HIDEF: a Data Exchange Format for Information Collected in Honeypots and Honeynets

CRISTINE HOEPERS[1]
NANDAMUDI L. VIJAYKUMAR[2]
ANTONIO MONTES[3]

[1]Brazilian Network Information Center - NIC.br
Computer Emergency Response Team Brazil - CERT.br, São Paulo (SP)
cristine@acm.org
[2]National Institute for Space Research - INPE
Computing and Applied Mathematics Associated Laboratory - LAC, São José dos Campos (SP)
vijay@lac.inpe.br
[3]Ministry of Science and Technology - MCT
Renato Archer Research Center - CenPRA, Campinas (SP)
antonio.montes@cenpra.gov.br

Abstract. The deployment of honeypots is one of the methods used to collect data about attack trends in computer networks. The lack of a standard format for data representation makes the exchange and centralization of data generated by different technologies difficult. This also restricts the correlation and analysis of this information. This paper presents the HIDEF (Honeypots Information and Data Exchange Format), a proposal for a format to enable the representation and exchange of data and information produced by organizations using honeypots and honeynets.

Keywords: network security, honeypots, honeynets, XML, data exchange format

(Received November 02, 2007 / Accepted February 02, 2008)

1 Introduction

In the past few years the number of computer security incidents on Internet connected networks has continuously increased [3, 4]. As a result, there is an increasing need for attack data correlation and tools to help understand attacks and identify trends.

The deployment of sensors in computer networks to gather malicious traffic is one of the methods used by researchers and security professionals to collect data for attack trends analysis. One type of sensor that has been used is a honeypot, a security resource whose value lies in being probed, attacked or compromised [17, 16]. Another type of sensor used is a honeynet, a network specifically designed for the purpose of being compromised, that has control mechanisms to prevent it from being used as a base of attacks against other networks [18, 12]. The honeypot technologies have considerably developed in the past few years, mainly because of the increase in research activities and the development of new ways to collect data about attacks [16].

However, it is very important to enable a more robust and complete analysis of the traffic captured by honeypots and honeynets that use different technologies and are deployed in different locations around the world. This analysis would make it possible to better understand the attacks' distribution and how they interrelate [18]. To allow the correlation of information from different honeypot and honeynet implementations, it is not only necessary to have a system to collect and analyze these data, but also to have a format to represent this information. The existence of a standard format would facilitate the exchange of data, because it would enable the development of tools to automate the generation and retrieval of

information from different sources.

The existing research in the area of honeypot data collection and analysis is concentrated in visualizing and correlating data from a unique honeynet or from a set of honeypots using similar technologies [18, 16]. However, none of them considers the interoperability issue of analyzing data generated in different architectures by different technologies. On the other hand, standard formats to represent data related to attacks and intrusion detection, such as the Intrusion Detection Message Exchange Format (IDMEF) [7] and the Incident Object Description Exchange Format (IODEF) [6], are not adequate to represent data from honeypots.

To contribute to this research area this work proposes HIDEF (Honeypots Information and Data Exchange Format), a format to represent and exchange data and information produced by organizations using honeypots and honeynets. This proposed format intends to solve the interoperability issue among different honeypot implementations, while preserving compatibility with other standards like IDMEF and IODEF.

The remainder of this paper is organized as follows. Section 2 describes the different honeypots technologies and types, the different data analysis approaches and the different data formats for representing data about attacks in computer networks. In this Section some limitations of the approaches and formats shown are also raised. In Section 3 the HIDEF format is discussed, along with its structure and an example of data captured in a low-interaction honeypot represented in HIDEF format. In Section 4 the conclusions and future work are presented.

2 Honeypots and Data Formats

This Section initially presents honeypot types and technologies, as well as the existing approaches to collect and analyze their data. Then, two data formats for the representation of attacks in computer networks will be presented. Finally, the limitations of the methods presented, regarding data interoperability for honeypots using different technologies, will be discussed.

2.1 Honeypots

Honeypots are security resources specially configured to collect information about attacks and whose value lies in being probed, attacked or compromised [17, 16]. In general, they are computers specially configured to register all activities directed to them, such as probes, attacks and intrusions [15]. Also, as a honeypot is not a production system, all traffic directed to it is most likely malicious or anomalous. This makes the data collected in honeypots really valuable, because there are no false positives and it is easy to extract signatures for malicious activities [17].

2.1.1 Types of Honeypots

Honeypots can be classified basically into two types: low-interaction and high-interaction [17, 16].

