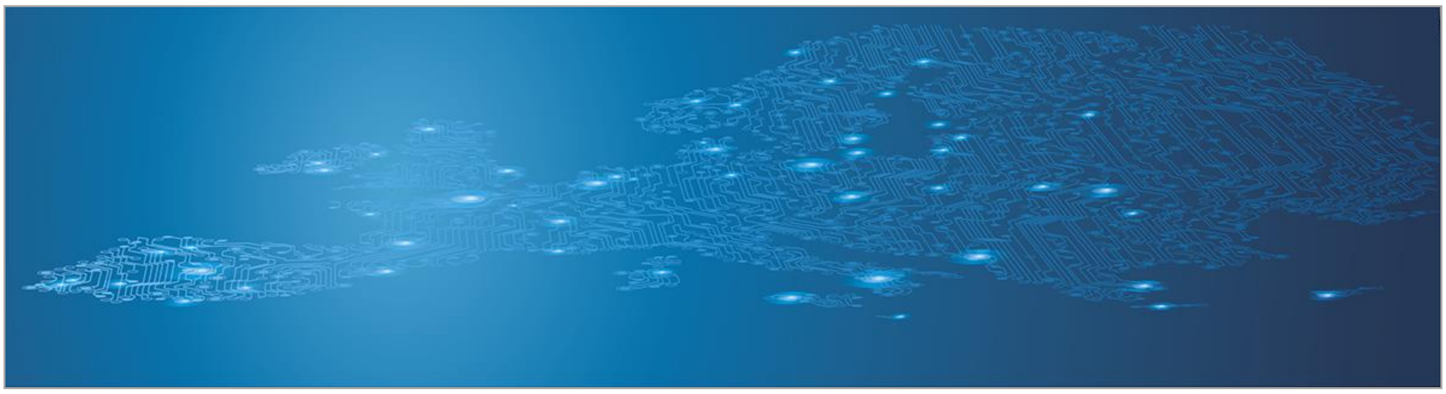
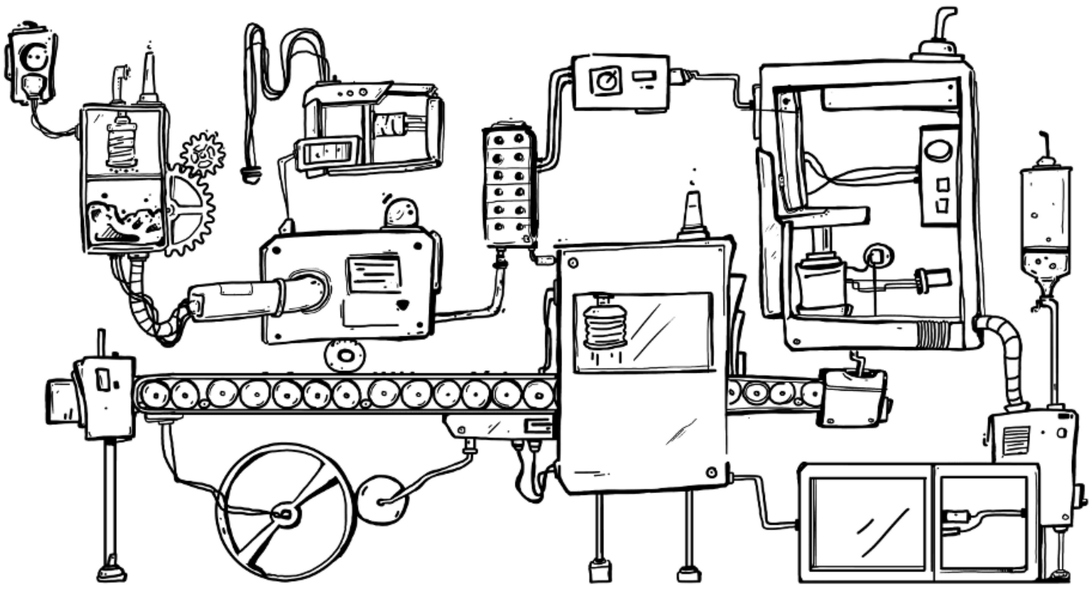




Standards and tools for exchange and processing of actionable information

November 2014

European Union Agency for Network and Information Security www.enisa.europa.eu

About ENISA

The European Union Agency for Network and Information Security (ENISA) is a centre of network and information security expertise for the EU, its member states, the private sector and Europe's citizens. ENISA works with these groups to develop advice and recommendations on good practice in information security. It assists EU member states in implementing relevant EU legislation and works to improve the resilience of Europe's critical information infrastructure and networks. ENISA seeks to enhance existing expertise in EU member states by supporting the development of cross-border communities committed to improving network and information security throughout the EU. More information about ENISA and its work can be found at www.enisa.europa.eu.

Authors :

This document was created by the CERT capability team at ENISA in consultation with CERT Polska /NASK (Poland)[1]

Acknowledgements (in alphabetical order):

- Luc Dandurand (NATO)
- Aaron Kaplan (CERT.at)
- Pavel Kácha (CESNET)
- Youki Kadobayashi (NAIST)
- Andrew Kompanek (CERT/CC)
- Tomás Lima (CERT.PT)
- Thomas Millar (US-CERT)
- Jose Nazario (Invincea)
- Richard Perlotto (Shadowserver)
- Wes Young (CSIRT Gadgets Foundation)

Contact

For contacting the authors please use cert-relations@enisa.europa.eu

For media enquires about this paper, please use press@enisa.europa.eu.

[1] Paweł Pawliński, Przemysław Jaroszewski, Janusz Urbanowicz, Paweł Jacewicz, Przemysław Zielony, Piotr Kijewski (CERT Polska), Katarzyna Gorzelak (CERT Polska)

ISBN: 978-92-9204-105-2 doi: 10.2824/37776 Catalog number: TP-04-14-999-EN-N

Table of Contents

1 Introduction

This document has been created as part of an ENISA-funded study of the state of security information sharing and is intended to supplement the main report, "Actionable Information for Security Incident Response." The purpose of this document is to give the target audience of this study - national and governmental CERTs - a better understanding of the standards and tools for processing actionable information that can be applied to their information-sharing missions.

The first part of the document covers a total of 53 different information sharing standards, a mix of formats, protocols, technical approaches and frameworks in common use. These are broken down into 7 main categories that are based on the scope of the standard:

1. Formats for low-level data
2. Actionable observables
3. Enumerations
4. Scoring and measurement frameworks
5. Reporting formats
6. High level frameworks
7. Transport and Serialization

The document also explores the relationships among the different standards.

The second part of the inventory consists of information management tools that we found relevant to the exchange and processing of actionable information. A total of 16 are listed. These are primarily open source solutions and capabilities that are available to the target group of the study.[2] These solutions are broken down into 4 main categories:

1. Automated distribution of data
2. Supporting analysis
3. General purpose log management
4. Handling High-level information

[2] Rather than attempt to exhaustively survey the vast number of potentially applicable tools, we selected those tools most commonly used operationally by the target audience of this study, national and governmental CERTs.

2 Information Sharing Standards

The use of common standards is important for the exchange of any information, but it can be especially critical when sharing data in a domain as diverse as information security. In particular, data format standards are especially important. A data format standard defines how particular information elements are represented in files or in communications by describing a the syntax of a description language (often based on a generic data format - see Section 3.2)) and, the semantics associated with those descriptions. Using a standard format has two main advantages over using ad hoc representations: implementations that use a standard can take advantage of existing processing tools built to support the standard; and the interpretation of a description should (generally) be less subject to misinterpretation because the standard defines the semantics for information elements.

This inventory lists standards that are relevant to the exchange of *actionable information* - which is understood as information that can be acted upon to prevent or eliminate threats. What constitutes *actionable* varies between stakeholders, however in general it should be usable for the purpose of mitigation with minimal analysis or verification on the part of the final recipient[3]. Therefore the focus of this document is on formats that are used to exchange data[4], although frameworks that specify the exchange process on a more abstract level were also included.

The term "standard" throughout this document is used in a broad sense that encompasses not only specifications published by traditional standards bodies like ISO or ITU but also formats developed by other entities, as long as they are commonly used by information security operations.

The figure on the following page illustrates relationships between all standards for sharing of security information described in this document with the exception of the high-level frameworks (i.e, SCAP and CYBEX), which we omitted simply to reduce clutter since these frameworks reference many of the other standards. It should be read as follows:

- Solid vertical and diagonal lines: lead from standards that embed or reference other standards, e.g. CVRF uses vocabularies defined by CVE and CWE.

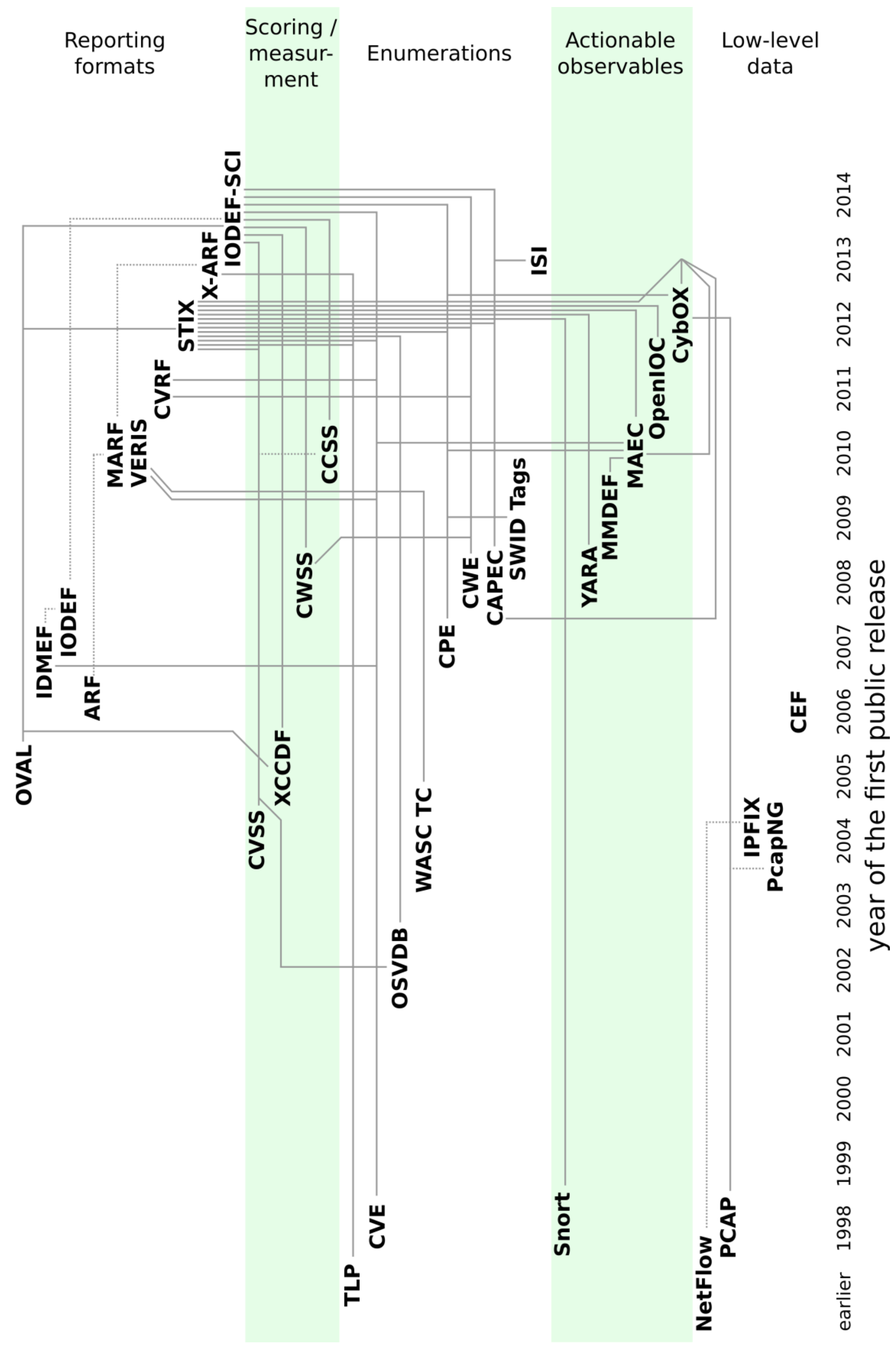
- Dotted vertical and diagonal lines: lead from standards that evolved from others, e.g. X-ARF is an extended variant of MARF.

