. For example, a new category "container" could be added to implement aggregation. In contrast to the usage of the category "private" this would require the acceptance of the X-ARF community to add this type to the official X-ARF specification. However, this would enable adding a formal specification that is supported by an open community not be limited to the GÉANT project partners. In the following, both alternatives are discussed.

4.1 Extension of X-ARF by Using the Private Category

A category *private* is introduced in the X-ARF specification to address extensions of the format without the need of a formal registration of a type. This allows using a private JSON schema that defines the YAML content of the second MIME part. The actual X-ARF JSON schema are flat enumerations of key-value fields and their data type. For example, the field "Destination" contains the IP address of the abusive computer. However, JSON allows to specify nested lists of key-value fields. Thus, this enables to implement the aggregation by defining a list that contains an arbitrary number of the previous flat enumerations of key-values field. This extension only affects the second MIME part of X-ARF reports. Thus, the X-ARF report still consists of previously defined parts while the aggregated data is stored exclusively in the second part. An example report is shown in Appendix B.

4.2 Extension of X-ARF by Introducing a Container Category

The alternative to the usage of the "private" category is to add a new category "container" that extends the number of MIME parts in a X-ARF report. Instead of the two or three parts as defined in the existing X-ARF schema, the new type allows an arbitrary number of MIME parts that depends on the number of events.

In analogy to the existing X-ARF types, X-ARF reports of the new type begin with a textual MIME part. The second MIME part contains a list of references to unique events included in the report. In this context an event is understood as an incident related to a

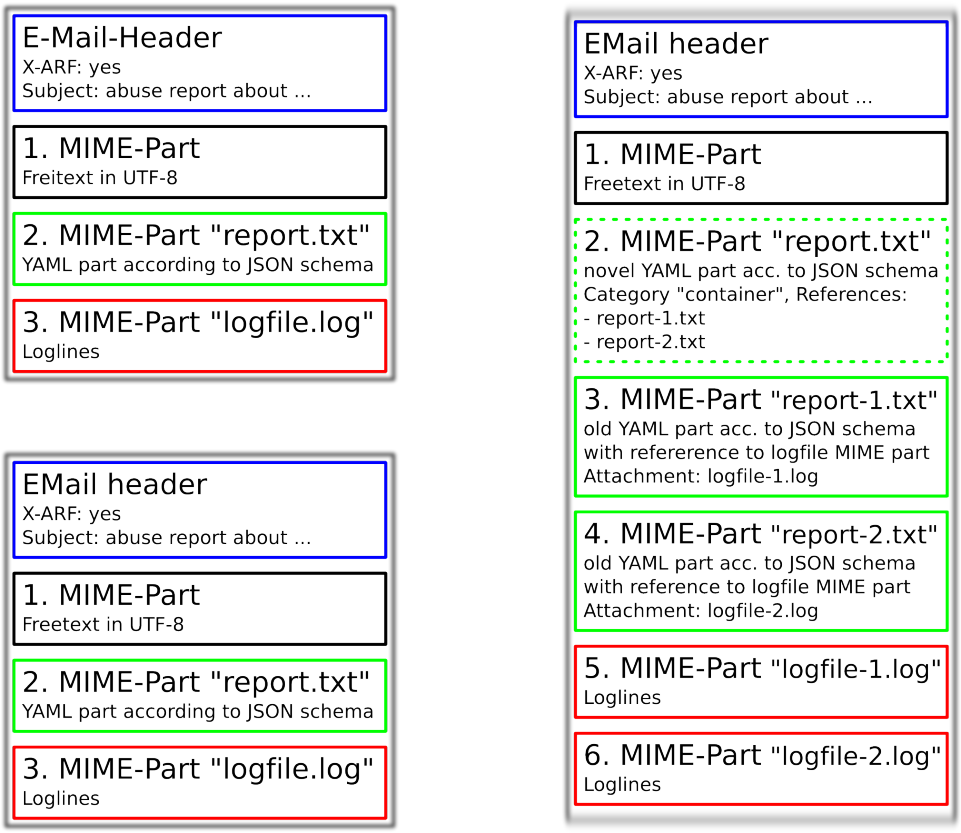

Figure 2: Extension of X-ARF by Introducing a Container Category

unique source and destination. The structure of this MIME part is defined by a new JSON schema that is specific to the "container" type. For each unique event two MIME parts are added to the X-ARF report. These correspond to the second and third MIME part in the existing X-ARF types for single IP addresses. The first contains the structured incident data including the source and destination of the attack. In analogy to the existing X-ARF types it is defined by a JSON schema and contains YAML content. The second MIME part is intended for the log data related to the incident. It is referenced by a field in the first MIME part.

The structure of the "container" type and its relation to the existing X-ARF types is illustrated in Fig. 2. The figure details how two X-ARF reports as shown on the left side are combined into a single report.

5 Summary and Conclusion

In this contribution an overview of the X-ARF format is given. The exchange format is light-weight but structured and intended for the exchange of computer security incident and other attack data. In contrast to other formats is kept as simple as possible.

