Designing and developing an Application for Incident Response Teams*

Kees Leune and Sebastiaan Tesink
{kees,sebas}@uvt.nl
Tilburg University, Infolab
The Netherlands

May 9, 2006

Abstract

Computer security incident response teams need to track incidents as they develop. To support day-to-day operations, teams need to be able to generate quick overviews of ongoing incidents, and they must be supported in their daily work by automating as much routine work as possible. AIRT is a web-based system to provide incident tracking capabilities to computer security incident response teams. Its design goals include to provide a comprehensive incident management console, ability to quickly associate external teams with IP addresses, the ability to record an incident in 30 seconds after receiving it, provisions for PGP signed mail, and more. This paper presents AIRT, its goals, architecture and its functionality.

1 Introduction

The increasingly hostile nature of the traffic flowing over the Internet has prompted many organization to rethink their computer and network security policies, and to establish formal computer security incident response teams (CSIRTs). While guidelines for establishing new teams are well documented, certain aspects lack

*Work on AIRT has been partially funded by a generous contribution of SURFnet

adequate attention. While most authors pay attention to the need for organizational embedding, proper documentation of incident response procedures, well-established workflows, etc., one aspect that regularly does not get sufficient attention is the way that procedures and workflows are supported by information systems.

New teams are often required to adopt a corporate standard for workflow monitoring and help desk ticketing. Teams that are not bound by such requirements usually set out to evaluate several workflow engines and helpdesk ticketing applications, and find that while most of the choices are acceptable for either workflow support, or helpdesk functions, computer security incident response teams have additional requirements.

In this paper we outline a new web-based information system that may be used by incident response teams to manage incidents, assess evolving situations and formulate and implement incident response tactics. The system has been developed with the support of SURFnet, and is currently in use within SURFnet-CERT and several of its constituency members. The goal of the project was to *design, develop and document a support system for computer security incident response teams that would function as a basic "operations console"*. A number of requirements were formulated:

- Interoperability with existing tools and programs must be possible
- Open development model and software license
- Web-based user interface
- Focus on small to medium-sized incident response teams (up to 250 incidents per day)
- Ability to create an incident record in less than 30 seconds after reception of an incident report.

Additionally, all software that was developed has well-documented and flexible interfaces for dealing with existing applications and tools.

For partial ensurance of the interoperable nature of the application, the development model that was chosen is open in many different senses of the word. First, it is open because all interfaces with the system are documented and documentation is freely available to anyone who is interested. Second, the development model is open because the program will be distributed as Free Software. One of

the properties of Free Software is that code must be made available freely. Lastly, the project is open because bugs, feature requests and future development will be discussed openly via a community website [1].

To ensure as much flexibility in the way that end-users interact with the application as possible, the choice was made for a web-based user interface. The web-based user interface consists of two parts: a graphical user interface using HTML for human users and a machine-usable web services interface.

Large incident response teams, especially those that play the role of coordination center, are dealing with vast information flows. Most of that information will be transported in the form of email, but additional flows are imaginable too. With the exception of the larger Internet Service Providers and backbone operators, most teams do not require provisions for such volumes. As a result, development focuses on a system for small and medium-sized incident response teams. By this, we mean incident response teams that have an incident volume of at most two hundred and fifty incidents per day.

To reduce the workload of incident handlers, we have set the goal that an incident must be created within less than half a minute from reception. The point that an incident is received is considered to be the point after the handler has successfully logged in to the system, and has taken knowledge of the nature of the incident. In other words, in case of an incident that is reported via email, we consider reception of the incident the point where the email has been read, rather than the point where it has been delivered to a mailbox.

In the following sections, we will discuss AIRT, the Application for Incident Response Teams. In Section 2, we discuss the architecture of our solution, the technology that is used and some implementation details. In Section 4, we outline how using AIRT is experienced by the teams that have worked with it, and what the principal benefits are. Section 5 briefly discusses related initiatives and in Section 6, a number of planned extensions are outlined. Finally, section 7 contains the summary and conclusions of this paper.