- All horizontal lines: lead to standards that are used by others any way, e.g. OVAL is used by multiple other standards.

Some of the standards in the diagram reference others that were published later. This is due to the fact that many of the standards incorporated additional references during their evolution.

[3] The "Actionable Information for Security Incident Response" provides more in-depth discussion of what "actionable" for incident response and information security in general.

[4] Information is often defined as more processed and structured, while data as raw and less organized. However, for the purpose of this document these two terms are used interchangeably.


Reporting formats
Scoring / measur-ment
Enumerations
Actionable observables
Low-level data
IODEF-SCI
X-ARF
ISI
STIX
CybOX
OpenIOC
CVRF
MAEC
MARF
VERIS
CCSS
MMDEF
SWID Tags
YARA
CWSS
CWE
CAPEC
IDMEF
IODEF
CPE
ARF
CEF
OVAL
XCCDF
CVSS
WASC TC
IPFIX
PcapNG
OSVDB
Snort
CVE
TLP
PCAP
NetFlow
earlier
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
year of the first public release

Many standards in some way related to sharing security information were omitted. They are listed below for completeness:

- Standards that are no longer developed or maintained and not used by any new tools: Common Digital Evidence Storage Format (CDESF)[5], Common Event Expression (CEE)[6], SECDEF and related standards, Resource-Oriented Lightweight Indicator Exchange (ROLIE)[7].
- Standards that are not used in practice: Real-time Inter-network Defence (RID)[8], Policy Language for Assessment Results Reporting (PLARR)[9], Asset Reporting Format[10] (ARF), Assessment Summary Results, Structured Assurance Case Metamodel (SACM), eXtensible Access Control Markup Language (XACML), Digital Forensics XML (DFXML), CMSS (Common Misuse Scoring System).
- Standards are too generic, not applicable for exchange of actionable information: Common Configuration Enumeration (CCE), Open Checklist Interactive Language (OCIL), Semantics of Business Vocabulary and Business Rules (SBVR), TMSAD (Trust Model for Security Automation Data), SACM (Security Automation and Continuous Monitoring), Common Weakness Risk Analysis Framework (CWRAF)
- Standards under development, driven by CERTs but still lacking adoption - Intrusion Detection Extensible Alert (IDEA) and Data Harmonization Ontology for Abuse Helper.
- Other standards with not relevant for sharing actionable information, in particular Trusted Network Connect (TNC) standards. For ETSI ISI standards we take only the first two (ISI-001-1 and ISI-002) into account, as the rest of the standard deals with aspects of security management that lie beyond the scope of this document.

[5] See http://www.dfrws.org/CDESF/
[6] See https://cee.mitre.org/
[7] See https://tools.ietf.org/html/draft-field-mile-rolie-00
[8] See http://tools.ietf.org/html/rfc6545
[9] See http://measurablesecurity.mitre.org/incubator/plarr/
[10] See http://scap.nist.gov/specifications/arf/

2.1 Formats for low-level data

The formats in this category were developed to represent data collected by security monitoring systems (e.g., on the network level). Most of them have not been formally standardized. Further analysis is generally required to extract useful (i.e., actionable) information. All of formats described in this section use a custom binary serialization format (see Section 3.2.7).

2.1.1 NetFlow

Full name:	NetFlow
Year of publication:	1990
Governing body:	Cisco Systems
Description:	NetFlow is a protocol that was originally developed for exporting traffic summaries in the form of IP flow records from active network devices (i.e., routers, switches) that is now used by many passive flow sensors. It was introduced by Cisco Systems, but similar export features are present in networking equipment from other vendors. Netflow data is produced by a network device or sensor which transfers the data by either UDP or SCTP to a collector that aggregates and organizes the data so that it can be queried and analyzed. A distinction should be made between NetFlow v5 which is based on a fixed record format, and NetFlow v9, which supports the definition of customized record formats.
Realtionships:	NetFlow v9 formed the basis for IPFIX
Types of indicators:	IP packet headers, traffic volumes, routing information (e.g., indexes of switch/router interfaces)
Examples of tools:	SiLK,[11] Argus[12]
Reference:	RFC 3954

2.1.2 IPFIX

Full name:	Internet Protocol Flow Information Export
Year of publication:	2004
Governing body:	IETF
Description:	IPFIX is a standard that evolved from NetFlow that is formalized in an RFC that defines how the flow information should be formatted for export from network devices and sensors ("flow meters"). Like NetFlow records, IPFIX records describe a single logical IP connection corresponding to the 5-tuple in an IP header, generally include fields describing traffic volumes (in bytes and packets) for the connection, but may also include any number of other fields that summarize information about the connection.

[11] See https://tools.netsa.cert.org/silk/
[12] See http://argus.tcp4me.com

Realtionships:	provides a generic framework for describing any flow-like data; based on Cisco NetFlow 9
Types of indicators:	IP packet headers, traffic volumes
Examples of tools:	SiLK, Argus, libIPFIX[13]
Reference:	RFC 3917

[13] See http://libipfix.sourceforge.net

2.1.3 PCAP

Full name:	packet capture file
Year of publication:	1998 (libpcap 0.4)
Governing body:	TCPDUMP project
Description:	PCAP is the format used by many popular packet capture tools, and is used to store or transmit captured network traffic. The format is very simple and allows the storage of time zone, clock accuracy and link type along with the captured network packets.
Realtionships:	used by CybOX
Types of indicators:	network packets
Examples of tools:	libpcap,[14] tcpdump, Snort,[15] Wireshark[16]
Reference:	http://www.tcpdump.org

2.1.4 PcapNG

Full name:	PCAP Next Generation Dump File Format
Year of publication:	2004
Governing body:	Wireshark project
Description:	PcapNg is a format for the storage and transmission of packet traces. The primary design goals of PcapNg are extensibility and portability: the format allows additional descriptive data to attached to PCAP traces. The PcapNg standard defines a format for representing additional capture metadata. This includes facilities for describing the capture device and the filter used at capture time. It also includes provisions for storing NetFlow and Remote Network Monitoring (RMON) data. PcapNg is now defined as an Internet Draft.
Realtionships:	based on PCAP, with extensions for additional metadata for encapsulating IP flow information
Types of indicators:	network packets
Examples of tools:	NTAR,[17] libpcap, Wireshark
Reference:	http://pcapng.com

2.1.5 CEF

Full name:	Common Event Format
Year of publication:	2006

[14] See http://www.tcpdump.org and http://www.winpcap.org
[15] See https://www.snort.org
[16] See https://www.wireshark.org
[17] See http://www.winpcap.org/ntar/

Governing body:	ArcSight Inc.
Description:	CEF is a syslog-based format for the transmission of event information between event producers and consumers. CEF messages contain data about the originating device, event signature, a human readable description, and severity. The specification also includes a dictionary of predefined keys for describing security-related events.
Realtionships:	based on syslog
Types of indicators:	devices involved, event type, network addresses, users, files, network metadata, firewall rules and events, other OS artifacts
Example of tools:	ArcSight SIEM[18]
Reference:	http://mita-tac.wikispaces.com/file/view/CEF+White+Paper+071709.pdf

2.2 Actionable observables

In contrast to unprocessed data, these formats are used to represent certain characteristics of threats (e.g., system libraries used by a malware sample) that have been explicitly identified by the producer of the information. This kind of information is often referred to as "indicators", "detection indicators" or, more narrowly, "indicators of compromise" since it can be used to detect attacks and other malicious activity (e.g., botnet communication).

2.2.1 CybOX

Full name:	Cyber Observable eXpression
Year of publication:	2012
Governing body:	MITRE
Description:	CybOX is a standardized language for representing observables. An observable may be used to detect an event, or check a property of a malware-infected system. Examples of observables that can be described in CybOX include file deletion events, changes to registry keys values and communication via HTTP.
Realtionships:	used in STIX, CAPEC, and MAEC to represent observables; uses PCAP.
Types of indicators:	OS artifacts, APIs, X.509 certificates, network flows, network artifacts, files, SMS messages, images, email messages
Serialization:	XML
Examples of tools:	python-cybox,[19] cybiet
Reference:	http://cybox.mitre.org

[18] See http://www.arcsight.com
[19] See https://github.com/CybOXProject/python-cybox

2.2.2 MAEC

Full name:	Malware Attribute Enumeration and Characterization
Year of publication:	2010
Governing body:	MITRE
Description:	MAEC is a structured language for describing malware behaviors, artifacts, and attack patterns. The MAEC schema defines a standard set of malware attributes that includes simple attributes like name, size, cryptographic hashes and, AV classifications. It also includes ways of describing low-level executable behavior, including descriptions of processes spawned, system calls made, files created, registry keys modified, and network communications. Finally, it can be used to describe malicious behaviors found or observed in malware at a higher level, indicating the vulnerabilities it exploits, behavior like email address harvesting from contact lists or disabling of a security service.
Realtionships:	uses CybOX and MMDEF, used in STIX
Types of indicators:	malware characteristics, malware actions
Serialization:	XML
Examples of tools:	Anubis,[20] ThreatTrack,[21] ThreatExpert,[22] Cuckoo Sandbox,[23] Thug[24]
Reference:	http://maec.mitre.org

2.2.3 MMDEF

Full name:	Malware Metadata Exchange Format
Year of publication:	2009
Governing body:	IEEE
Description:	MMDEF is an XML format allowing for the sharing of malware samples, behavioral information and other metadata. MMDEF was developed to facilitate information exchange in the antivirus industry.
Realtionships:	MMDEF can be used in MAEC to describe malware
Types of indicators:	filenames, file hashes, malware behavior, origin
Serialization:	XML
Examples of tools:	Cuckoo
Reference:	http://standards.ieee.org/develop/indconn/icsg/mmdef.html

[20] See https://anubis.iseclab.org
[21] See http://www.threattracksecurity.com
[22] See http://www.threatexpert.com
[23] See http://www.cuckoosandbox.org
[24] See https://github.com/buffer/thug

