

# **OTRS: Issue Management System Meets Workflow of Security Team**

**Pavel Kácha**

12. 6. 2006

**Keywords:** OTRS, CSIRT, security, incident, ticket management, issue management

## **1 Abstract**

Hand in hand with computer security incident response team growth the need for more scalable and flexible support tools is needed. The aim of this article is to document the review and selection of issue management systems, suitable for middle sized CSIRT team and subsequently confront the most suitable one (OTRS) with real needs, and consider its strengths and weaknesses.

## **2 Introduction**

This review intends to document the journey to find the issue (ticket) manager suitable to manage security incidents in a small CSIRT team. This article does not attempt to provide a comprehensive analysis of all issue management systems in the wild and all incident management problems that a growing security team is likely to face. Yet, as we have long searched for a system which would most closely fit our expectations, and we have by no means arrived at a definite solution yet, we feel that our findings, troubles and the review may be of some help for teams which find themselves at the start of the same journey, be it for use as a guideline or as a springboard for their own research.

## **3 Typical security incident report lifecycle**

In order to refine the basic need of any CSIRT team, it is necessary to analyse a typical security incident report lifecycle.

Once the report is received, its relevancy is assessed and if necessary, additional information is requested. Next, reports are categorised according to the

networks affected and resent to their respective administrators, after consulting internal databases or *whois* information). The administrator responsible then communicates directly with the original complainant (if needed) and finds a solution. If everything goes fine, from this point onwards, CSIRT acts only as a spectator and a recorder. According to the importance of the report, the relevant administrator responsible may be contacted and response requested where CSIRT has not been informed about the resolution in time. Afterward, the report is then finalised and marked with the appropriate outcome - solved, no solution needed, no response from administrator, etc.

The person who handles the report should be able to find reports on previous incidents related to the affected IP address or network block easily.

Actions taken based on ticket communication can have significant consequences and data flowing around in emails may be sensitive. Accordingly, communication is usually encrypted or at least cryptographically signed so that its source can be verified.

In the course of the report lifecycle, indications may show that the incident is in fact a part of another incident or a larger event or conversely that the incident consists of several unrelated events. In that case, reports may be grouped together or broken up.

## **4 Needs refinement**

Several requirements for management system may be derived from the analysis of a typical workflow.

- Most communication is carried out through email; the ticketing system must therefore be able to cope with large amount of incoming email, generate email messages correctly and actively support keeping messages under the relevant ticket, if possible even beyond usual (error-prone) capabilities of email threading.
- We should be able to split and merge individual reports.
- Searchable metadata. This can be partially substituted by folders or queues (for network blocks), but the best solution would be to link together incidents and IP addresses.
- Unambiguous ownership of manager. This could be considered as a standard feature of most issue management systems nowadays, but as several unusual solutions are considered in this text, I mention it explicitly.

- Templates for routine answers/forwards, possibly programmatically modifiable (automatic inserting of IP address, etc.).
- Ability to handle and produce PGP signed/encrypted messages, S/MIME would be convenient.
- Reasonably intuitive interface, not inhibiting, as lightly skilled staff will work as forefront prior to seasoned security team.
- It should be software libre, or reasonably open source. "Should" means almost "must" here because no system will fit flawlessly, and home adjustments will be necessary, not talking about high demands on security and quick response (even "self-made") around security flaws.

## **5 Current state**

The starting point from which we begun looking for a more supportive approach is the traditional email and shared IMAP folders. Several report managers may view the same set of IMAP folders - changes made to the folder by someone else are instantly visible in all modern IMAP clients.

The messages are stored in a hierarchy of folders according to their state and affected networks.

The strength of this approach is the ability to handle signed and encrypted messages easily, be it PGP or S/MIME. This functionality is usually an inherent feature of latest email clients. Some clients even support templates adequately.

The weaknesses include complicated linking of a particular message with its author and threading and merging of messages belonging to one case. Standard email capabilities of message-id and references are often broken, be it by obsolete email clients and remailers or by users during chain of forwards of organizational hierarchy cruising. This includes also the inability to track split and merged reports.