Low-interaction honeypots are configured just to emulate certain systems and services. When an attacker interacts with one of these honeypots, she is not interacting directly with the system, but with a program or set of programs designed to emulate the system's characteristics, like operating system (OS), application version, vulnerabilities, etc. This way the honeypot is not actually compromised, but is still able to capture several pieces of information about the type of attack being perpetrated, like scan trends and new malicious code, among others [16, 11].

High-interaction honeypots are real systems, usually located in a network with malicious traffic control mechanisms, known as honeynet. These systems, once attacked, can be compromised and the intruder can completely take over the machine. In this kind of honeypot is possible to observe the intruder installing tools, attempting attacks against other networks and modifying the system configuration [18, 16].

Nowadays, honeypots and honeynets are high-end security resources used both to better understand the intruders actions and to help in activities like intrusion detection [16]. It is also a consensus that they can capture valuable information about different attacks on the Internet, as well as help computer security incident handling activities and provide details about the attackers' tactics and motives [17, 18, 12].

2.1.2 Honeypot Technologies

There are several technologies available for the implementation of low- and high-interaction honeypots. In this Section we will present the technologies that are more mature and widely adopted.

The most used programs to implement low-interaction honeypots are Honeyd and Nepenthes [16]. Honeyd is a framework that allows the emulation of hundreds of different systems in several different IP (Internet Protocol) addresses. For each IP address it is possible to specify a different OS to be emulated, and which applications will be emulated in each OS. To emulate the applications and services it is possible to use Honeyd native subsystems or external programs [16, 15]. Honeyd can run in different operating systems, and some logs are generated by the host system. Nepenthes is a honeypot designed to collect samples of malware that propagate automatically, like worms and bots [16, 8]. It emulates the behaviour of a vulnerable software, decodes the attack and retrieves a copy of the malware [8].

As high-interaction honeypots are in fact real systems

deployed in a controlled environment or in a honeynet, the technologies used to implement them have a greater degree of complexity. Each honeypot may have a different OS and this makes the information available to the analysis, after a compromise, very diverse. This information includes OS logs, application logs, process states, kernel related information, files left by the intruder, among several other data. The capture and control mechanisms can also be implemented through different technologies. One of the widely adopted technologies is the honeywall, comprised of a set of integrated tools that restrict the outgoing traffic generated by intruders, therefore stopping them from attacking third party networks, but still allowing the intruder interactions with the honeypot [18, 16]. Honeywall also has mechanisms that enable the capture of any traffic generated by the attackers, and the aggregation and preliminary analysis of the data collected in a given honeynet.

2.2 Data Collection Performed by Groups Studying Honeypots

This Section discusses the approach taken by three well known groups that study and deploy honeypots, to implement a central data collection infrastructure for further data analysis.

2.2.1 Honeynet Project

The Honeynet Project maintains several honeynets around the world, whose data are all stored in a central server. To allow further correlation of the data, a set of requirements that must be fulfilled during the data capture in all honeynets was defined [18]:

- a record with the configuration of all active honeypots in the honeynet must be maintained;
- the data captured by the firewall, router or IDS (Intrusion Detection System) must be stored in GMT (Greenwich Mean Time) timezone;
- each honeynet must have a unique identifier, a name convention and a mapping, that allow to identify its location and configuration;

Although the description of data collection to a central server is available, there is no information about how this server is structured or which type of correlation is performed.

2.2.2 Honeynet Research Alliance

The Honeynet Research Alliance is a forum where the participants are organizations from several countries, that perform research on honeypots and honeynets [18].

Up to now there is no method available for these institutions to share data or analysis among them. However, some organizations send data to a central server maintained by the Honeynet Project. This central server can deal only with data generated by this specific set of capture and control tools: the `libpcap` library binary files and Linux `iptables` firewall logs [18]. This implies that only organizations using these technologies can send data to the central server.

It is also important to notice that this architecture does not enable the exchange of information directly among the organizations. There is also no available information about which kind of data analysis or correlation is being done with the data.

2.2.3 Brazilian Honeypots Alliance

The Brazilian Honeypots Alliance publishes daily statistics about the activities observed in the distributed honeypots that are part of its infrastructure [11]. The data used to generate the statistics are logs in `libpcap` format generated by the OpenBSD packet filter (pf) [10].

Once all the honeypots' data are available at the data collection server, these data are converted into flow format, which stores a summary about the traffic between two IPs, taking into account the ports and protocol used. These flows are processed by ORCA (`http://www.orcaware.com/orca/`) and by RRDtool (`http://oss.oetiker.ch/rrdtool/`), which are the tools used to generate the daily statistics and graphics in the project's website.