2.2.4 OpenIOC

Full name:	Open Indicators of Compromise
Year of publication:	2011
Governing body:	MANDIANT
Description:	OpenIOC is an XML-based language designed to group and communicate forensic information. It is suitable for descriptions of technical characteristics that identify a known threat, an attacker's methodology, or other evidence of compromise. OpenIOC is focused on describing malware artifacts, indicators and attacker TTPs. The format is extensible, allowing the definition of new data types using custom indicator sets.
Realtionships:	used in STIX
Types of indicators:	networking information, browser artifacts, OS artifacts, memory forensic information
Serialization:	XML
Examples of tools:	MANDIANT IOC Editor,[25] Mandiant IOC Finder,[26] OpenIOC-to-STIX[27]
Reference:	http://openioc.org

2.2.5 Snort rules

Full name:	Snort
Year of publication:	1998
Governing body:	Sourcefire / Cisco Systems
Description:	Snort rules are designed for the real-time analysis of network traffic. The IDS uses a set of snort rules to detect and alert on packets that might represent harmful or suspicious network traffic. The rules operate at various layers of OSI model, including elements of IP packet headers, HTTP protocol headers, and patterns in packet payload.
Realtionships:	used in STIX
Types of indicators:	IP addresses, ports, flags, protocol, direction, patterns in payload, HTTP request and response parameters
Serialization:	custom text-based
Examples of tools:	Snort, Suricata[28]
Reference:	http://snort.org

[25] See http://www.mandiant.com/resources/download/ioc-editor/
[26] See http://www.mandiant.com/resources/download/ioc-finder/
[27] See https://github.com/STIXProject/openioc-to-stix
[28] See http://suricata-ids.org

2.2.6 YARA rules

Full name:	Yet Another Regex Analyzer
Year of publication:	2008
Governing body:	none, community engagement coordinated by Víctor Manuel Álvarez
Description:	YARA is a tool and a signature format for the analysis and identification of malware. YARA allows an analyst to write logical expressions based on built-in signature-matching functions to test for malware features.
Realtionships:	used in STIX
Types of indicators:	binary signatures, strings
Serialization:	Custom text-based
Examples of tools:	YARA[29] (software), VirusTotal,[30] jsunpack[31]
Reference:	https://github.com/plusvic/yara

2.3 Enumerations

The term enumeration is used here to signify a standard that is used to define the meaning of a vocabulary used by the security community. That vocabulary is generally either a set of (a) global identifiers for shared data objects that need to be referenced in a common way, or (b) data labels that need to be clearly defined. Some of the standards specify such vocabularies directly or while others provide guidelines for the creation of enumerations for a particular domain but do not actually enumerate the vocabulary.

2.3.1 CAPEC

Full name:	Common Attack Pattern Enumeration and Classification
Year of publication:	2008
Governing body:	MITRE
Description:	CAPEC is a publicly available catalog of attack patterns that includes a description language schema and classification taxonomy. CAPEC entries are descriptions of particular attack patterns, that is, the techniques and procedures used to carry out the sequence of steps that makes up the pattern.
Realtionships:	used in STIX, IODEF-SCI; uses CybOX
Reference:	https://capec.mitre.org

[29] See http://plusvic.github.io/yara/
[30] See https://www.virustotal.com
[31] See http://jsunpack.jeek.org

2.3.2 CPE

Full name:	Common Platform Enumeration
Year of publication:	2007 (ver. 1.1)
Governing body:	NIST
Description:	The CPE standard provides a consistent and structured naming scheme for operating systems, software packages and classes of hardware devices. It is based on the generic syntax for Uniform Resource Identifiers (URIs) and includes a method by which vendors can validate that their product names are accurately represented in the CPE system. CPE identifies abstract classes of products, not specific instances (e.g., it does not include serial numbers). As an example, consider the entry for Internet Explorer ver. 8.0.6001: cpe:/a:microsoft:internet_explorer:8.0.6001.
Realtionships:	included in IODEF-SCI, and used in MAEC and CybOX
Reference:	https://capec.mitre.org

2.3.3 CVE

Full name:	Common Vulnerabilities and Exposures
Year of publication:	1998
Governing body:	MITRE
Description:	CVE is a list of known security vulnerabilities and exposures. The main goal of CVE is to define a standard set identifiers (CVE-ID numbers) that can be used to reference publicly-known vulnerabilities. The CVE list is maintained by MITRE, which publishes the list through the National Vulnerability Database (NVD)[32]. CVE-ID numbers are assigned by CVE Numbering Authorities (CNAs), software vendors and other organizations that have met requirements specified by MITRE.
Realtionships:	used by IODEF-SCI, STIX, CVRF, IDMEF, VERIS; referenced by OSVDB
Reference:	https://cve.mitre.org

2.3.4 CWE

Full name:	Common Weakness Enumeration
Year of publication:	2008
Governing body:	MITRE
Description:	CWE is a list of commonly occurring software weaknesses and vulnerabilities. The primary goal of the CWE project is to avoid introducing vulnerabilities in the first place by educating software developers.

[32] NVD is the registry of CVE with extra information and a search engine.

Realtionships:	used by IODEF-SCI, CVRF, CWSS, STIX
Reference:	http://cwe.mitre.org

2.3.5 ISI

Full name:	Information Security Indicators
Year of publication:	2013
Governing body:	ETSI
Description:	ISI is a security management framework consisting of a family of standards for describing and assessing an organization's security posture. The family includes ISI-001, an approach for the classification of vulnerabilities, and ISI-002, which is used to categorize event and incidents. The ISI framework is meant to provide an organization with measurement approaches for assessing the impact of events and vulnerabilities.
Realtionships:	uses CAPEC
Reference:	http://www.etsi.org/deliver/etsi_gs/ISI/001_099/00101/01.01.01_60/gs_isi00101v010101p.pdf

2.3.6 OSVDB

Full name:	Open Sourced Vulnerability Database
Year of publication:	2002
Governing body:	OSF
Description:	OSVDB is a vendor-independent vulnerability database, managed by a non-profit foundation. An entry consists of a disclosure timeline, description, classification, possible solutions or mitigations, a vulnerable products list, references (including a CVE-ID if available), CVSS score, and credits.
Realtionships:	used by STIX and VERIS; uses CVSS; references CVE
Reference:	http://osvdb.org

2.3.7 SWID Tags

Full name:	Software Identification Tags
Year of publication:	2009
Governing body:	TagVault (ISO/IEC 19770-2:2009)
Description:	The software identification tag (SWID tags) standard defines an XML schema that can used to describe software assets. It records unique information about an installed software application, including its name, edition, version, serial number, whether it is a part of a bundle and more. SWIDs are mostly used to create local inventory of applications installed on computers within an

	organization. This tagging supports compliance and asset management procedures, and can improve reaction times to security issues.
Realtionships:	used in SCAP; can be used to generate CPE names
Reference:	http://tagvault.org/swid-tags/

2.3.8 TLP

Full name:	Traffic Light Protocol
Year of publication:	unknown
Governing body:	US-CERT
Description:	TLP is a simple protocol used to label sensitive information to ensure that only the correct audience is given access to it. The TLP defines a fixed vocabulary of four colours (white, green, amber, red) to indicate information sharing levels. The TLP is used by various organizations, inside and outside of IT security. It may be used to mark the sensitivity of a document, or information element, in any format that supports labels of this sort.
Realtionships:	data classification in STIX and X-ARF, can be informally used to tag any information in human-readable form
Reference:	https://www.us-cert.gov/tlp

2.3.9 WASC TC

Full name:	Web Application Security Consortium Threat Classification
Year of publication:	2004
Governing body:	WASC
Description:	WASC Threat Classification project maintains a dictionary of types of attacks against web applications. The aim of the project is to develop and promote an industry standard of terminology for describing attacks. WASC TC assigns “WASC IDs” to types of attacks so that any given attack methodology can be unambiguously referenced by its WASC ID number.
Realtionships:	used by VERIS
Reference:	http://projects.webappsec.org/w/page/13246978/Threat%20Classification

2.4 Scoring and measurement frameworks

These standards specify frameworks for the quantitative description of threats. They support the decision making process by enabling a more formal analysis of risks than would be possible from an informal description.

2.4.1 CCSS

Full name:	Common Configuration Scoring System
Year of publication:	2010
Governing body:	NIST
Description:	Like CVSS, CCSS defines several groups of scoring metrics (base, temporal and environmental) to help assess an impact of misconfiguration of a particular asset (e.g., the impact of misconfiguration of user rights to invoke certain commands). CCSS is based on the Common Vulnerability Scoring System (CVSS) and can can assist organizations in developing a view of the overall security state of a system.
Realtionships:	based on CVSS; used as a part of SCAP (from ver. 1.2); can be used in IODEF-SCI
Reference:	http://csrc.nist.gov/publications/nistir/ir7502/nistir-7502_CCSS.pdf

2.4.2 CVSS

Full name:	Common Vulnerability Scoring System
Year of publication:	2004
Governing body:	FIRST
Description:	CVSS is a scoring system for describing and rating IT vulnerabilities. CVSS is composed of three groups: base, temporal, and environmental, each of which includes a set of metrics. The base score has the most significant influence on the final score. Scores range from 0 to 10, where 10 corresponds to the most critical vulnerability.
Realtionships:	used by IODEF-SCI, STIX, OSVDB; similar to CWSS
Reference:	http://www.first.org/cvss

2.4.3 CWSS

Full name:	Common Weakness Scoring System
Year of publication:	2008
Governing body:	MITRE
Description:	CWSS is a mechanism for scoring weaknesses discovered in software so that fixes can be prioritized. Conceptually, CVSS and CWSS are very similar: where CVSS is used to score specific vulnerabilities, CWSS is used to score weaknesses that could potentially be exploited. CWSS scores are associated with weaknesses identified on the CWE list, and range from 0 to 100. CWSS 1.0 was released in July 2014 (3 years after version 0.8) and according to the authors future development is uncertain.
Realtionships:	used by IODEF-SCI; uses CWE; similar to CVSS

Reference:	http://cwe.mitre.org/cwss

2.4.4 XCCDF

Full name:	Extensible Configuration Checklist Description Format
Year of publication:	2005
Governing body:	NIST
Description:	The XCCDF standard defines a language for expressing security policy and configuration guidance (e.g., a policy for minimum length of users' passwords). It is typically combined with OVAL, which is used to assess security compliance by performing low-level checks based on the variables and values in an XCCDF policy document. The XCCDF specification also provides a data model and format for storing the output of these checks.
Realtionships:	used in SCAP and can be combined with OVAL; can be used in IODEF-SCI
Reference:	http://scap.nist.gov/specifications/xccdf/

2.5 Reporting Formats

The formats described in this section provide a reporting structure that can be used to capture high-level, comprehensive descriptions of threats. They support reporting that combines multiple types of information, including indicators, affected assets, actions that were taken, and other contextual information, and to that end often include mechanisms for incorporating data represented in other standard formats.