2 AIRT

Before the active development of AIRT started, a number of design objectives were set. The most important ones were that the system should scale well enough to still be usable when the team handles up to one hundred incidents a day and

[1] http://www.airt.nl

that processing an incident should be easy and fast. As a matter of fact, teams that now use AIRT confirm that volumes that exceed these numbers do not pose problems, and that incident creation takes less than half a minute, as was stated as a goal.

AIRT is designed around the concept of an incident. Incidents are categorized by an incident type, an incident state, an incident status and have some kind of log information attached to them. Each incident affects some IP addresses, and can be associated with one or more persons. IP Addresses are part of networks, which belong to constituencies and each constituency is managed by constituency contacts.

A basic premise of AIRT is that everything you see should be customizable by the team that uses the application. In other words, all the characteristics of an incident, as well as networks, constituencies and constituency contacts must be fully configurable. As mentioned, incidents are characterized by an incident type (e.g., copyright, ddos, virus, compromise, spam, portscan, etc), an incident state and an incident status. AIRT has been developed as an open application, which when it matures, must be able to communicate directly with other AIRT instances. The benefits of this are obvious, as it eliminates the need for human users to copy and paste incident details into mail messages, and the receiver of the message to do the same thing over again.

By incident status, we mean a simple label which is meant to be communicated with other teams (or end-users) and reflects the level of activity that can be expected of the team. AIRT is shipped with three default states: open, closed and stalled. The incident state represents a phase in the internal workflow of the team, and it is not meant for communication to external teams. Example states are inspection requested, inspected, blocked, forwarded, etc. While the distinction between state and status might appear to be a mostly theoretical one, it has been proven to be very useful. For example, UvT-CERT uses it to regularly generate router configuration files based on the criteria that all incidents with status open or closed and which are in the state blocked must be null-routed.

By associating incidents with IP addresses, and providing AIRT with the knowledge of networks, constituencies and constituency contacts, finding the correct team to complain to becomes a very easy process. In the situation of UvT-CERT, which coordinates incident response for a university which has five schools which all have decentralized computing, finding the appropriate person became very easy, as shown in figure 1.

When a network is not yet known to AIRT, it will attempt to find a contact via `whois` queries. Unfortunately, different `whois` servers are still inconsistent

Detailed information for host ig0213.uvt.nl

Search results for the following host:

IP Address : 137.56.127.213
Hostname : ig0213.uvt.nl
Network : Infolab users (137.56.127.192/26)
Constituency : Infolab, (Dept Information Management)

Constituency Contacts

Name	Email	Phone
Leune, Kees	c.j.leune@uvt.nl	+31 13 466 2688

Figure 1: Finding constituency and constituency contact for an IP address

in the way that information is presented, which makes automatic parsing of the search results often a difficult problem.

3 Architecture

AIRT has been designed and implemented as an open system with well-defined interfaces. It is possible to interact with the system via a multitude of channels. The most common interface is the HTML-based interface which is used by users with web browsers. In addition to the HTML interface, AIRT also provides a web services interface. Web services technology implements the vision of service-oriented computing, which is a new software design philosophy which strives to make web-based applications self-describing, loosely coupled and distributed applications. Lastly, AIRT provides a command line interface.

The web services interface allows AIRT to interact with other data sources and provides an extension mechanisms for plugins and external software. The web services interface is also used by command line interface and by the import queue, which will be discussed later.

Figure 2 shows a graphical representation of the AIRT architecture. Centrally located is AIRT-Core, which provides all the basic functionality of the system related to incidents, users, mail templates, etc. AIRT core can be interacted with via the web services interface and via the HTML interface. The figure also shows two extension mechanisms. The first mechanism is the event mechanism. All AIRT