2.3 Other Data Collection and Analysis Initiatives

2.3.1 Honeynet Security Console

The Honeynet Security Console (HSC) is a tool developed by Jeff Bell, from the Florida Honeynet Project, to correlate events from a local network or honeynet. This tool allows storing and querying the following types of data: `libpcap` files generated by `tcpdump`, `syslog` logs, firewall logs stored by `syslog` and logs from the Sebek tool, which captures all commands executed by an intruder in a high-interaction honeypot [18].

The HSC authors have chosen to store the data as simply as possible in a SQL database. This was achieved by using programs already available to convert data from each application to a relational database. Each one of these programs creates its own table structure, with different mapping and data types. Thus similar data created by different applications will be represented differently. To solve these inconsistencies HSC needs to perform data type conversions and correlate some data after it performs the queries to the database. In some of the cases if the data were mapped differently, the correlations could be done directly as queries to the database.

2.3.2 A Statistical View of The Recorded Activity On a Honeynet – HoneyStats

The Internet Systematics Lab, from the National Center of Scientific Research "Demokritos" in Greece, maintains the Greek Honeynet Project. In May 2006 they released a tool called "HoneyStats 1.0", which creates statistics about logs generated by the laboratory's honeynet firewall. A demo version of this tool is available at `http://www.honeynet.gr/`.

2.3.3 Georgia Tech Honeynet Project Statistics and Data Visualization

The Georgia Tech Honeynet Project has developed some tools to generate statistics and data visualization from the data collected in their honeynet [5].

The tool HoneyReport parses files in `libpcap` format and generates flows as output. These flows are then analysed to generate statistics about the top activities in the honeynet, like most scanned ports and source of attacks. The tools Rumint and SecVis generate graphics in parallel coordinates based on the traffic captured by the honeynet.

2.4 Data Exchange Formats

XML (Extensible Markup Language) is widely adopted to store and transmit information used by different software and systems [1, 9]. It is actually becoming a universal format for data exchange between applications [13].

In the following Sections we are going to present two XML data formats to represent data captured by IDSs and data from computer security incidents.

2.4.1 Intrusion Detection Message Exchange Format

The IETF (Internet Engineering Task Force) standard described in RFC 4765: "The Intrusion Detection Message Exchange Format (IDMEF)" [7] defines procedures and a data format to share information about intrusion detection. More specifically, it defines a data format to be used by automated alert notification systems [7]. One of its objectives is to allow the correlation of information collected by systems from different vendors and from open source tools.

The IDMEF is an object-oriented representation, implemented in XML, of alert data sent by IDS sensors to their data analysis systems. The definition of the data model has these main requirements: allow the interoperability among different IDSs and accommodate different levels of information provided by different data types, like network traffic, OS logs and application logs [7, 19].

2.4.2 Incident Object Description Exchange Format

The IODEF (Incident Object Description Exchange Format) is a standard format for the representation of data and statistics usually required to effectively respond to a security incident. The IODEF data model is described in the RFC 5070 "The Incident Object Description Exchange Format", from December 2007 [6]. This document also shows an XML implementation of the data model.

As it is common for an incident to be initially observed in control and monitoring systems like IDSs, the IODEF kept its definition compatible with the IDMEF format. This way, data generated in IDMEF format can be easily imported to an IODEF document.

2.5 Limitations in the Collection, Analysis and Data Format Areas

As we could see in the previous Sections the existing studies in the areas of collection and analysis of honeypots' data are focused in visualization and/or correlation of data in a given honeynet or in a set of honeypots that use similar technologies [18]. Although some of these studies present interesting characteristics and have promising results, none of them takes into consideration the issue of interoperability to analyze data generated by different technologies in different architectures.

On the other hand, the IODEF and IDMEF standards have a focus in interoperability and a structure that allows representing network traces and information about IDSs alerts adequately. However, these models are too focused on network attack signatures and on mapping relationships among security incidents. From the point of view of research and data correlation of events observed in a honeypot, it is necessary to also exchange information about the technology being used. This would allow, for example, the comparison between data from low- and high-interaction honeypots or determining if filters applied to a network have implicated in differences in the results.

To address the existing limitations of the honeypot research area, this paper proposes the HIDEF, an exchange format for data collected in honeypots, and information about the architecture and technologies used by honeypots. Section 3 will present HIDEF, its requisites and data model.