2.5.1 ARF

Full name:	Abuse Reporting Format
Year of publication:	2005
Governing body:	none formal, coordinated by Yakov Shafranovich
Description:	ARF is an extension to MIME developed for email spam reporting. ARF allows the creation of email messages that contain spam reports with spam samples attached. The main part of an ARF specification is the definition of the *message/feedback-report* MIME content type that is intended to represent a machine-readable spam report. In addition to the original spam message, the report contains the source IP of the message, original message ID, the date the message was received and a message classification (abuse, spam, virus, other, non-spam). According to the standard an ARF message should also contain a human-readable version of the report.
Realtionships:	superseded by MARF
Types of indicators:	spam reports and samples
Serialization:	text-based (MIME)

Examples of tools:	Email::ARF,[33] Email::ARF::Report, arffilter
Reference:	http://www.shaftek.org/publications/drafts/abuse-report/

2.5.2 CVRF

Full name:	Common Vulnerabilities Reporting Framework
Year of publication:	2011
Governing body:	ICASI
Description:	CVRF is a data exchange format designed to support the automation of software vulnerability data reporting and consumption. A CVRF document describes the whole vulnerability handling lifecycle, from the discovery of the vulnerability to shipping a patched version of vulnerable software.
Realtionships:	uses CVE and CWE
Types of indicators:	product vulnerabilities
Serialization:	XML
Examples of tools:	none publicly available, used internally in vendors' communications
Reference:	http://www.icasi.org/cvrf

2.5.3 IODEF

Full name:	Incident Object Description Exchange Format
Year of publication:	2007
Governing body:	IETF, Managed Incident Lightweight Exchange (MILE) working group
Description:	Incident Object Description Exchange Format (IODEF) is an XML format for exchanging operational and statistical security incident information. The data model allow the encoding of information about hosts, networks, services, attacks methodologies and forensic data. IODEF was designed for information exchange between CERTs.
Realtionships:	based on, and compatible with IDMEF, extended to IODEF-SCI
Types of indicators:	timing, incident description with confidence rating, network and OS artifacts, exploit and vulnerability references, contact information, incident history
Serialization:	XML
Examples of tools:	Collective Intelligence Framework (CIF),[34] ArcSight
Reference:	RFC 5070

[33] See http://search.cpan.org/~rjbs/Email-ARF/lib/Email/ARF/Report.pm
[34] See https://code.google.com/p/collective-intelligence-framework/

2.5.4 IODEF-SCI

Full name:	IODEF for Structured Cybersecurity Information
Year of publication:	2014
Governing body:	NICT
Description:	IODEF-SCI is a set of IODEF extensions for embedding structured information within an IODEF document. IODEF-SCI uses other formats for the representation of the embedded information, and defines new classes for types of information that are not defined in IODEF.
Realtionships:	uses CAPEC, CVE, CVRF, CCE, CWE, CPE, CVSS, CWSS, CCSS, , OVAL, XCCDF, CRE; based on IODEF
Types of indicators:	attack patterns, platforms, vulnerabilities, weaknesses, scores, event reports, incident remediation description, verification checklists
Serialization:	XML
Examples of tools:	IODEF SCI tools[35]
Reference:	RFC 7203

2.5.5 IDMEF

Full name:	Intrusion Detection Message Exchange Format
Year of publication:	2007
Governing body:	IETF
Description:	IDMEF defines a data and transport model for sharing security event data exported by intrusion detection systems and by event correlation engines.
Realtionships:	uses CVE; base for IODEF
Types of indicators:	IDMEF descriptions includes information about the analyzer itself, timing (analyse/create/detect time), network data about the source and target and an event classification that can include a CVE reference
Serialization:	XML
Examples of tools:	Snort, Prelude,[36] Suricata, OSSEC,[37] Samhain,[38] ArcSight
Reference:	RFC 4765, RFC 4766

2.5.6 MARF

Full name:	Messaging Abuse Reporting Format

[35] See https://github.com/TakeshiTakahashi/IODEF-SCI/wiki/IODEF-SCI-tools
[36] See https://www.prelude-ids.org
[37] See http://www.ossec.net
[38] See http://www.la-samhna.de/samhain/

Year of publication:	2010
Governing body:	IETF
Description:	The MARF is a version of ARF standardized by IETF. The format is an extension to MIME for email spam reporting. MARF allows the creation of email messages that contain spam reports with spam samples attached. In addition to supporting the reporting of spam, MARF can be used to report DKIM, SPF, and SMTP authentication failures.
Realtionships:	based on ARF; extended by X-ARF
Types of indicators:	spam reports and samples
Serialization:	text-based (MIME)
Examples of tools:	Email::ARF, Email::ARF::Report, arffilter
Reference:	RFC 5965, RFC 6430, RFC 6590, RFC 6650, RFC 6651, RFC 6652, RFC 6692

2.5.7 OVAL

Full name:	Open Vulnerability and Assessment Language
Year of publication:	2005 (version 3, previous releases were significantly different)
Governing body:	MITRE
Description:	OVAL is a language for specifying automated tests of system configurations and defines the format for the results of such assessments. Vendors include OVAL specifications in vulnerability advisories to share information about vulnerabilities and misconfigurations in a machine-readable format. OVAL may also be used to distribute descriptions of threat indicators.
Realtionships:	used as a part of SCAP, STIX, IODEF-SCI
Types of indicators:	information about vulnerabilities, configuration policies, threat indicators like information about modified registry keys, etc.
Serialization:	XML

Examples of tools:	Tripwire Enterprise,[39] OpenVAS,[40] SAINT[41]
Reference:	https://oval.mitre.org

2.5.8 STIX

Full name:	Structured Threat Information eXpression
Year of publication:	2012
Governing body:	MITRE, DHS
Description:	STIX is a language for describing a wide range of security-related information. Its data model is built upon eight principal concepts: observables, indicators, incidents, TTPs[42], exploit targets, campaigns, threat actors and course of actions. STIX use cases include: analyzing threats, specifying indicators for threats, managing prevention and response activities, and sharing threat information.
Realtionships:	STIX uses CybOX, MAEC, CAPEC, CVRF, OVAL, Snort and YARA signatures, CVSS, OSVDB, TLP, CPE, CWE, CAPEC, OpenIOC
Types of indicators:	IP addresses and ranges, e-mail messages, files, DNS domains, URLs, malware artifacts, C&C activity, anonymous activity, malicious hosts, data exfiltration activity, compromised PKI certificates, compromised login credentials, IMEI and IMSI numbers
Serialization:	XML
Examples of tools:	Microsoft Interflow,[43] CRITs,[44] MANTIS,[45] python-stix[46]
Reference:	http://stixproject.github.io

2.5.9 VERIS

Full name:	Vocabulary for Event Recording and Incident Sharing Framework
Year of publication:	2010
Governing body:	Verizon
Description:	VERIS is a format used in Verizon's yearly "Data Breach Investigation Report" for defining and sharing incident information. VERIS also provides a set of categories and metrics designed to provide a common language for describing security incidents in a structured form that is used to characterize trends within industry sectors.

[39] See http://www.tripwire.com/it-security-software/scm/tripwire-enterprise/
[40] See http://www.openvas.org
[41] See http://www.saintcorporation.com
[42] Tactics, Techniques and Procedures
[43] See http://www.microsoft.com/interflow
[44] See https://crits.github.io
[45] See section 4.2.5
[46] See https://github.com/STIXProject/python-stix

Realtionships:	CVE, WASC TC
Types of indicators:	IPs, URLs, malware hashes, attack vectors, victim characteristics
Serialization:	JSON
Examples of tools:	none publicly available
Reference:	http://www.veriscommunity.net

2.5.10 X-ARF

Full name:	Extended Abuse Reporting Format
Year of publication:	2013
Governing body:	Abusix
Description:	X-ARF is an extension to ARF/MARF intended to enable reporting of other types of abuse incidents. The extension allows the reporting of login attacks, fraud (phishing), malware and malicious domains. X-ARF messages can be encrypted and cryptographically signed.
Realtionships:	based on MARF; uses TLP
Types of indicators:	IP addresses, domain names
Serialization:	text-based (MIME)
Examples of tools:	abusehq.com[47]
Reference:	https://github.com/abusix

2.6 High-level frameworks

This section covers standards for process frameworks for exchanging security information. Neither of the two standards in this section directly define new data standards. Instead, they outline generic frameworks for interoperability and automation that leverage formats and protocols defined by other standards.

2.6.1 CYBEX

Full name:	Cybersecurity Information Exchange, Recommendation ITU-T X.1500
Year of publication:	2011
Governing body:	ITU-T
Description:	Recommendation ITU-T X.1500 describes techniques for the exchange of security information. Its includes guidance in on several key functions related to information exchange: structuring security information, identifying security information and entities; establishment of trust between entities; requesting and responding with security information; and assuring the integrity of the security information exchange.

[47] See https://abusehq.abusix.com

Realtionships:	CYBEX consists of 29 formats and protocols, including the following: ARF, CAPEC, CPE, CVE, CVSS, CWE, CWSS, IODEF, MAEC, OVAL, SCAP, XCCDF
Examples of tools:	cybiet
Reference:	ITU-T X.1500 series documents

2.6.2 SCAP

Full name:	Security Content Automation Protocol
Year of publication:	2010
Governing body:	NIST
Description:	SCAP is a process framework for the automation of security procedures related to vulnerability management, measurement, and remediation. The SCAP standard is the result of a community-driven synthesis of a number of open security standards and protocols. The framework defines methods for enumerating and assessing software weaknesses and vulnerabilities, and for the automation of policy compliance evaluation.
Realtionships:	SCAP defines 11 formats as its components, including the following: CCE, CPE, CVE, CVSS, OVAL, XCCDF, CCSS
Examples of tools:	Intel Policy Auditor, OpenSCAP,[48] CIS-CAT[49]
Reference:	http://scap.nist.gov

[48] See http://www.open-scap.org
[49] See http://benchmarks.cisecurity.org/downloads/audit-tools/

3 Transport and Serialization

By adopting one of the standard formats described in the previous chapter, an organization can minimize ambiguity in the information, while also benefiting from the tools that support exchange using those standards. Nevertheless, there are other important technical considerations for an information exchange implementation, in particular, the transport mechanisms that are used to query, request and transfer data. Also, when using custom (i.e., non-standard) data formats, the choice of a serialization method can have significant consequences for overall performance and ease of integration with existing tools.

3.1 Transport mechanisms

This section describes some of the most common mechanisms for the transport of actionable information. In principle all of these mechanisms can be used to transport arbitrary data formats (see Section 3.2), however from a practical point of view various technical limitations (e.g., maximum message size) make some combinations infeasible (e.g., sending a large XML document through Twitter). Additionally, binary data usually has to be encoded (e.g., in base64), unless the underlying protocol supports transferring this kind of payload verbatim.

The security of the whole information exchange process depends largely on the security of the underlying transport mechanism, therefore it is crucial that appropriate measures are taken to protect this layer. There are multiple approaches to this problem - TLS with PKI is commonly used to ensure integrity of information,[50] while encryption combined with some authentication scheme (e.g., API keys or passwords) provides confidentiality. Many transport mechanisms do not explicitly address availability, although certain technologies are inherently more resilient (e.g., ones that allow easy replication of resources) than others.