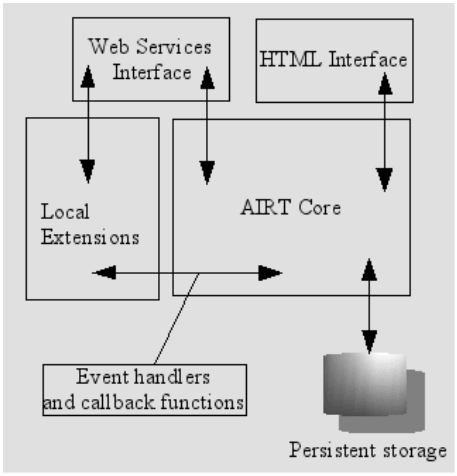

Figure 2: AIRT reference architecture

operations will generate events before and after they are executed. Via the configuration files, it is possible to catch these events and add additional operations to them. The event mechanism is meant to *extend* AIRT functionality. The second mechanism consists of a number of predefined local hooks, which can be used to *modify* AIRT functionality.

3.1 Import queue

As mentioned before, the web services interface is used by the import queue. The AIRT import queue aims to collect structured data from heterogeneous sources and semi-automatically import it into the AIRT system. The import queue consists of an import command line interface, a number of import filters and a queue visualizer. Assume that an upstream team is able to generate messages containing structured content. A prime example of this are the reports sent by the MyNetwatchman project. These messages are received and parsed by a mail filter, which delivers them to the AIRT importer. If the message is parsed successfully, the relevant data is extracted from it and placed in the database.

Next, an AIRT user can inspect the queue and determine whether or not each report should be accepted as a new incident. The import queue is intelligent enough to handle multiple reports about the same source address. If the importer finds an open incident in the database with a corresponding source address, it will

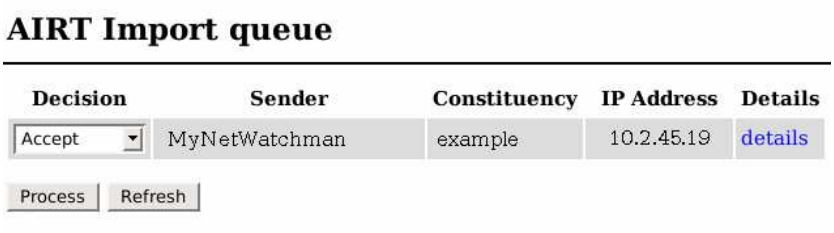

Figure 3: AIRT import queue

offer the ability to import the additional logging into the existing incident.

The import queue can also be used to import incident data that is sent by external teams. The AIRT mail templates provide the ability to include incident data in an XML format that can be parsed by the import queue. The result of this is that the combination of mail templates and import queue provides a convenient exchange mechanism for incident data.

3.2 Mail templates

Email is often the preferred medium for communicating with external teams and end-users. Additionally, many sensors often provide alerts in the form of automated email messages. Consequently, by improving the process by which email is processed (both incoming, as well as outgoing), CSIRTs can enhance their overall incident response process.

AIRT provides support for outgoing email using templates. The outgoing email system features a context-aware variable expansion mechanism and PGP signing support. The mail template component is context-aware because it will carry over the incident data of the incident that the handler is currently working on. This is particularly useful in combination with the variable expansion mechanism. For example, consider figure 4. On the left-hand side is the mail template, which is expanded on the right-hand side.

AIRT provides a wide range of variables which can be expanded in such a way. Since email by itself is an unreliable medium, and it is virtually impossible to authenticate the origin of a message, AIRT provides optional PGP signing support. If an application manager provides a keypair and the location of GnuPG, outgoing messages may be digitally signed.

```
UvT-CERT has detected that
a computer known as

 @HOSTNAME@ [@IPADDRESS@]

contacts honeypot machines at
Tilburg University.
--- Begin logfiles ---

@LOGGING@

--- End Logfiles ---
```

```
UvT-CERT has detected that
a computer known as

evil.example.com [10.2.4.6]

contacts honeypot machines at
Tilburg University.
--- Begin logfiles ---

Source ip  : 10.12.14.16
Source name: evil.example.com
   ......
   ......
   ......
--- End Logfiles ---
```

Figure 4: Email templates

4 AIRT in daily operations

Having described AIRT, the assumptions that were made and the way that it was designed, this section provides a brief overview of how AIRT is used in day-to-day operations. A typical information flow which results in an incident is that the team is notified of suspicious behavior of one of their nodes. Such notification can originate from other teams by means of email, from users by means of a telephone call, or by one of the larger sensor networks (SpamCop, MyNetWatchman, etc.).