3 HIDEF

As seen in Section 2, honeypot data correlation has been performed only in data with the same format and among honeypots using similar technologies. However, it is very important to enable a more robust and complete analysis of the traffic captured by honeypots and honeynets that use different technologies and are deployed in different

locations around the world. This analysis would make possible to better understand the attacks' distribution and how they interrelate [18].

This work contributes to this research area in proposing a data format for the exchange of information and data collected in honeypots. This format is the HIDEF – Honeypots Information and Data Exchange Format.

3.1 Requirements

The HIDEF format takes into account some requirements for the representation of relevant information to analyze and correlate activities captured by honeypots.

To perform a correct analysis it is necessary to consider the architecture used and the context were the data was captured. To enable this, it is necessary that the format allows representing data about the honeypots configuration and the technologies used, such as: what is the type of the honeypot; if it is connected to a honeynet; which OS and services are being used or emulated; open TCP, UDP and ICMP ports in the honeypot; and if there are filters that prevent certain types of traffic to reach the honeypot.

It is equally important to be able to represent different types of data captured, like intruder activity logs and attack attempts received by the honeypot. Among these data it is important to highlight the network traces, malicious codes, artifacts resulting from malicious activities and the analysis performed. It is also necessary that the format allows exchanging sanitized information, that is, information that omits sensitive network details from the organizations that are interested in exchanging data collected by their honeypots.

Another requirement that needs to be considered is the compatibility with standard formats already defined for other sets of data related to network security. This is important to enable the correlation among data from honeypots and data from IDSs or from computer and network security incidents. To fulfill this requirement it is necessary that the HIDEF remains compatible with the IDMEF [7] and the IODEF [6] formats, discussed in Section 2.4.

3.2 Data Types

Whenever possible, XML Schema native data types [2] are used in the definition of the HIDEF classes and attributes. However, to represent some specific data we defined a few data types. These data types are derived from IODEF for compatibility reasons. The types defined are listed below.

ML_STRING – this type is derived from IODEF. It is used to represent any text related to analysis, general descriptions or any other information that may be written in multiple languages. It is represented by a string (`xs:string` [2]) that has a language (`xs:language` [2]) as an attribute.

PORTLIST – this type is also derived from IODEF. It represents a list of network services ports. This format allows representing ports (N) and port sequences (N-M) separated by commas. This type is represented by a string (`xs:string` [2]) restricted by the following regular expression:
`\d+(\-\d+)?(,\d+(\-\d+)?)*`.

ENUM – enumerated data, also derived from IODEF, are used in several attributes. They are defined as sequences of NMTOKEN (`xs:NMTOKEN` [2]). The description of each specific enumerated type will be made together with the description of the class that uses it for the first time.

3.3 Data Model

The HIDEF is an object-oriented representation of information related to the configuration and the technologies used by honeypots, as well as data captured by them. The HIDEF classes represent XML elements and their attributes represent XML attributes. Throughout this Section the following conventions are used: the classes represented in gray are derived from the IODEF model; the names of the attributes and all classes are written in sans-serif font.

Each HIDEF document is an instance of the HIDEF-Document class, which is comprised of one or more instances of the Honeypot class, as we can see in Figure 1. The HIDEF-Document class has three attributes: version, that represents the version of the HIDEF model being used; lang, that represents the language of the document according to the values defined in [14]; and instructions, a field reserved to contain parsing instructions, whose semantic must be defined by the organizations exchanging information.

The Honeypot class provides a representation of the several components related to the data from a given honeypot. It has the following attributes: restriction, type and ext-type. The restriction attribute, as it is used also in IODEF, will have the same meaning in HIDEF, as well as the same possible values. This attribute is inherited by all classes that are children of Honeypot. Some of these classes also have a restriction attribute, in which case it is possible to assign new values to them. The possible values are: public, need-to-know, private and default. When the value is default it means that a policy predefined by the organizations involved should be used. The type attribute determines which type of honeypot is being represented, and can have the values low-interaction, high-interaction or ext-value. The ext-type attribute is optional and was

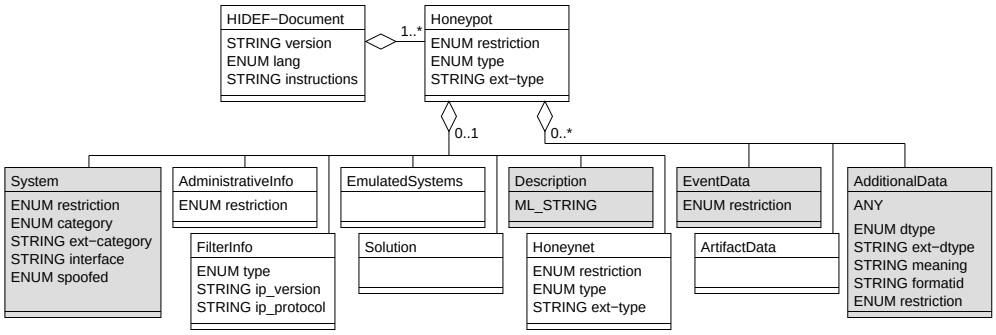

Figure 1: HIDEF-Document and its aggregated classes.

defined as a mean to extend the type attribute. More details about how to extend this and other similar types is discussed in Section 3.4.