## **6 The selection**

We looked through dozens of systems and reviewed and tested half a dozen of them. For brief reviews and notices see the attachment.

RTIR, a tool created especially for incident response teams, adds operational complexity suitable in particular for large enterprise security teams. Its development model seems unpredictable, as Best Practical Solutions directs its effort

mainly to its flagship product - Request Tracker, and RTIR, which is based on RT, does not keep up with RT development.

The recent growth in open source development community needs has initiated a number of bug tracking projects with sound and dynamic groups of developers created around them. The fact that they are strongly development-orientated, with centralized architecture and weak support for external communication may be seen as their drawbacks.

Another active community has grown around the relatively new Python programming language. Several ticket systems have been developed based on Python. If we put aside those based on complex frameworks (Zope) which carry the burden of nontrivial management with them, RoundUp issue tracking system is worth keeping an eye on, and if it successfully passes its infancy and design shake up period, it may become a viable contender.

The most interesting project so far is OTRS (Open source Ticket Request System). Even though it has its shortcomings (no forwarding templates, rather slow development, some performance issues), it is worth to continue its exploration.

For now we have decided (while we continue with our opened up IMAP based workflow) to start the testing period of OTRS and investigate further options for implementing of missing bits and pieces.

## **7 OTRS discussion**

The decision has been made at last. The most complete project in terms of small security team needs is clearly OTRS. Below is a more detailed discussion of its strengths and shortcomings in the light of our (incident handling) needs. For an overview of other revised and tested systems, see attachments.

### **7.1 Basics**

OTRS workflow is based on the separation of roles of customer, agent and administrator. A customer is usually any client, authorised only to raise issues and insert notes to the system but has no right to change the status of the ticket nor any other ticket properties. Where anonymous access is allowed, the author of any ticket raised who is not authenticated is considered as customer. Agents are any staff authorised to change the status of the ticket or the associated data. Administrators are the clients of system authorized to change the system configuration, to create and delete queues and add/modify users and their rights.

As rights can be assigned with a detailed granularity, division into these three

groups is not mandatory. Agents may have more refined or broadly defined rights compared to the basic setup.

Sometimes, the need can arise for a low privileged account for third party administrators to enable them to comment on the ticket from within the OTRS interface. The common practice is to generate a unique one-time URL for any forwarded incident information. Considering the OTRS's extensible template system, our first estimate is that it should be possible without any deep invasive changes to the code.

## **7.2 Tickets**

Each ticket which can be created through web interface or by sending an email (if properly configured), consists of a sequence of articles, in fact notes, added to the ticket by a customer or an agent, possibly accompanied with a change in status.

The ticket keeps a complete history of the changes made to it, either by human interference or through some automatic means (timeout, or based on some changes of system internal status). The ticket can be split into two, possible independent, cases, and more tickets relating to one case can be merged.

It is typical for big issue tracking systems that every electronic mail generated by system is supplied with a unique article ID that is injected in parseable form into the subject of the message. The system thus does not have to rely on shaky In-Reply-To/References use in email headers which often get lost on its way forth and back in the course of multiple remailing. The subject of the message (or its beginning at least) is usually in some form preserved in the new subject and the ID can be guessed. Where the ID corresponds with some previously generated ID, the mail can be more or less reliably paired with the original ticket.

Each article is in fact an email message in the RFC 2822 format, in the same form in which it was received (or generated). That allows for a seamless integration of signatures and encryption - OTRS in that way utilizes existing standards, both S/MIME and OpenPGP.

Saving messages in the native format is an ideal solution for archiving security team communication. The message does not need to be reconstructed; the binary image of the message is not tampered with, and can be used for security data mining, origin analysis, or used as evidence, especially when supplied with the electronic signature.

Aside from usual data, the ticket can bear an arbitrary name/data pairs. This metadata can be unalterably named by the administrator, or left changeable for the storing of any information that seems to fit in the time of the creation of the article.

For the CSIRT usage, these metadata seem to be ideal for storing the links of the incident with a particular network block, and lodgement of IP addresses and contact information extracted from mails or from external databases (NIC, RIPE). All this metadata can be used as a criterion for searching.