3.1.1 Static files over HTTP

Serving files over HTTP[51] is by far the most popular approach for transferring data.[52] It is generally straightforward to publish files to a web site and expose those files through HTTP, requiring no special infrastructure beyond a web server. Data files can be published with the same ease as web pages. HTTP is the simple text-based protocol that powers the web, and can be used to transfer information (including binary data) between a client and a server, most notably a web browser and a web server. An HTTP server can be set up easily and most common web server applications like Apache[53] or Nginx[54] are ready to use almost immediately after installation. HTTP is a request-response stateless synchronous protocol which defines several methods to indicate actions to be performed on an identified resource and is able to transport large files. An HTTP resource, which can be a static file or dynamically generated content is referred to by its Uniform Resource Locator (URL), which can function as a unique identifier for a logical chunk of information.

Sharing data as files served by web servers works best for publishing static datasets. The HTTP protocol does not provide a built-in mechanism for filtering requested data, which means a client must always download a whole resource. The advantage of this sharing method is the ease of implementation.

[50] The terms "confidentiality", "integrity" and "availability" are used in accordance with the CIA triad model (http://www.albany.edu/acc/courses/ia/classics/clark87.pdf).
[51] See https://tools.ietf.org/html/rfc2616
[52] See http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.168.5917&rep=rep1&type=pdf
[53] See http://httpd.apache.org
[54] See http://nginx.org

Sharing data is as simple as generating a file and placing it in a path accessible for the web server, and access requires nothing more than an HTTP client.

HTTP-based access may be secured using a username and password (possibly as part of a multi-factor authentication scheme), or using certificate-based authentication. Access controls can be tailored for each resource, distinguished by a unique URL. The SSL/TLS protocol (i.e., HTTPS) can be enabled for confidentiality and data integrity.

Sharing static files over HTTP is actively used by security data clearinghouses (e.g., both Shadowserver Foundation and Team Cymru), large providers of security information (e.g., Spamhaus) and other big vendors and communities, including EmergingThreats, Abuse.ch, DShield.

3.1.2 Email

Email messages can carry any type of data, including text and binary files, which allows them to be used for exchange of security related information represented in various formats. Typically, the email body carries some free-form text description of data, which is attached as a file that uses a structured file format. Email messages are transferred using SMTP, which is an asynchronous protocol. It should be noted that email is not an efficient way to share large datasets. Also, large files sent via email are often blocked by mail servers in order to protect them from abuse of storage and processing resources. In fact, email transport cannot be considered 100% reliable as messages are commonly scanned for spam and malware and filtering methods are never completely reliable. This fact should be kept in mind when email is used to send important alerts and announcements.

SMTP is one of the oldest Internet protocols and is implemented in many tools and libraries. The implementation of a system receiving or sending events - incident reports, alerts - should in most cases be fairly easy. Mail delivery agents (MDA) such as procmail[55] or maildrop[56] can be used to simplify integration and perform filtering of received messages, which in later steps can be processed by automated systems or delivered to user accounts where information can be easily accessed using an email client.

SMTP does not provide a mechanism for secure transport which means messages must be secured using other methods, typically S/MIME[57] or OpenPGP.[58]

Email is used in systems used by national and governmental CERTs, including RTIR, Megatron, AbuseHelper, and by major security organizations like Google.[59]

3.1.3 RESTful interface

An interface conforming to the REST (REpresentational State Transfer) architectural style, also called a RESTful interface, provides a uniform way to create, read, update and delete resources. RESTful interfaces are gaining popularity thanks to their simplicity, which differentiates them from some older technologies like SOAP and other standards for implementing remote procedure calls over a network.

Communication between a client and a server via a RESTful interface is request-response based and stateless. Like HTTP, resources are identified via their URIs. Services using RESTful interfaces typically support additional operations on resources, for example, searching or filtering and consequently provide more flexible interfaces to data than is possible by simply hosting a static file on an HTTP

[55] See http://www.procmail.org
[56] See http://www.courier-mta.org/maildrop/
[57] See http://tools.ietf.org/html/rfc5751
[58] See http://tools.ietf.org/html/rfc4880
[59] Google Safe Browsing Alerts for Network Administrators: http://www.google.com/safebrowsing/alerts/

server. There are many frameworks for developing applications that expose RESTful interfaces. Popular frameworks include Django,[60] Flask,[61] Apache Wink[62] and Symfony.[63]

The most common implementation of the REST style is on top of the HTTP protocol. HTTP RESTful interfaces can be secured the same way as any HTTP-based service. Additionally, access to the interface can be allowed only for clients which hold valid API key.

The popularity of REST continues to grow - many familiar web services provide RESTful APIs — and most of modern systems for data exchange offer it, including CRITs, MISP, and n6.

3.1.4 Farsight SIE

Farsight Security Information Exchange (SIE) is a platform that was developed specifically for sharing large volumes of security information in real time. Using the platform, information producers can upload data feeds from sensors they operate. Consumers can receive information from the platform by subscribing to channels that contain live streams with data from passive DNS, darknet traffic, firewall and IDS alerts, Conficker sinkhole, as well as data related to spam and phishing.

Typically, receiving the data requires a server connected to the broadcast ethernet switch within the Farsight Security data center. The exchange of data happens over tagged VLANs configured on this network. Each channel is presented as a data stream of UDP datagrams on an independent VLAN, allowing consumers to select the ones containing information they wish to receive. This design allows the efficient sharing of data at rates reaching 300 Mbps. The data is shared using the NMSG[64] binary format, which is based on Google’s Protocol Buffers.[65]

Data feeds can be also accessed through the Farsight SIE Remote Access service.[66] The service allows consumers to obtain data feeds over an SSH tunnel without the needing to place any hardware in Farsight’s data center. The protocol allows the selection of SIE channels, and provides a way to rate limit traffic. It also allows filtering, which is especially important when accessing high volume SIE channels. The protocol is lossy due to the rate limiting of data transfers, but the service notifies users about loss. This allows users to adapt to losses by, for example, adding more strict filtering parameters.

3.1.5 Twitter

Twitter is an online social networking and microblogging service that enables users to send and read short 140-character text messages ("tweets"). Registered users can read and post tweets, but unregistered users can only read them. Twitter resources can be accessed through the website interface, SMS, mobile device app, RESTful API[67] (publishing and requesting information) or Streaming API[68] receiving a low-latency stream of Tweet data). Twitter is a great tool for sharing short messages like alerts and notifications with links to external sources of information. These messages are public by default, but a Twitter account can also be made private so that messages are only accessible by users who have been approved by the account owner.

[60] See https://www.djangoproject.com
[61] See http://flask.pocoo.org
[62] See http://wink.apache.org
[63] See http://symfony.com
[64] See https://archive.farsightsecurity.com/NMSG_Data_Types/
[65] See https://github.com/google/protobuf/
[66] See https://www.farsightsecurity.com/Services/SRA/
[67] See https://dev.twitter.com/docs/api/1.1
[68] See https://dev.twitter.com/docs/api/streaming

Integration of a Twitter based channel for information sharing requires a developer account and authentication. A user can develop his own client application for accessing information by using various client libraries. The API delivers data in JSON format, although no structure is imposed on the information contained in the "tweet" itself.

Examples of channels where an automated system (bot) publishes information include @MalwareChannel and @fail2ban. An example of a channel with human-produced information is @bgpmon.

3.1.6 Internet Relay Chat

Internet Relay Chat (IRC) is a protocol for real-time text-based asynchronous communication in a client-server architecture. It was designed for human communication - exchanging of free-form text messages. It does not store history and messages are limited to 512 characters and might be rate-limited by IRC servers. While IRC can be used to implement a simple notification system, it is not suitable for sharing large amount of data. The protocol is open[69] and can be easily implemented from scratch or integrated using one of many available libraries.

IRC is based on TCP and can make use of TLS to secure the communication. The most common way of exchanging messages is through a channel, which can be described as a chat room facilitating group communication. Messages send to a channel are broadcasted to all clients that joined it. It is also possible for users to communicate with each other without necessity of joining a channel.

IRC servers connect in a form of a spanning tree and create an IRC network. The network, aside from allowing communication between users, can provide services by means of IRC bots. These are usually IRC clients equipped with scripts for automated processing of user requests. They are typically used to provide a query service and can act as a messaging proxy forwarding information from monitoring systems to a set of subscribers (channels or individual clients).

The Shadowserver Foundation uses IRC to share data about observed malicious IP addresses and provides query bot services for obtaining information about malware samples processed by their sandbox.

3.1.7 Extensible Messaging and Presence Protocol

The Extensible Messaging and Presence Protocol (XMPP), originally known as Jabber, is an XML-based protocol for message-oriented communication. Its main purpose is to provide near real-time messaging (i.e., chat services), presence information and contact list maintenance. It is an open standard, designed to be extensible, which has found many uses in various applications and platforms, both proprietary and open-source. The architecture of the XMPP network allows anybody to run their own server and the protocol itself leverages SASL and TLS for securing communication between peers.

XMPP was primarily designed with the intent to support the implementation of human text-based communication software. The body of an XMPP message holds content in the form of unstructured text, using a structure that is similar to IRC. The protocol has been adapted through various extensions to enable the machine-to-machine (M2M) exchange of structured data.

XMPP can be used to set up an information sharing channel but it was not designed to transmit large data volumes. Because it uses an XML-based format, the overhead of XMPP messages can be high, both in terms of processing time and bandwidth required, especially when transmitting binary content, which has to be base64 encoded. XMPP can be transported over TCP and, in cases where

[69] See https://tools.ietf.org/html/rfc1459

firewalls may block it, it can be implemented over HTTP and WebSocket. The protocol is implemented in various clients, servers and libraries, allowing rapid deployment of an XMPP data exchange channel.

AbuseHelper is an example of an information management system built on XMPP. The protocol is used for inter-component communication.

3.1.8 hpfeeds

The hpfeeds[70] platform and protocol were created by the Honeynet Project[71] in order to carry high-volume real-time data from a globally distributed network of honeypots. Data shared within hpfeeds is strictly security related but it is primarily used to support research rather than incident handling operations. The hpfeeds protocol is binary and lightweight. It supports authentication and arbitrary binary payloads, including the ability to share malware samples. The protocol provides separation of data feeds by using named channels to which a client can subscribe or publish. The producer of information decides on the format for the transmitted data. Access to the channel is protected by an authentication key. The hpfeeds protocol uses TCP for transport and can be secured with SSL/TLS.

The hpfriends[72] system is a modification of hpfeeds that employs a novel data sharing model: a social graph representing relationships between users is used to determine permissions. The hpfriends system uses the same wire protocol as hpfeeds, which makes it backwards compatible.

Reference implementations of hpfeeds clients exist in the Python, C, Go, Ruby and JavaScript (with node.js) languages. The hpfeeds broker is licensed as open source software,[73] but the source code of hpfriends has not been publicly released.

The hpfeeds broker and hpfriends system are still under development. Their support is already built into some tools, for example, the Dionaea[74] honeypot.

3.1.9 Really Simple Syndication

The Rich Site Summary or Really Simple Syndication (RSS) standard is used to publish lists of new information available on frequently updated web sites, like blog entries and news headlines. It is primarily used to deliver abstracts from recently published articles together with additional meta-data. An RSS document is an XML file which can be transported over any communication protocol, but it is typically made available via HTTP. The file is read by RSS client software that periodically checks the server for updates. Such software is able to import information from various RSS feeds and allows user to perform common actions on it, including searching and filtering.