Each Honeypot class instance may have zero or one instance of the following classes: System, AdministrativeInfo, EmulatedSystems, Description, FilterInfo, Solution and Honeynet. These classes represent the information about the technologies used and the context in which the data was collected. To represent different types of data captured, the logs of the intruders activities and attack attempts received, each instance of the Honeypot class may have zero or more instances of the classes EventData, ArtifactData and AdditionalData. These classes and their relationship with the Honeypot class can be seen in Figure 1.

The System, Description, EventData and AdditionalData classes are derived from IODEF and their full description, including attributes and aggregated classes, is available in [6]. In the HIDEF context a System class will hold information about the honeypot system, including operating system, services, applications, IP address version 4 or 6, among others. The Description class may hold a general description of the honeypot and comments that the organization creating the document considers relevant. EventData is the class that will store the data about most of the malicious activities captured by the honeypot. This class allows representing the start and end time of an activity, network traces, involved protocols, logs, attack methods, and assessment about the impact of the attack registered, among other data. The AdditionalData will be discussed in Section 3.4.

The AdministrativeInfo class, seen in Figure 2, aggregates the classes Hardware, EmulatedAddress, Sanitization, Contact and AdditionalData. It can have zero or more instances of these aggregated classes, which hold administrative information about the honeypot. In case of a low-interaction honeypot the EmulatedAddress class, which has the same definition of the IODEF Address class [6], can be used to represent the network range being used by the honeypot for the emulated systems. Hardware enables to store relevant text information about the hardware being used by the honeypot. The Sanitization class can be used to reference the relation between the honeypot real addresses and the sanitized addresses stored in the System and EventData classes. It is very important to keep this relation because it allows an institution generating the logs with the addresses already sanitized to share the information about the real addresses. The Contact class is derived from IODEF [6] and permits, for example, that members of groups studying honeypots or similar projects share information about who is responsible for a given honeypot. The AdditionalData class will be discussed in Section 3.4.

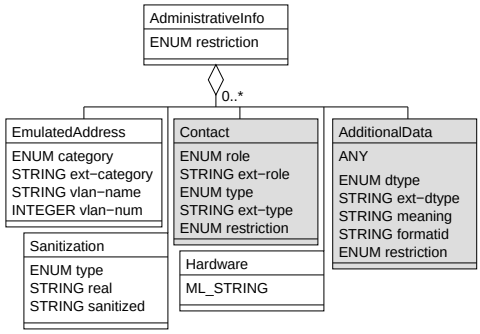

Figure 2: AdministrativeInfo and its aggregated classes.

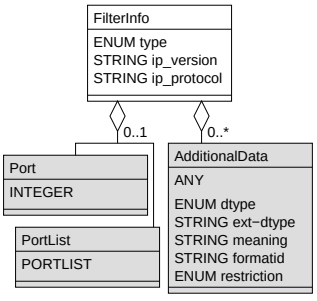

Figure 3: FilterInfo and its aggregated classes.

The FilterInfo class (Figure 3) maps the information about filters applied between the Internet traffic and the honeypot. This information is very important for the correlation of data captured in different honeypots. Without

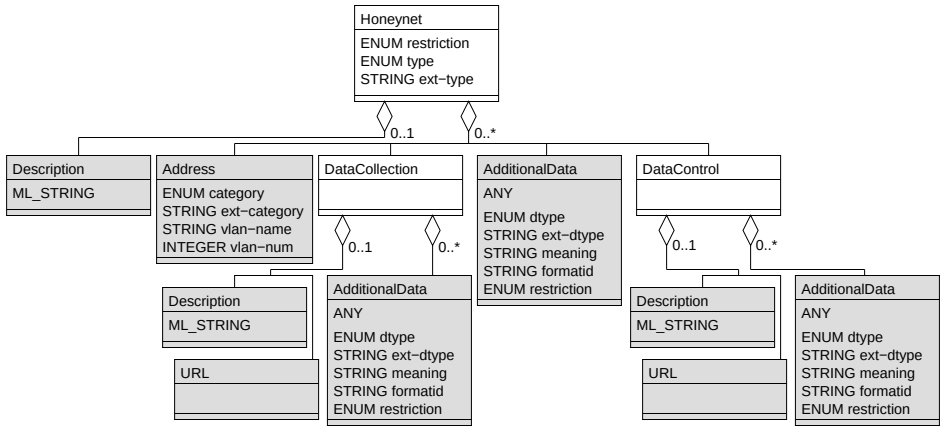

Figure 5: Honeynet and its aggregated classes.