### **7.3 Queues**

It is also typical for this type of systems that tickets are organized into several queues that can be created by the administrator and connected with particular users with defined rights. The typical scenario in the security team could be two queues: incoming one which would be managed by the "first line" basic-trained personnel who are able to solve or delegate via mail the basic types of incidents. The remaining ones would be moved into another queue, managed by specialists and highly-trained staff who can then thus focus only on important or unusual incidents.

### **7.4 Drawbacks**

OTRS has a powerful templating system which enables it to prepare an outgoing mail in a runtime - it consists of an introduction and signature (which are appropriate for the particular queue) and the main body (chosen when replying). The drawback is that it supports templated replies but does not support forwards. Forwarding of incidents is an essential feature of any security team - information needs to be passed to constituency and subnetwork administrators who do not have access to central OTRS. There are many reasons justifying not pushing OTRS as the central authority, either because customers/subproviders have their own systems and processes and are not willing to take additional burden or because managing authorized staff from various companies would not be administratively feasible.

OTRS stores articles in the form of the original mail and in general has a very strong support for accepting and generating a native email. The system only supports forwarding of mails as inline (possibly reformatted) text which eliminates the possibility to check the source (header values) or the electronic signature of the original message. Authenticity of messages is often considered as a standard feature by security professionals so this could be viewed as a drawback.

The performance of the testing setup does not seem to be perfect; the response is rather sluggish, even on a nearly empty database. But the documentation states no major performance problems and we assume that with some additional research and setup adjustments it should gradually improve.

The project seems rather large and live, although only a few developers seem to work on it actively. Mailing lists are live but mostly only trivial questions are answered. Many more complex requests or problems remain unresolved.

That leaves us with the question whether the author is willing to accept some alterations made by us to reduce the burden to maintain in-house patches, since some of adjustments are only possible by altering the main codebase.

## 8 Conclusion

Finding a tool which would be an added value to the incident response team and would not have any significant drawbacks is by no means an easy task. As it turns out, no ticket management tool is readily usable for small or middle sized teams. Even the most advanced projects are connected with nontrivial management or programming requirements.

For now we have made a promising decision.

## 9 Attachments

### 9.1 Overview of more closely considered systems

Following systems were chosen to undergo closer testing from all analysed systems (see later for list).

#### 9.1.1 OTRS (Open source Ticket Request System)<sup>1</sup>

- Perl,multiple RDBMS
- ticket data and notes are based on email format, thorough mail support
- PGP and S/MIME support
- support for subject ID mail responses pairing (not relying on "unreliable" means in mail headers)
- main code mostly one man show, but looks like community is starting to grow
- templates support for replies, NOT for forwarding
- arbitrary ticket metadata
- localization
- various authorization and user database schemes (SQL, LDAP, remote HTTPAuth)

---

<sup>1</sup><http://otrs.org/>

- interface based on templates, customizable independently from main code
- fulltext searching, searching over metadata
- intuitive interface

### 9.1.2 Trac<sup>2</sup>

- Python, SQLite, Subversion
- dynamic, active community
- but heavily oriented to bug tracking and software development
- crude mail support
- no crypto

### 9.1.3 Mantis bug tracking system<sup>3</sup>

- PHP, MySQL
- dynamic, active community
- heavily oriented to bug tracking
- mail notifications only
- no crypto

### 9.1.4 RTIR (Request Tracker for Incident Response)<sup>4</sup>

- Perl/Mason, multiple RDBMS
- enterprise oriented
- very complex administration
- handles mail
- PGP only on incoming side, used for authorization
- unnecessarily complex incident report workflow for small team
- metadata (IP, network blocks)

---

<sup>2</sup><http://www.edgewall.com/trac/>

<sup>3</sup><http://www.mantisbt.org/>

<sup>4</sup><http://www.bestpractical.com/rt/>

- WHOIS support (albeit with issues)
- does not work with RT 3.4, although compatibility is stated
- looks dead - scarce information at both BestPractical and JANET-CERT

### 9.1.5 RoundUp<sup>5</sup>

- Python, several RDBMS
- ambitious design
- now reengineering and rewrite is considered
- requires nontrivial management
- no crypto