RSS feeds usually contain a list of short descriptions represented as free-form text and meta-data that includes a link to the original source of information. RSS is not typically used by information sharing systems as a primary channel for delivering the data itself, but may be used to communicate information about new data that has been made available, as is the case for Project Honey Pot[75] and malwaredomainlist.com.

3.1.10 Trusted Automated eXchange of Indicator Information

Trusted Automated eXchange of Indicator Information (TAXII) is a threat information exchange standard designed for sharing data in XML format. In particular, TAXII was designed as the transport

[70] See https://github.com/rep/hpfeeds
[71] See http://honeynet.org
[72] See http://hpfriends.honeycloud.net
[73] See https://github.com/rep/hpfeeds
[74] See http://dionaea.carnivore.it
[75] See https://www.projecthoneypot.org

protocol for STIX. Simplicity and speed of sharing of information across organization were the main objectives of TAXII development. TAXII uses the HTTP protocol for message transfer. Future versions of the standard may support other message formats and protocols.

The TAXII specification defines services (the functional components of an architecture based on TAXII), messages and message exchanges (sequences of messages including handshakes) which form the basis of a TAXII-based information sharing platform. TAXII enables three data-sharing models: producers and consumers sharing information via a central hub; a single publisher producing data for multiple subscribers; and pairs of producer/consumers sharing information through a peer-to-peer arrangement. Libraries for handling TAXII messages exist in the Python and Java languages. YETI,[76] a reference implementation of TAXII, is written in Python.

One of example of a commercial sharing platform using TAXII is Microsoft Interflow[77]. TAXII is also implemented within CRITs (see section 4.2.2) to share STIX/CybOX documents with other CRITs instances. The development of TAXII was sponsored by the U.S. Department of Homeland Security (DHS) and the specification was written by the MITRE Corporation.

3.2 Serialization methods

In order to be stored and exchanged between different systems, data structures containing actionable information must be translated into a stream of bytes. This process is called serialization. There are many methods of serialization, producing different types of output. The choice of a serialization approach can have an enormous impact on data storage and exchange, impacting data volumes and the processing performance. The choice of serialization format can also affect how complex relations between objects are represented. Finally the choice of serialization format will affect the tools that will be available for working with the data.

This section covers serialization methods most commonly used for representation of actionable information in the context of incident response. While most of them produce output in text, binary formats may be preferable whenever data are binary by nature (e.g., PCAP and images) or the size of resulting files is a critical issue. One may choose to use different methods for different purposes, for example, BSON for storage of raw data and an elaborated XML formats for results of meta-analysis to be exchanged with others.

All standard formats described in the first chapter are coupled with particular serializations, usually based on XML. Therefore, a choice of serialization method is generally only available when using a custom data format.

3.2.1 Freeform text

Security data represented in a free-form text format includes alerts sent via email from one person to others, security advisories and other high-level reporting. The text may be provided in the form of flat text file, a PDF or in any of the many office document formats. Information in this form is certainly very easy to produce (no specialized tools required) and is a natural in way to communicate among human analysts. However, since they depend on human interpretation, processing free-form text documents can be time consuming and prone to errors. At the same time, it can be difficult or impossible to automate processing these documents. Mail systems and most ticketing systems (e.g., RTIR) will happily accept free-form text messages. However, their further processing is limited – usually only the most common types of information (e.g., email addresses and IP addresses) can be automatically extracted.

[76] See https://github.com/TAXIIProject/yeti

[77] http://technet.microsoft.com/en-us/security/dn750892

Typical uses:

- email communication
- advisories
- security alerts (human to human)

Pros:

- very easy to produce and read by humans

Cons:

- difficult for machine processing

Example 1) "Wanted! John Smith. Born January 21, 1985. Height: approximately 183 cm. Seen wearing a green military jacket…"

3.2.2 Raw logs

Many system logs contain security-related information. Most logs are text files with one event per line whose structure was arbitrarily defined by the authors of the application. While text-based system logs are usually easy to read for humans, the huge variation of formats and susceptibility to different configuration settings makes them difficult for machine processing. Custom parsers are required for logs from different systems, devices and applications. Such parsers for commonly encountered log formats are included as part of the collection frameworks of many security log management products.

Typical uses:

- web servers
- firewalls

Pros:

- easy to produce and distribute
- easy to read by humans

Cons:

- difficult to parse by machines
- difficult to represent structured data

Example 2) 2014-07-16 08:23:12 New person added: John Smith, 30 years of age, 183 cm height

3.2.3 CSV

The acronym CSV,[78] which stands for "character-separated values", or, alternatively, "comma-separated values", refers to any flat text format with one information record per line, where field values are separated with an arbitrary character, typically a comma, a semicolon or a vertical bar. In some cases, extra white characters are used for padding. CSV files can be read by humans and are suitable for machine handling with simple text-processing tools. They can also be easily imported into many data processing tools. However, since the meaning of fields in a particular file is implicitly defined by the data producer, additional information is needed to properly interpret the data. In many cases, the first line of a CSV file contains labels for the fields, but this may or may not be enough information to properly interpret a file. In practical terms, each CSV file format may require a different parser in order to be processed automatically.

[78] https://www.rfc-editor.org/rfc/rfc4180.txt

For data that can be naturally represented as text, CSV files introduce very little overhead since only the values themselves and separators are stored in a file. Binary data must of course be encoded, which will introduce some storage and processing overhead. Finally, like any “flat” format, a CSV file format will make it more difficult to represent relationships between data records.

Typical uses:

- any information extracted from log files
- bulk lists of events with a number of fixed attributes

Pros:

- easy to produce and parse by humans and machines
- little overhead in file size

Cons:

- only one type of records per file, no relations between records
- each CSV requires custom parser configuration

Example 3)
```
"DATE_OF_BIRTH","NAME","HEIGHT","FAVORITE_COLOR"
"1985/01/21","John Smith",183,"Green"
"1990/01/12","Jill Smith",162,"Blue"
```

3.2.4 XML

Extensible Markup Language (XML) is a standard for encoding data tended for both human and machine consumption. Documents in XML-based formats are hierarchically structured text files, where data structures, fields and values are represented as text organized around a syntax based on nested elements delimited tags. Almost any arbitrary data structure can be represented in XML in some way, including complex description languages like OpenIOC. However, the XML standard itself does not specify the structure for particular document types. Instead, XML includes a powerful (and complex) language for defining schemas. For any given application, an XML schema will need to be defined for each document type. All of the XML-based security information exchange standards define such schemas: STIX, IDMEF and IODEF (see section 2.5) all define their vocabulary in terms of a specific set of XML element types.

XML documents can be easily parsed. There are many tools and software libraries available for processing XML, and dedicated tools and APIs exist for specific XML-based formats. XML-based documents can be re-encoded in JSON or YAML.

Typical uses:

- almost any data and data structure can be represented in XML
- most industry standards for security information exchange are XML based

Pros:

- easy to parse by humans and machines
- very flexible

Cons:

- adds overhead due to verbosity of the representation

Example 4)

```
<person name="John Smith">
<height>183</height>
<favorite_color>Green</favorite_color>
</person>
```

3.2.5 JSON

Javascript Object Notation is a standard text format used for encoding data into human-readable structured collections of name-value pairs that can also be easily processed by machines. The JSON data structure syntax is based on Javascript, but is widely used in a language-independent context a standard way of transmitting data. JSON, like XML, is a generic description syntax. Virtually any document or data structures can be described using JSON. Additional information is required to validate and understand the structure of specific JSON-based formats. Although a formal scheme for defining JSON schemas[79] does exists, descriptions of JSON structures are generally described informally as part API documentation. JSON is often described as a lightweight alternative to XML because of its simpler syntax.

Typical uses:

- almost any data and data structure can be represented in JSON,
- many modern REST APIs use JSON-formatted requests and results

Pros:

- easy to parse by humans and machines,
- directly maps to existing data structures in most programming languages
- very flexible

Cons:

- with the structure comes some overhead

Example 5)

```
{
        "name": "John Smith",
        "height": 180,
        "favoriteColor": "Green"
}
```

3.2.6 Other text formats

Many security tools define their own text-based formats for representing configuration information and output. Some examples include YARA and Snort rules. They all can be read by humans (while not necessarily fully understood) or easily applied directly in appropriate software. However, automated extraction of certain pieces of information from such data (e.g., IP addresses from Snort rules) is not trivial and calls for dedicated libraries or parsers. This is because data structures are represented in application-specific languages, with their own unique syntax.

Example 6)

```
rule WantedDeadOrAlive
{
```

[79] http://json-schema.org

```
        strings:
                $a = "John Smith"
                $b = "Joe Black"
        condition:
                any of them
}
```

3.2.7 Binary formats

Finally, data can be exchanged in the form of streams of binary records or as binary files. Clearly, binary data cannot be easily processed by humans and requires tools to parse or otherwise extract information, as well as other background information in order to interpret that information. On the other hand, binary formats are usually more compact than corresponding representations as text.

In practice, binary formats are used either to address a performance issue encountered using an existing text format, or because an appropriate binary standard already exists for the data that needs to be handled. In the former case, typically a generic serialization protocol, like Protocol Buffers or BSON, is used, rather than designing a new format. The most common examples of existing formats are those used to transmit and store network monitoring data. The PCAP format is widely used to store raw packet capture data, while NetFlow (or IPFIX) is used to transmit or store network flow data. A variety of tools are available for working with both formats.

4 Information Management Tools

This chapter describes a variety of tools used by incident response teams to collect, manage and analyze security information. The goal of this chapter is not to provide an exhaustive list of all available solutions but rather to point out important representative examples of the types of tools available. To make sure that only relevant examples were included, we only selected tools that are used operationally by incident response teams and that are actively maintained. This requirement led to the exclusion of some abandoned projects (e.g., the EU-funded National and European Information Sharing and Alerting System, NEISAS), as well as systems still under development like the Cyber Defence Data Exchange and Collaboration Infrastructure (CDXI), which might be important but are currently still in the design phase.

Free and open-source projects were given preference, although some exceptions were made, most notably for Splunk, which is a commercial tool with a free version available; for IFAS and Taranis, which are freely available to a selected group of CERTs; and for n6, only parts of which are open-sourced.

This chapter consists of three sections. Each section describes a group of tools that share a primary purpose. For each tool, we describe the features provided for each of the processing stages defined in the "Actionable Information for Security Incident Response."

4.1 Automated distribution of data

Systems in this category were created primarily with the goal of collecting large volumes of threat data coming from multiple sources and distributing it to potentially impacted parties. The main users of such tools are national and governmental CERTs.

4.1.1 AbuseHelper

AbuseHelper is a software framework for the automated processing of incident reports, developed by CERT-FI, CERT-EE and Codenomicon (formerly Clarified Networks). It was released under an open-source license in 2010 and since then has been adopted by several national-level CERTs. AbuseHelper has an extensible, modular, event-oriented architecture consisting of multiple specialized scripts (*bots*) communicating over XMPP (See Section 3.1.7) and processing data streams in real time. The software is available at https://bitbucket.org/clarifiednetworks/abusehelper.