this information it is not possible to know if a given malicious traffic was not seen because the honeypot was not attacked or the malicious traffic was blocked. If the type attribute has the value open this means that the traffic to the ports listed in Port and Portlist is allowed and everything else is denied. If the type attribute is closed, this means that the traffic to the ports listed in Port and Portlist is blocked and everything else is allowed. The Port and Portlist classes are derived from IODEF and their complete description is available in [6]. The AdditionalData class will be discussed in Section 3.4.

The EmulatedSystems class represents different systems being emulated in a low-interaction honeypot. It is formed by one or more instances of the System class, as we can see in Figure 4. The System class has basically the same definition as in IODEF [6], but in HIDEF it has been extended to aggregate one more class: the EmulatedApplication. This class allows storing specific information about the programs used to emulate the applications. Each instance is formed by a software description, represented in Description, and an URL (Uniform Resource Locator) pointing to its web page, represented in the URL class.

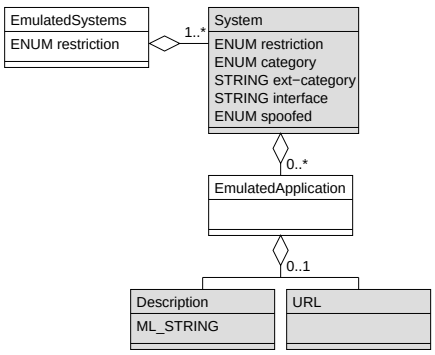

Figure 4: EmulatedSystems and its aggregated classes.

The Honeynet class, that we can see in Figure 5, allows representing information about the honeynet where a high-interaction honeypot may be located. Its type attribute may have three values: physical, virtual and ext-value. The ext-type attribute permits the extension of the type attribute, as will be discussed in Section 3.4. The Honeynet class may have zero or one Description class, that can contain a general description of the honeynet. The other classes may occur zero or more times. The Address class is derived from IODEF and represents the following types of addresses: IP, MAC (Media Access Control), AS (Autonomous System), among others related to a honeynet. The DataCollection and DataControl classes have similar structures and represent data about the technologies used in a honeynet to collect data and control the intruders' activities [18]. The AdditionalData class will be discussed in Section 3.4.

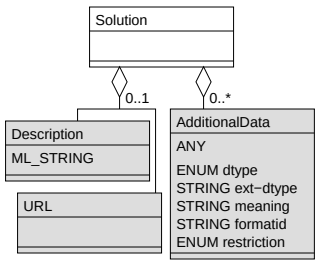

Figure 6: Solution and its aggregated classes.

The Solution class represents, in low-interaction honeypots, the solution used to implement the honeypot. As we can see in Figure 6 an instance of this class is comprised of zero or one instances of the Description class, where we can store a general description of the solution, and of the URL class, that can hold a URL to the application site. The AdditionalData class can hold additional information needed and will be discussed in Section 3.4.

The ArtifactData, seen in Figure 7, represents artifacts

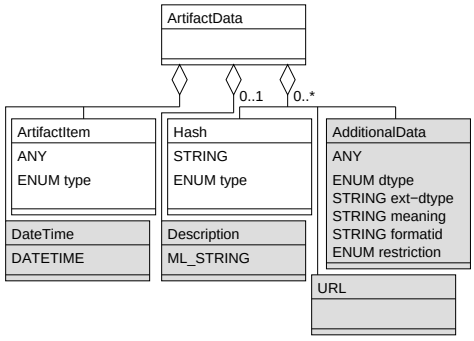

Figure 7: ArtifactData and its aggregated classes.

collected in low- or high-interaction honeypots, together with their capture and analysis information. An instance of this class has exactly one ArtifactItem, where the artifact itself is stored, and the timestamp of the capture, represented in DateTime. In Description we can store a general description of the artifact or its analysis. Each ArtifactData instance may also have zero or more instances of the Hash, URL and AdditionalData classes. The Hash class may store a cryptographic hash or a digital signature for the artifact, this data can be used to uniquely identify each artifact. Its type attribute determines what kind of hash was used, and may have one of the following values: md5, sha1, sha256, sha512, rmd160 or pgp-signature. The URL class can be used in case the artifact is already in public domain and has a URL associated to it. The AdditionalData class will be discussed in Section 3.4.