We propose to use the X-ARF format in GÉANT 3 JRA2 T4 to exchange incident data collected by netflow sensors and honeypots. To increase the effectiveness, two alternative methods for aggregation of X-ARF reports have been introduced. Both methods are advantageous when large quantities of incident data are supposed to be transferred. This is likely to happen between NRN networks.

Aggregation of X-ARF reports can be conducted using two alternative methods which have individual advantages and drawbacks. Both are implemented by using specific X-ARF types. The first alternative uses the “private” type of the X-ARF specification. Aggregation is done by enumerating the different incidents in the second MIME part of the X-ARF report. Thus, like the existing X-ARF types, the aggregated report consists of two MIME parts. The second alternative introduces a completely new “container” type. Two MIME parts are reserved for each incident in the X-ARF report, where the first is a YAML part containing the incident data and the second is intended for the log-data. The X-ARF report consists of a varying number of MIME parts depending on the number of incidents. The advantages and disadvantages are summarised:

Method	Advantage	Disadvantage
Private type	No acceptance of the X-ARF community required	Closed user group
Container type	Open user group	Formal acceptance of X-ARF community required

Appendix A: Sample X-ARF message

Part I (text/plain UTF-8):

```
Dear DFN-CERT,

this is an automated report for ip address 10.11.229.178
in format "X-ARF" generated on 2010-10-22, 10:26:46.044883
IP address 192.168.229.178 produced 1892 log lines, sample
log lines attached.
```

Part II (structured YAML key-value format)

```
---
Category:      abuse
Source-Type:   ipv4
Report-Type:   login-attack
Service:       tsshp 0.7
Report-ID:     12877360064637dfn-cert.de
Reported-From: xxxx@dfn-cert.de
Source:        10.11.229.178
Schema-URL:    http://www.x-arf.org/schema/
                       abuse_login-attack_0.1.1.json
Attachment:    text/plain
Date:          Wed, 13 Oct 2010 12:09:50 +0200
User-Agent:    tsshp 0.7
Port:          22
```

Part III (arbitrary content; recorded ssh connections):

```
2010-10-19 20:33:34,888 WARNING login attempt from:
'10.11.229.178:42658', to: '192.168.39.234:22221',
user: 'root', password: 'XXXXXXXXX', failure #1
2010-10-19 20:33:35,471 WARNING login attempt from:
'10.11.229.178:42658', to: '192.168.39.234:22221',
user: 'root', password: 'XXXXXXXXX', failure #2
--- MARK ---
2010-10-20 21:19:52,473 WARNING blocked attempt from:
'10.11.229.178:55558', to: '192.168.37.233:22221'
2010-10-20 21:19:53,080 WARNING blocked attempt from:
'10.11.229.178:55575', to: '192.168.37.233:22221'
```

Appendix B: Sample X-ARF Container Message Using the Private Type

```
---
Reported-From: xxxx@dfn-cert.de
Category: private
Report-Type: login-attack
User-Agent: xarf-ssh-reporter.bulk.sh
Report-ID: 131058247595300@dfn-cert.de
Date: Wed, 13 Jul 2011 20:41:15 +0200
Source: 127.0.0.1
Source-Type: ipv4
Attachment: none
Schema-URL: http://www.dfn-cert.de/somewhere/private_login-attack_0.0.1.json
Attacker-List:
  -
      ISource: 192.168.244.163,
      ISource-Type: ipv4,
      Service: ssh,
      Port: 22,
      Occurrences: 7,
      TLP: green,
      Loglines:
        - 2011-07-13 20:41:15 +0200 XXXXXX sshd[10437]: Invalid user postgres
                                                        from 192.168.244.163
        - 2011-07-13 20:41:15 +0200 XXXXXX sshd[10441]: Invalid user postgres
                                                        from 192.168.244.163
         - -- MARK --
        - 2011-07-13 20:41:26 +0200 XXXXXX sshd[10541]: Invalid user smbuser
                                                        from 192.168.244.163
        - 2011-07-13 20:41:26 +0200 XXXXXX sshd[10545]: Invalid user smbuser
                                                        from 192.168.244.163
  -
      ISource: 192.168.133.40,
      ISource-Type: ipv4,
      Service: ssh,
      Port: 22,
      Occurrences: 1613,
      TLP: green,
      Loglines:
        - 2011-07-11 04:04:40 +0200 XXXXXX sshd[11178]: Did not receive identification
                                                        string from 192.168.133.40
        - 2011-07-11 05:45:44 +0200 XXXXXX sshd[22313]: User root from 192.168.133.40
                                             not allowed because not listed in AllowUsers
        - 2011-07-11 05:45:44 +0200 XXXXXX sshd[22352]: User root from 192.168.133.40
                                             not allowed because not listed in AllowUsers
        - -- MARK --
        - 2011-07-11 05:50:34 +0200 XXXXXX sshd[26178]: User root from 192.168.133.40
                                             not allowed because not listed in AllowUsers
```