Collection:	• multiple specialized modules for fetching data from different sources • transport using HTTP or email via IMAP • native support for streaming sources (e.g., using IRC)
Preparation:	▪ dedicated parsers for each source ▪ normalization to a flat key-value format with predefined keys ▪ semantics of keys defined through a proposed ontology[80] (in development) ▪ dedicated modules for enrichment (*expert bots*) ▪ enrichment using GeoIP, passive DNS (implemented as queries to external services) ▪ mapping events to registered clients, implemented through XMPP multi-user chat rooms

[80] https://bitbucket.org/clarifiednetworks/abusehelper/wiki/Data%20Harmonization%20Ontology

Storage:	▪ no-built in database ▪ logging to text files
Analysis:	▪ no-built in analyses
Distribution:	▪ native exchange via XMPP possible ▪ emails with CSV files attached

4.1.2 Information Feed Analysis System

IFAS is a system developed by HKCERT and CSIRT Foundry. It uses AbuseHelper for collection, normalization and enrichment of threat data from external sources. Internally, IFAS employs Logstash and Elasticsearch for log transformation and storage. IFAS provides an alerting capability, a real-time dashboard, statistical reports and search (via Kibana). At the time of writing this report there was no public website available.

Collection:	▪ via AbuseHelper
Preparation:	▪ via AbuseHelper
Storage:	▪ Elasticsearch
Analysis:	▪ custom reports defined by analysts (IFAS Reporter)
Distribution:	▪ alerts via emails (IFAS Alerter) ▪ JSON/CSV/STIX planned ▪ human-oriented interface

4.1.3 IntelMQ

IntelMQ is an open source project developed by CERT.PT (with contributions done in the spare time of an employee of CERT.at) whose goal is to create a highly-modular system for the collection, processing and distribution of security information. It is the second iteration of the Incident Handling Automation Project[81] (IHAP) project and its distributed architecture is inspired by AbuseHelper. IntelMQ tries to improve on previous iterations by adding persistent storage (unlike AbuseHelper), and simplifying module development. IntelMQ uses Redis[82] for inter-component communication and provides a graphical web interface to manage the system's configuration. It is available from https://github.com/certtools/intelmq.

Collection:	▪ realized through many source-specific modules ▪ native support for real-time data feeds
Preparation:	▪ multiple specialized modules for parsing and enrichment ▪ data model based on the ontology proposed for AbuseHelper[80] ▪ uses JSON for serialization ▪ compatible with the AbuseHelper internal key-value format in order to maintain the existing time investment of AbuseHelper installations
Storage:	▪ multiple backends supported: Splunk, Elasticsearch, MongoDB, PostgreSQL

[81] http://www.enisa.europa.eu/activities/cert/support/incident-handling-automation
[82] Redis - in-memory key-value store, http://redis.io

Analysis:	▪ no automated analyses
Distribution:	▪ data can be sent to multiple systems in real-time ▪ modules for distributing data to external organizations (not yet released publicly)

4.1.4 Megatron

Megatron is an automation system for CERTs, created by CERT-SE. Its main function is processing indicator data from multiple sources, storing it for reporting purposes, and notifying affected constituent organizations. Initially the source code for Megatron was only shared within a limited group of CERTs, but in 2013 it was released publicly under an open-source license. It is available from https://github.com/cert-se/megatron-java.

Collection:	▪ batch processing ▪ external scripts used to fetch files to a designated local location
Preparation:	▪ built-in configurable parsers for text-based and XML formats ▪ enrichment: GeoIP, DNS ▪ a subscriber database is used to determine which organizations were affected by an event (based on IP, autonomous system or domain)
Storage:	▪ SQL database, simple data model ▪ lines of unprocessed input files are stored for reference
Analysis:	▪ none
Distribution:	▪ affected organizations can be notified via email (text data sent inline) ▪ limited JSON and XML export capability for internal use by a CERT

4.1.5 n6 - Network Security Incident eXchange

n6 (Network Security Incident eXchange) is a complete system for the automated collection, storage and distribution of security data that was developed by NASK, the parent organization of CERT Polska.[83] It provides a centralized repository of threat data and flexible sharing mechanisms that allow fine-grained control over the information exchange process. n6 has been used by CERT Polska to distribute information to its constituents since 2011. Access to the platform is free of charge, but the software is currently closed source.[84] In 2014 the system underwent a major upgrade, which introduced stream processing, a unified data model and a new REST API.

Collection:	▪ multiple dedicated modules for various external and internal sources ▪ asynchronous, event-oriented architecture
Preparation:	▪ full normalization - single data model for events coming from all sources ▪ enrichment using GeoIP and DNS ▪ on-the-fly aggregation of similar events ▪ a subscriber database is used to determine which organizations were affected by an event (based on IP, autonomous system, domain or country)

[83] Authors of this report are involved in the development of n6.
[84] A large part of the n6 source code will be released under an open-source license by the end of 2014.

Storage:	▪ unprocessed input data preserved for reference ▪ main repository is stored in an SQL database
Analysis:	▪ none
Distribution:	▪ REST API for external organizations ▪ multiple output formats: JSON, CSV, IODEF ▪ authentication via client X.509 certificates

4.1.6 Warden

Warden is an information sharing system developed by CESNET and used by multiple academic institutions in the Czech Republic. A central Warden server receives automated feeds containing security events from participating organizations and is responsible for distributing it to subscribed clients. Source code for Warden is available under an open source license.[85] It is available from https://csirt.cesnet.cz/Warden/Intro.

Collection:	▪ multiple detection systems (e.g., IDS, honeypots) deployed in participating organizations ▪ event-oriented architecture
Preparation:	▪ all events use a simple format with several fixed attributes (e.g., sensor, attacking IP)
Storage:	▪ no long term storage
Analysis:	▪ none
Distribution:	▪ client's subscribe to event streams ▪ API based on SOAP ▪ authentication via client X.509 certificates

4.2 Supporting analysis

This diverse category covers tools that can support various aspects of the analytical work of incident response teams. Most of the systems help with the management of information, providing ways to store and query information. To some degree they also provide facilities for collaborative work and information exchange but usually less emphasis is placed on this functionality.

4.2.1 Collective Intelligence Framework

CIF is an open-source tool developed by REN-ISAC for warehousing security information. It allows the collection of data from multiple sources in a central repository, and provides facilities to query the data. It is available from https://code.google.com/p/collective-intelligence-framework/.

Similar functionality is provided by Cikl, a reimplementation of CIF in Ruby. It is currently available as an "experimental" release as work continues on the conversion from Perl to Ruby. Its main design goals are improved performance, scalability, functionality, and ease of installation. It is available from https://github.com/cikl/cikl.

[85] Source code of Warden is available under following address: ftp://homeproj.cesnet.cz/tar/warden/ (there is no link on the official website).

Collection:	▪ batch processing ▪ fetching files from multiple sources over HTTP ▪ sources defined via configuration files
Preparation:	▪ built-in configurable generic parsers for CSV, JSON, XML and free-text ▪ periodic enrichment of data through queries to external services: DNS, reputation databases
Storage:	▪ explicitly uses IODEF data model ▪ event-oriented ▪ PostgreSQL ▪ confidentiality level set per event
Analysis:	▪ none built-in
Distribution:	▪ server provides two APIs: REST and RPC based on Protocol Buffers ▪ client application provides multiple output formats: IODEF, JSON, CSV, Snort rules, text ▪ periodic feed (e.g., blacklist) generation

4.2.2 Collaborative Research Into Threats

Recently released under an open source license, CRITs is a tool with the specific goal of supporting the workflow of analysts responsible for developing indicators from threat data. It is a centralized repository for various types of threat information (IoC, binaries, PCAP) and uses a data model influenced by STIX and CybOX. By employing a plugin-based architecture, it allows the integration of scripts that analyze and correlate collected data that can be used to automate aspects of analysts' workflow. It can be dowloaded from https://crits.github.io.

Collection:	▪ multiple types of supported data: binaries, PCAP, emails, IP, domains, IoC, campaigns, certificates, events, raw data, and targets ▪ Import: CybOX, STIX ▪ transport: TAXII, REST API
Preparation:	▪ enrichment of stored data implemented using plugins ("services") ▪ correlation with external data (e.g., passive DNS, VirusTotal, DNS lookups) ▪ matching YARA rules
Storage:	▪ MongoDB ▪ data model based on CybOX and STIX
Analysis:	▪ analysis of stored data is done by plugins ▪ extraction of DNS and HTTP data from PCAPs (via ChopShop) ▪ identification of similar binaries by fuzzy hashing ▪ unpacking of binaries (UPX) ▪ extraction of metadata from multiple file types ▪ integration with external analysis services (e.g., sandboxes)
Distribution:	▪ supported output formats: CybOX, STIX, JSON ▪ transport via TAXII ▪ REST API for internal use

4.2.3 Malware Information Sharing Platform

MISP is an open source system for the management and sharing of IoCs, initially built to support the NATO Computer Incident Response Capability (NCIRC). Its main goal is to facilitate sharing information related to targeted attacks and malware. Its primary features include a centralized searchable data repository, a flexible sharing mechanism based on defined trust groups and semi-anonymized discussion boards. MISP focuses on the exchange of the most valuable indicators selected and annotated by analysts, and not on processing high-volume automated data feeds. It is available at https://github.com/MISP/MISP.

Collection:	▪ REST API ▪ import via web form, XML, OpenIOC, and CSV
Preparation:	▪ whenever data is imported into MISP, new detection indicators are matched against previously collected ones, providing links between new and old events
Storage:	▪ central SQL database ▪ normalized format (events/incidents and atomic attributes) ▪ not designed for high-volume automated feeds
Analysis:	▪ no automated analyses
Distribution:	▪ export as OpenIOC ▪ export as IDS signatures ▪ export as custom XML and CSV ▪ REST API

4.2.4 MalCom

Malware Communications Analyzer (MalCom) is a tool for analysis of network traffic, in particular traffic generated by malware samples. It correlates observed activity with multiple sources of threat information and presents it using interactive graph-based interface. It can be downloaded from https://github.com/tomchop/malcom.

Collection:	▪ PCAP or live network traffic ▪ indicators from threat feeds (predefined open sources or private ones
Preparation:	▪ DNS data ▪ YARA rule matching
Storage:	▪ MongoDB
Analysis:	▪ correlation between communications and known indicators ▪ visual graph-based representation ▪ supports manual analysis
Distribution:	▪ REST API ▪ access control is supported by mapping API keys to the tags associated with stored data

4.2.5 The MANTIS Cyber Threat Intelligence Management Framework

MANTIS is an open-source web application (implemented using the Django framework) whose main purpose is the management of structured threat information. It has native support for working with documents in the STIX, CybOX, OpenIOC and IODEF formats, and aims to provide an environment that can be used for the research and development of these standards. It is available from http://django-mantis.readthedocs.org.

Collection:	▪ import via web interface or using an API ▪ supports IODEF, STIX, CybOX, OpenIOC
Preparation:	▪ none
Storage:	▪ internal data model preserves original structure of supported formats ▪ limited capacity
Analysis:	▪ none
Distribution:	▪ REST API ▪ multiple formats: IODEF, STIX, CybOX, OpenIOC, JSON

4.3 General purpose log management

There are many solutions for log management currently on the market, both commercial and open source. Typically, these systems are deployed to centralize log collection and monitoring of an organization's infrastructure as part of a security monitoring solution. However, they can be also configured or adapted to handle other types of data (e.g., indicators) and to implement many of the functions of the more specialized tools that were described in the previous two sections.