Using the import queue, reports originating from well-known sources that provide computer-parsable reports can be processed automatically. Reports enter the import queue and are visually inspected, after which a decision is made whether or not to accept them as incidents. If so, a new incident is created with state 'imported'.

Another common vector for incident creation is to use an IP address or host name as a starting point. If the IP address belongs to a network that is managed by a constituency known to AIRT, new incidents can be created and they will be addressed to the correct constituency contacts. If the IP address belongs to a network that is not managed by a known constituency, AIRT will execute a regular `whois` lookup to find an abuse contact.

For example, consider a common situation as outlined in figure 5. One of

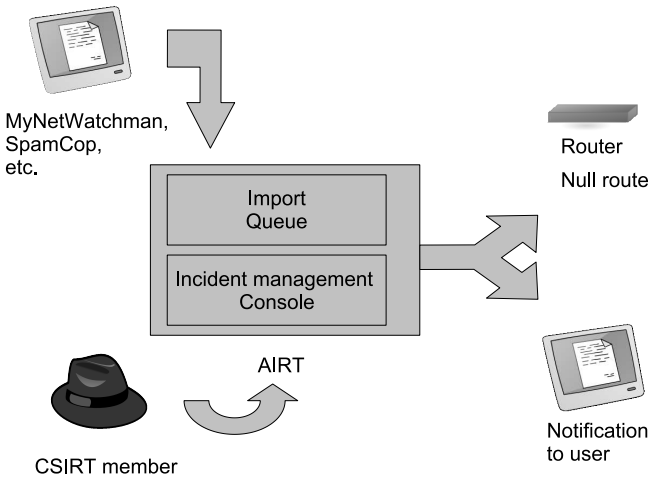

Figure 5: AIRT usage example

the sensor networks sends a notification of alleged network abuse. The import queue processes the report and prepares two incidents. A team member inspects the queue and discovers the two new reports, of which one contains additional information for an IP address that is already being tracked by AIRT as incident. AIRT flags this incident and the team member decides that he wants to add the information of the report to the existing incident, instead of creating a new one. The other incident is indeed new, and the handler decides to accept the incident.

After processing the queue, AIRT will have created a new incident. The handler decides that the incident is serious enough to proceed to block the user's access to the Internet by implementing a null route on the router, and by notifying the user of his decision with a digitally signed message. AIRT has been configured to automatically send out such messages when the handler changes the incident's state to 'blockrequest' and to push a new configuration to the router. The danger that is posed by the offending node is now mitigated to acceptable proportions.

Once incidents are created, they become accessible via the incident management console. The console offers the ability to filter on incident state or incident status, and to sort by any column. As a result, team members can quickly assess a situation and determine their tactics accordingly.

5 Related work

As discussed in section 3.1, AIRT has the ability to import incidents that are delivered to it in well-defined XML format. The AIRT import facilities are meant to provide interoperability to teams with a desire to automatically exchange incident data. As a result, the AIRT interface uses a publicly available and open XML format which hopefully will be adopted as standard by other incident response teams.

Before starting the design process of the Extensible Incident Representation Language (XIRL), we specified a number of requirements.

1. XIRL must be *machine readable*. As a result, XIRL should not have to worry about formatting or any other human-interface specific requirements;

2. XIRL must be *minimal*. In other words, the language should only contain a minimal set of attributes for each incident, such as source and target, time of incident, and log information. Some of the common log information may be represented by simple constructs, such as source IP address, source port, target IP address, target port, protocol used, etc.

3. XIRL must contain *data only*. The explicit goal of XIRL is to exchange incident *data*. Any kind of process information which describes the way that CSIRTs are dealing with the incident is left out of the language.