### 9.1.6 Mozilla Thunderbird in cooperation with shared IMAP<sup>6</sup>

- mentioned because of its extensibility, and potential to overcome threading and templating issues
- problem with metadata (IMAP offers no support for embedding of various data, only not widely used IMAP labels)
- problem with action author identification (solvable with new messages, but not for deleting/moving/tagging)
- problem with managing of references (threading)
- templates can be successfully solved with Quicktext extension<sup>7</sup>

## 9.2 Overview of remaining reviewed systems

The following systems were analysed and their strengths considered. Along with the name, the link to the main WWW page, system platform and some highlights which played major role in subsequent considerations are mentioned. Even some older, obsolete, or only remotely related projects were considered, as they could serve as the base or skeleton for home modifications.

---

<sup>5</sup><http://roundup.sourceforge.net/>

<sup>6</sup><http://www.mozilla.com/thunderbird/>

<sup>7</sup><http://extensions.hesslow.se/quicktext/>

### 9.2.1 Issue Tracker<sup>8</sup>

- PHP, PostgreSQL, MySQL
- mail facilities separated from main function
- no crypto

### 9.2.2 IssueTrackingProduct<sup>9</sup>

- Python, Zope
- mail notifications only
- no crypto
- spartan but well-arranged

### 9.2.3 Issue Management Tool<sup>10</sup>

- Perl, plain files
- mail facilities separated from main function
- no crypto

### 9.2.4 phpticket<sup>11</sup>

- PHP, MySQL
- homegrown hobby project
- small, elegant
- no mail functions

### 9.2.5 batts (Barnhard Associates Trouble Ticketing System)<sup>12</sup>

- Perl, MySQL
- purely mail based
- no crypto
- one man show
- probably dead (2003)

---

<sup>8</sup><http://www.issue-tracker.com/>

<sup>9</sup><http://www.issuetrackerproduct.com/>

<sup>10</sup><http://www.majgaj.com/imt/>

<sup>11</sup><http://downshift.org/code.php>

<sup>12</sup><http://www.xisp.net/batts/>

### 9.2.6 Ticketsmith<sup>13</sup>

- PHP, MySQL
- mail oriented
- small, elegant
- homegrown hobby project
- probably dead (2001)

### 9.2.7 whups<sup>14</sup>

- PHP/Horde, MySQL, PostgreSQL
- no mail support
- bug tracking oriented

### 9.2.8 Keystone<sup>15</sup>

- PHP, MySQL
- enterprise oriented (asset and customer management)
- mail notifications only

### 9.2.9 phpSupport<sup>16</sup>

- PHP, MySQL
- accepts incoming mail, but reassignment possible only to internal users
- barely alive

### 9.2.10 DCL (Double Choco Latte)<sup>17</sup>

- PHP, multiple RDBMS
- mail notifications only
- enterprise oriented (work orders)

---

<sup>13</sup><http://www.voxel.net/projects/ticketsmith/>

<sup>14</sup><http://www.horde.org/whups/>

<sup>15</sup><http://www.stonekeep.com/keystone.php>

<sup>16</sup><http://phpsupport.jynx.net/>

<sup>17</sup><http://dcl.sourceforge.net/>

### 9.2.11 **frontdesk**<sup>18</sup>

- Perl, plain files
- purely mail based
- old, probably dead (1998)

### 9.2.12 **JitterBug**<sup>19</sup>

- C, plain files
- knows to track mail replies and follow-ups
- bug oriented
- no crypto
- officially dead

### 9.2.13 **Teacup PRMS**<sup>20</sup>

- Perl, plain files
- handles mail
- no crypto
- enterprise oriented
- old, probably dead (2000)

### 9.2.14 **Helpdesk (Central Manchester CLC helpdesk issue management system)**<sup>21</sup>

- PHP, PostgreSQL
- mail notifications only
- no crypto
- old, probably dead (2003)

---

<sup>18</sup><http://www.gnacademy.org/tech/dbedit/frontdesk.html>

<sup>19</sup><http://samba.anu.edu.au/cgi-bin/jitterbug>

<sup>20</sup><http://www.altara.org/teacup.html>

<sup>21</sup><http://helpdesk.centralmanclc.com/>

### 9.2.15 Mantis Helpdesk<sup>22</sup>

- PHP, plain files
- no mail support
- still in its infancy
- no connection with Mantis bug tracker<sup>23</sup>!