4.3.1 Enterprise Log Search and Archive

Enterprise Log Search and Archive (ELSA) is a centralized repository for log data whose design was inspired by Splunk (see section 4.3.3), and which was developed with the goal of supporting high data ingest rates. ELSA provides a graphical web interface with an integrated query language that can be used to search the database, correlate and aggregate the data. Source code is available on an open source licence from https://code.google.com/p/enterprise-log-search-and-archive/.

Collection:	▪ no built-in collection mechanisms specific to security data ▪ generic syslog input ▪ HTTP and local IPC inputs for batch imports
Preparation:	▪ parsers for several common log formats (e.g., Snort and Cisco) ▪ parsers for other formats can be defined through configuration files
Storage:	▪ SQL database ▪ full-text indexing
Analysis:	▪ customizable dashboards ▪ correlation with external data sources (e.g., databases, VirusTotal) through query language
Distribution:	▪ queries can be scheduled and results sent by email

4.3.2 Elasticsearch + Logstash + Kibana

Elasticsearch, Logstash and Kibana (ELK) together form a popular software stack to parse, index, search and visualize data. Logstash is used to receive and transform data from multiple sources, Elasticsearch provides a distributed datastore with advanced query capabilities and Kibana provides an interactive graphical frontend that can be used to build customized dashboard views. All three tools are mature open source projects. They are available from http://www.elasticsearch.org.

Collection:	▪ collection of data performed by external tools ▪ internally handled by Logstash ▪ multiple channels (files, message brokers, scripts, etc) ▪ event-oriented, works in real-time
Preparation:	▪ Logstash filters ▪ GeoIP, DNS ▪ matching of IP addresses to a list of predefined networks
Storage:	▪ Elasticsearch ▪ scalable document store
Analysis:	▪ no automated analyzes ▪ graphical user interface provided by Kibana: generic query language and configurable dashboards
Distribution:	▪ Elasticsearch has a REST API but it is designed for internal use, not sharing with external parties ▪ output format: JSON

4.3.3 Splunk

Splunk is a commercial monitoring and analytics tool that can handle large volumes of data. A free version of Splunk is available, although it has somewhat limited capabilities. It is an example of general-purpose log management software that CERTs can use to process security information. By extending it with a couple simple scripts, it is possible to build a complete data processing pipeline on top of it. More information about Splunk can be found at the company's website: http://www.splunk.com.

Collection:	▪ no built-in collection mechanisms for security data so external tools are required ▪ multiple transport channels supported (syslog, files, message queues, etc.) ▪ real-time or batch processing
Preparation:	▪ no built in parsers for specific formats ▪ generic facilities for parsing text, CSV, JSON and XML ▪ simple data model based on lists of key-value pairs ▪ semantics and syntax are not enforced ▪ enrichment possible by integration with external sources (e.g., GeoIP and SQL databases)
Storage:	▪ custom database suited for large-scale deployments ▪ full-text search

Analysis:	▪ generic query language with statistical capabilities (modeling, trend analysis, etc.) - can be used for manual interaction or automated analyses ▪ users can create custom dashboards
Distribution:	▪ no built-in mechanisms specific to information security ▪ alerting capability based on arbitrary criteria ▪ external scripts can be easily integrated to export data in desired formats

4.4 Handling high-level information

While previous categories covered tools that were focused on managing lower-level security data, logs, and indicators, the two systems described below were designed to manage information at a higher level of abstraction. This includes artifacts like tickets in a workflow system, incident reports, security advisories, warnings, and similar types of information that will be interpreted by a human analyst or incident handler.

4.4.1 Request Tracker

Request Tracker (RT) is an open-source ticketing system, that offers a plugin (RTIR) that supports incident handling workflows. This is an example of a tool that can be used to manage the whole lifecycle of incidents, from reporting to remediation. Email is the primary mean of interaction with the system but it also offers a web user interface and REST APIs. Unlike the systems we have described so far, RTIR is not a security data storage and analysis platform. Instead, it provides incident handling specific semantics on top of the task tracking features of RT. That said, the architecture of RTIR allows for a wide range of customizations, and can be integrated with external tools that support analysis functions. RTIR is available from http://bestpractical.com/rtir/.

Collection:	▪ accepts emails but contents are not parsed by default ▪ processes incident reports and related communication
Preparation:	▪ through external tools only
Storage:	▪ SQL database, can be accessed directly for integration or customization purposes
Analysis:	▪ none relevant to security information
Distribution:	▪ emails, using built-in templates or content generated by external tools

4.4.2 Taranis

Taranis is a system developed by NCSC-NL to facilitate situational awareness and manage the flow of advisories, announcements and other high-level reporting typically generated by a typical CERT. NCSC-NL uses it internally to gather data from approximately a thousand different sources, and to manage the production and distribution of reporting based on that information. The software is not available publicly but trusted CERTs can obtain it free of charge and deploy their own instances of the system with any set of sources. More information is available at https://www.ncsc.nl/english/services/incident-response/monitoring/taranis.html.

Collection:	▪ many sources, support for multiple transport mechanisms (email, web pages, RSS) ▪ most sources provide unstructured text ▪ built-in crawler that detects changes in monitored sources automatically ▪ most sources provide vulnerability alerts and other warnings related to threats that might be relevant to constituents
Preparation:	▪ the system assists with clustering of similar news items ▪ includes a tool to rate the risk associated with vulnerabilities (through a custom model based on impact and chance) but the assessment itself is performed manually by analysts
Storage:	▪ internal PostgreSQL database
Analysis:	▪ analysis is a step in the workflow supported by Taranis, but analysis itself is not automated within the tool
Distribution:	▪ includes multiple methods to output information, including email, website, SMS ▪ output is in textual form, generated through templates ▪ integration with external instances of Taranis is possible

Annex A: Abbreviations

The table below presents the list of abbreviations used in the document.

Abbreviation	Explication
API	Application Programming Interface
ARF	Abuse Reporting Format
ARF	Asset Reporting Format
BSON	BInary JSON
CAPEC	Common Attack Pattern Enumeration and Classification
CCE	Common Configuration Enumeration
CCSS	Common Configuration Scoring System
CDESF	Common Digital Evidence Storage Format
CDXI	Cyber Defence Data Exchange and Collabiration Infrastructure
CEE	Common Event Expression
CEF	Common Event Format
CERT	Computer Emergency Response Team
CIF	Collective Intelligence Framework (software)
CMSS	Common Misuse Scoring System
CPE	Common Platform Enumeration
CRE	Common Remediation Enumeration
CRITs	Collaborative Research Into Threats (software)
CSV	Character-Separated Values or Comma-Separated Values
CVE	Common Vulnerability Expression
CVRF	Common Vulnerabilities Reporting Framework
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
CWRAF	Common Weakness Risk Analysis Framework
CWSS	Common Weakness Scoring System
CYBEX	Cybersecurity Information Exchange
CybOX	Cyber Observable Expression
DBIR	Data Breach Investigation Report (Verizon publication)
DFXML	Digital Forensics XML
DHS	Departament of Homeland Security
DNS	Domain Name System
ELK	Elastic Search, Logstash, Kibana (software stack)
ELSA	Enterprise Log Search and Archive
ETSI	European Telecommunications Standards Institute
FIRST	Forum of Incident Response and Security Teams
HTTP	Hypertext Transfer Protocol
ICASI	Internet Consortium for Advancement of Security on the Interne
IDEA	Intrusion Detection Extensible Alert
IDMEF	Incident Object Description Exchange Format

IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IFAS	Information Feed Analysis System
IHAP	Incident Handling Automation Project
IOC	Indicator of Compromise
IODEF	Incident Object Description Exchange Format
IODEF-SCI	IODEF for Structured Cybersecurity Information
IPFIX	Internet Protocol Flow Information Export
IRC	Internet Relay Chat
ISI	Information Security Indicators
ITU	International Telecommunication Union
JSON	JavaScript Object Notation
MAEC	Malware Attribute Enumeration and Classification
MalCom	Malware Communications Analyzer
MARF	Messaging Abuse Reporting Format
MDA	Mail Delivery Agent
MILE	Managed Incident Lightweight Exchange
MIME	Multi-purpose Internet Mail Extension
MISP	Malware Information Sharing Platform
MMDEF	Malware Metadata Exchange Format
n6	Network Security Incident eXchange
NEISAS	National and European Information Sharing and Alerting System (software)
NICT	National Institute of Information and Communications Technology (Japan)
NIST	National Institute of Standards and Technology (USA)
NTAR	Network Trace Archival and Retrieval library
OASIS	Organization for the Advancement of Structured Information Standards
OCIL	Open Checklist Interactive Language
OpenIOC	Open Indicators of Compromise
OSF	Open Security Foundation
OSVDB	Open Sourced Vulnerability Database
OVAL	Open Vulnerability and Assessment Language
PCAP	Packet CAPture [file]
PDF	Portable Document Format
PLARR	Policy Language for Assessment Results Reporting
REST	Representiationl State Transfer
RFC	Request For Comments [document series]
RID	Real-time Inter-network Defense
RMON	Remote Network Monitoring
ROLIE	Resource-Oriented Lightweight Indicator Exchange
RSS	Rich Site Summary or Really Simple Syndication
RT	Request Tracker (software)

RTIR	Request Tracker for Incident Response (software)
SACM	Security Automation and Content Monitoring
SACM	Structured Assurance Case Metamodel
SASL	Simple Authentication and Security Layer
SBVR	Semantics of Business Vocabulary and Business Rules
SCAP	Security Content Automation Protocol
SCTP	Stream Control Transmission Protocol (Internet standard)
SecDEF	Security Description and Exchange Format
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
SSL	Secure Socket Layer
STIX	Structured Threat Information Expression
SWID Tags	Software Identification Tags
TAXII	Trusted Automated eXchange of Indicator Information
TCP	Transmission Control Protocol
TLP	Traffic Light Protocol
TLS	Transport Layer Security
TMSAD	Trust Model for Security Automation Data
TTP	Tactics,Techniques and Procedures
UDP	User Datagram Protocol
UPX	Ultimate Packer for eXecutables (software)
URL	Uniform Resource Locator
VERIS	Vocabulary for Event Recording and Incident Sharing
WASC	Web application Security Consortium
X-ARF	Extended Abuse Reporting Format
XACML	eXtensible Access Control Markup Language
XCCDF	Extensible Configuration Checklist Description Format
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol
YAML	YAML Ain't Markup Language
YARA	Yet Another Regex Analyzer (software)

ENISA
European Union Agency for Network and Information Security
Science and Technology Park of Crete (ITE)
Vassilika Vouton, 700 13, Heraklion, Greece

Athens Office
1 Vass. Sofias & Meg. Alexandrou
Marousi 151 24, Athens, Greece

ISBN: 978-92-9204-105-2

doi: 10.2824/37776

PO Box 1309, 710 01 Heraklion, Greece
Tel: +30 28 14 40 9710
info@enisa.europa.eu
www.enisa.europa.eu