To exemplify the use of HIDEF, Figure 8 presents a document with the representation of an SSH scan against a low-interaction honeypot.

3.4 HIDEF Extensions

Taking into account the dynamic nature of the computer security area, HIDEF has means for the definition of extensions to the model allowing the inclusion of new characteristics to the data model. As it is an object-oriented model it is always possible to extend it through inheritance, defining subclasses with attributes not present in the super-class. As we will see below, it is also possible to include new classes.

The AdditionalData class, derived from IODEF, acts as an extension mechanism. It allows including information not originally represented in the data model or including new classes. This class permits the inclusion of atomic elements, like integers and strings, or more complex data, like complete XML documents, derived from other Schemas (like IDMEF or IODEF). Its type attribute holds the definition of the data type, and its meaning attribute defines which is the meaning of that data in the HIDEF document context.

```
<?xml version="1.0" encoding="UTF-8" ?>
<HIDEF-Document version="1.00" lang="en"
 xmlns="hidef-1.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<Honeypot restriction="public" type="low-interaction">
 <System restriction="private" category="target">
  <Node>
   <Address category="ipv4-addr">192.0.2.1</Address>
  </Node>
  <OperatingSystem vendor="OpenBSD" version="3.9" />
  <Description>
   Low-interaction honeypot real OS.
  </Description>
 </System>
 <AdministrativeInfo restriction="private">
  <EmulatedAddress category="ipv4-net">
   10.0.0.0/26</EmulatedAddress>
  <Sanitization type="ipv4-addr" real="10.0.0.1"
                sanitized="xxx.xxx.xxx.1" />
 </AdministrativeInfo>
 <EmulatedSystems>
  <System restriction="private" category="target">
   <Node>
    <Address category="ipv4-addr">10.0.0.1</Address>
   </Node>
   <Service ip_protocol="6"><Port>22</Port></Service>
   <OperatingSystem vendor="Linux" version="2.4.7" />
   <EmulatedApplication>
    <Description>
     SSH emulator derived from the dropbear server.
    </Description>
    <URL>http://matt.ucc.asn.au/dropbear/dropbear.html
    </URL>
   </EmulatedApplication>
  </System>
 </EmulatedSystems>
 <Solution>
  <Description>Honeyd</Description>
  <URL>http://www.honeyd.org/</URL>
 </Solution>
 <EventData>
  <StartTime>2006-11-03T16:12:31Z</StartTime>
  <EndTime>2006-11-03T16:12:45Z</EndTime>
  <Flow>
   <System category="source">
    <Node>
     <Address category="ipv4-addr">
      192.168.8.2</Address>
    </Node>
    <Service ip_protocol="6">
     <Portlist>2458,2535,2598,2752</Portlist>
    </Service>
   </System>
   <System category="target">
    <Node>
     <Address category="ipv4-addr">
      xxx.xxx.xxx.1</Address>
    </Node>
    <Service ip_protocol="6"><Port>22</Port></Service>
   </System>
  </Flow>
  <Record>
   <RecordData>
    <RecordItem dtype="string">
    Nov 03 16:12:31 192.168.8.2.2458 > xxx.xxx.xxx.1.22
    Nov 03 16:12:34 192.168.8.2.2535 > xxx.xxx.xxx.1.22
    Nov 03 16:12:37 192.168.8.2.2598 > xxx.xxx.xxx.1.22
    Nov 03 16:12:45 192.168.8.2.2752 > xxx.xxx.xxx.1.22
    </RecordItem>
   </RecordData>
  </Record>
 </EventData>
</Honeypot>
</HIDEF-Document>
```

Figure 8: HIDEF document with data about an SSH scan against a low-interaction honeypot.

There is another mechanism that enable the extension of some attributes defined as enumerated data types. All attributes whose name is of the form “ext-<name>” can be used to extend the attributes of the same “<name>”. An example are the attributes type and ext-type from the Honeypot class (Figure 1). Nowadays the honeypots are

classified in the literature as low- and high-interaction. However, this classification may be expanded in the future. If this happens the type attribute can be extended by assigning to it the value "ext-value", and assigning a "new-value" to the optional attribute ext-type, representing a new type of honeypot.