Rather than trying to re-invent the wheel, we looked at other initiatives to see if they could be used. After an initial inventarization, two open standards were quickly identified and considered; CAIF and IODEF. Both of these standards were commonly referred to while looking at the combination of incident handling, CSIRTs and XML standards.

The Common Announcement Interchange Format (CAIF) is actively being developed at the University of Stuttgart. CAIF is an XML-based format to store and exchange security announcements in a normalized way [CAIF, 2005]. Although the CAIF's focus is on the exchange of security advisories, it can also be used as a basis to develop new document formats since the set of mandatory elements is small [Goebel, 2005]. While looking at the requirements for XIRL, CAIF turned out to be a good candidate for integration: it appears to shares all the goals that were stated for XIRL, however the standard was not adopted in AIRT. After careful consideration we decided that we were not going to use CAIF as a basis for our work. The most important reason was not lack of functionality, but more in the

still unstable nature of the CAIF specification. However, having said that, XIRL is being developed with CAIF in mind, and through using XML namespaces, it is possible to combine CAIF data and XIRL data in a common message.

IODEF is a standard data format for computer security information exchange and it is currently under development by the INCH (Incident Handling) working group of the Internet Engineering Task Force (IETF) [IETF, 2005]. The goals of IODEF are [Jan Meijer, 2004]:

- Increased ease to collaborate with other CSIRTs, on behalf of its constituency, to resolve incidents;
- Increased automation in the processing of incident data, since the obligation of security analysts to interpret free-form textual document will be reduced;
- Decreased effort in normalizing similar data from different sources; and
- A common format on which to build inter-operable tools for incident handling, such as correlation systems that process data from different sites.

While these goals do not appear to be conflicting with the XIRL goals, careful analysis of the IODEF format made it clear that it is does not meet the requirement of simplicity. IODEF is a specification which allows for an overwhelming number of XML-elements to describing an incident. It would, for instance, not be within the scope of AIRT to be able to describe the impact of an incident in the technical impact, the impact with respect to time, the impact in respect to money and the impact with respect to human life, i.e. the number of deaths and/or injuries.

A second reason for not adopting this standard, is the fact that IODEF has not been widely accepted. Had this been the case, the benefits of adopting it as a standard for incident data exchange may have outweighed the fact that the specification is too large.

A full discussion of XIRL is beyond the scope of this paper, but we refer the reader to [Leune, 2005].

6 Future work

A process is currently on the way to expand the AIRT community and to provide interoperability with more tools, such as the SURFnet IDS tool, Switch's netflow analyser NFsen, etc. In addition, we are investigating the possibility to

add S/MIME mail signing and mail encryption to the AIRT-core feature list. An import filter that converts IDMEF data into the AIRT format is currently being tested, and work on multilingual support is progressing.

7 Summary and conclusions

AIRT has been in use in several incident response team for over a year. During that year, the application has gradually grown into a full-fledged application for incident response teams that can be used to easily manage ongoing incidents. It provides teams with tools to quickly assess situations and formulate strategies accordingly. AIRT is most powerful when its import queue is configured in such a way that automatic reports, such as SpamCop and MyNetWatchman reports, are processed automatically. The original design goal of AIRT have all been met, and the application has shown to be easily customizable for integration in existing information technology landscapes. AIRT's future looks bright, as more developments and community efforts are on their way.

References

[CAIF, 2005] CAIF (2005). *Common Announcement Interchange Format (CAIF)*. RUS-CERT, Universitt Stuttgart.

[Goebel, 2005] Goebel, O. (2005). *CAIF Format Specification - Draft*. RUS-CERT, Universitt Stuttgart, 1.2 edition.

[IETF, 2005] IETF (2005). *Extended Incident Handling (inch)*. IETF.

[Jan Meijer, 2004] Jan Meijer, Roman Danyliw, Y. D. (2004). *The Incident Data Exchange Format Data Model and XML Implementation*. Internet Engineering Task Force, 2 edition.

[Leune, 2005] Leune, K. (2005). Extensible incident representation language. Technical report, Tilburg University, Infolab.