### 9.2.16 OcoMon<sup>24</sup>

- PHP, MySQL
- all documentation and pages in Portugal only

### 9.2.17 Techtables<sup>25</sup>

- PHP, PostgreSQL, MySQL, Gentle-DB
- in its infancy
- probably dead, no contacts

### 9.2.18 PHPTasks<sup>26</sup>

- PHP, MySQL
- homepage dead
- probably dead, last SourceForge info from 2002

### 9.2.19 Gedeon<sup>27</sup>

- PHP, PostgreSQL
- all documentation and pages in French only

---

<sup>22</sup><http://mhelpdesk.sourceforge.net/>

<sup>23</sup><http://www.mantisbt.org/>

<sup>24</sup><http://ocomonphp.sourceforge.net/>

<sup>25</sup><http://techtables.sourceforge.net/>

<sup>26</sup><http://phptasks.sourceforge.net/>

<sup>27</sup><http://gedeon.u-bordeaux.fr/>

### 9.2.20 WebCall<sup>28</sup>

- Perl, MySQL
- in its infancy
- one man show
- no contact to author

### 9.2.21 WREQ<sup>29</sup>

- Perl, DB1
- complex, ambitious
- but mostly one man show
- obscure timeframe - updated 2000, then 2005
- no crypto

### 9.2.22 TrainWREQ<sup>30</sup>

- Perl, DB1
- WREQ<sup>31</sup> fork
- dead 2002

### 9.2.23 PEST<sup>32</sup>

- PHP, MySQL
- non-functional web
- probably dead

### 9.2.24 oTasks<sup>33</sup>

- PHP, MySQL
- dead 2002

---

<sup>28</sup><http://myrapid.com/webcall/>

<sup>29</sup><http://www.math.duke.edu/%7Eyu/wreq>

<sup>30</sup><http://www.owlriver.com/projects/trainwreq>

<sup>31</sup><http://www.math.duke.edu/yu/wreq>

<sup>32</sup><http://sourceforge.net/projects/pest>

<sup>33</sup><http://otasks.sourceforge.net/>

### 9.2.25 **EdenCRM**<sup>34</sup>

- JAVA, IMAP
- client-server
- in its infancy - work only on client part now
- dead 2001

### 9.2.26 **urqm**<sup>35</sup>

- C, SQLite
- in its infancy
- supports mail
- no crypto

### 9.2.27 **PHPHelpdesk**<sup>36</sup>

- PHP, MySQL
- no mail
- officially mostly dead

### 9.2.28 **openTicket**<sup>37</sup>

- PHP, several RDBMS
- dead 2001

### 9.2.29 **BugIn**<sup>38</sup>

- PHP, several RDBMS
- no mail
- probably dead 2004

---

<sup>34</sup><http://edencrm.sourceforge.net/>

<sup>35</sup><http://www.ivarch.com/programs/urqm.shtml>

<sup>36</sup><http://phphelpdesk.sourceforge.net/>

<sup>37</sup><http://openticket.sourceforge.net/>

<sup>38</sup><http://sourceforge.net/projects/bugin>

### 9.2.30 **PHPSAT**<sup>39</sup>

- PHP
- dead before taken off

### 9.2.31 **GNATS**<sup>40</sup>, **Gnatsweb**<sup>41</sup>

- C, miscellaneous, own flat file based DB
- ambitious architecture
- multiple backends (mail, web)
- but gnatsweb seems dead 2003
- no crypto

### 9.2.32 **IssueDealer**<sup>42</sup>

- Python, Zope
- ambitious (wiki)
- heavy platform (nontrivial management)
- no mail support

---

<sup>39</sup><http://sourceforge.net/projects/phpsat/>

<sup>40</sup><http://www.gnu.org/software/gnats/>

<sup>41</sup><http://savannah.gnu.org/projects/gnatsweb>

<sup>42</sup><http://issuedealer.com/>