4 Conclusions and Future Work

The data collected by honeypots and honeynets, and the information about their technologies, are complex. This requires an equally complex data structure to represent them. In the HIDEF model these data are represented in a structured way, giving the data appropriate semantics. This enables quick access to the data and facilitates the interoperability and the processing automation.

The proposed format is compatible with the IODEF and IDMEF standards, allowing data collected by honeypots to be more easily correlated with data from IDSs and network security incidents. Additionally, the data collected from honeypots and represented in HIDEF format can easily be exported to the IODEF format, allowing them to be notified as a security incident. This is possible mainly because HIDEF uses a class derived from the IODEF EventData to represent network traces and logs.

The next steps include the development of a client-server system that will allow the creation and exchange of HIDEF documents in a secure way. This system will also validate and import these data into a database system. A case study will be implemented to exchange and correlate data from a low-interaction honeypot network and a honeynet, both already operational.

References

[1] Extensible Markup Language (XML). http://www.w3.org/XML/. Access date: Oct 27, 2007.

[2] XML Schema Part 2: Datatypes Second Edition – W3C Recommendation. http://www.w3.org/TR/xmlschema-2/, October 2004. Access date: Oct 27, 2007.

[3] Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT.br). Incidentes Reportados ao CERT.br. http://www.cert.br/stats/incidentes/. Access date: Oct 27, 2007.

[4] CERT Coordination Center. CERT/CC Statistics 1988-2007. http://www.cert.org/stats/cert_stats.html. Access date: Oct 27, 2007.

[5] Conti, G. and Abdullah, K. Passive visual fingerprinting of network attack tools. In *Proceedings...*, pages 45–54. VizSEC/DMSEC '04: 2004 ACM Workshop on Visualization and Data Mining for Computer Security, ACM Press, October 2004.

[6] Danyliw, R., Meijer, J., and Demchenko, Y. RFC 5070: the incident object description exchange format. http://www.ietf.org/rfc/rfc5070.txt, December 2007. Access date: Feb 2, 2008.

[7] Debar, H., Curry, D., and Feinstein, B. RFC 4765: The intrusion detection message exchange format (IDMEF). http://www.ietf.org/rfc/rfc4765.txt, March 2007. Access date: Oct 27, 2007.

[8] Göbel, J., Hektor, J., and Holz, T. Advanced honeypot-based intrusion detection. *;login: The USENIX Magazine*, 31(6):17–25, December 2006.

[9] Harold, E. R. and Means, W. S. *XML in a Nutshell, A Desktop Quick Reference*. O'Reilly and Associates, Inc., 2nd edition, June 2002. ISBN-10: 0-596-00292-0.

[10] Hartmeier, D. Design and Performance of the OpenBSD Stateful Packet Filter (pf). In *Proceedings...* FREENIX Track: 2002 USENIX Annual Technical Conference (FREENIX '02), June 2002.

[11] Hoepers, C., Steding-Jessen, K., Cordeiro, L. E. R., and Chaves, M. H. P. C. A National Early Warning Capability Based on a Network of Distributed Honeypots. In *Proceedings...* Annual FIRST Conference on Computer Security Incident Handling, June 2005.

[12] Hoepers, C., Steding-Jessen, K., and Montes, A. Honeynets Applied to the CSIRT Scenario. In *Proceedings...* Annual FIRST Conference on Computer Security Incident Handling, June 2003.

[13] Milo, T., Abiteboul, S., Amann, B., Benjelloun, O., and Ngoc, F. D. Exchanging intensional XML data. *ACM Trans. Database Syst.*, 30(1):1–40, 2005.

[14] Phillips, A. and Davis, M. RFC 4646: Tags for identifying languages. http://www.ietf.org/rfc/rfc4646.txt, September 2006. Access date: Oct 27, 2007.

[15] Provos, N. A virtual honeypot framework. In *Proceedings...*, pages 1–14. USENIX Security Symposium, August 2004.

[16] Provos, N. and Holz, T. *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Addison-Wesley Professional, 1st edition, July 2007. ISBN-13: 978-0321336323.

[17] Spitzner, L. *Honeypots: Tracking Hackers*. Addison-Wesley Professional, 1st edition, September 2002. ISBN-10: 0321108957.

[18] The Honeynet Project. *Know your Enemy: Learning About Security Threats*. Addison-Wesley Professional, 2nd edition, May 2004. ISBN: 0-321-16646-9.

[19] Wood, M. and Erlinger, M. RFC 4766: Intrusion detection message exchange requirements. `http://www.ietf.org/rfc/rfc4766.txt`, March 2007. Access date: Oct 27, 2007.