Abstract: Hand in hand with the growth of the computer security incident response team the need for more scalable and flexible support tools increases. The aim of this paper is to describe and document the architecture and tools developed by us (based on OTRS) as well as approaches usable for mid-size CSIRT team.

Key-Words: OTRS, CSIRT, security incident, issue management, metadata, Bayesian analysis

1 Introduction
In order to refine the basic need of any CSIRT team, let us first analyse the life-cycle of a typical security incident report.

Once the report is received, its relevancy is assessed and, where necessary, additional information is requested. Next, reports are categorized according to the networks affected and forwarded to their respective administrators, after consulting internal databases or WHOIS information. The responsible administrator 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 seriousness of the report, the relevant administrator responsible may be contacted and response requested in case CSIRT had not been informed about the resolution in time. Afterwards, the report is finalised and marked with the appropriate outcome.

A range of tools for issue management exists (see [1] for an overview of suitable ones), however, none of them directly supports the incident report handling work-flow.

2 Basic problems
Let us list basic problems (apart from rudimentary issue management), that the majority of incident report handling teams is facing.

Searchable and reliable metadata. Each incident should be accompanied at least with the IP address of origin, and possibly also with the associated network name and responsible person's contacts. Human analysis and manual metadata extraction is repetitive and rather error prone. Possible automated method would set the basis for more advanced processing.

Incident categorisation. Classification as per the incident type (and consequently its seriousness) forms the basis for statistics and trend analysis.

Incoming traffic sanitization. Spam, virus and backscatter are well known and documented fields of expertise. However in a specific case of incident reports, usual statistical and heuristic methods face unexpected challenges. An incoming incident report itself may contain a sample of spam, virus or unsolicited bounce, and often gets classified as such as a whole. Additional measures are therefore necessary.

Lifetime and bulk checks. Incident reports often get stale, without any downstream response. On the other hand, individual responses may be swift, but the number of incidents of a particular origin may reach suspicious amounts. Simplistic human processing in this case is error prone.

3 OTRS introduction
OTRS (Open source Ticket Request System) is GPL licensed, Perl based trouble ticket (or issue management) system, used as the basis for our applications.

3.1 Tickets
The ticket is composed of a series of articles – textual updates to its state, usually e-mails. The ticket keeps a complete history of the changes made to it, either by human interference or through some automatic means. The ticket can be split into two, possible independent, cases, and more tickets relating to one case can be merged.

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 – in that way, OTRS utilizes existing standards, both S/MIME and OpenPGP.

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

Aside from the 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.

3.2 Queues and states
Tickets are organized into several queues that can be created by the administrator and connected to 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 of 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 focus only on important or unusual incidents.

During its lifetime, each ticket goes through series of states. A state is property completely orthogonal to the queue which can represent important turning points in its history – for example external update, timeout or closing reason.

4 Automated metadata extraction
OTRS is able to store key/data pairs along with the data. These pairs can be arbitrary, but key names can be specified and defined as unchangeable. As we plan to attach at least an IP address, its network pertinence (according to RIPE database) and a responsible administrator's contact, deduced from network block information, we have defined these keys as the fixed first three metadata values, under the names NETNAME, IP, ADMIN.

These fields are editable, so any human operator can spot and correct possible errors. However, data should be pre-filled in some way, to ease the burden of hunting them down and filling them up by hand.

We considered various schemes of an automatic mail analysis. After some testing we finally came up with an automated approach.

An overwhelming majority of incidents contains only one IP address from a particular autonomous system. Our analyser breaks mail into its MIME subparts and searches in subject, main body and all attached data recursively for anything conforming to an IP address format. This can result in a large number of addresses, which have no connection with CSIRT constituency networks, thus we filter out only those belonging into governed network space and remove any duplicities. This usually yields only a single IP address. Where the result contains more addresses, we leave the decision on the human operator at a later stage. Only a human, being aware of the respective context from the mail message, can conclude whether the incident report concerns more IP addresses (and should be separated into two tickets) or whether the second address is a bogus.

Fig. 1: Metadata filled in from extracted IP

Obtained addresses are then screened through the RIPE database (via a custom developed module) and accompanied with RIPE network name and responsible administrator's contact. Resulting information is inserted into mail headers in a form understandable by OTRS which extracts the data and assigns it to the respective metadata fields. This is an example of the generated headers:

X-Otrs-TicketKey1: NETNAME
X-Otrs-TicketValue1: CESNET-BB4
X-Otrs-TicketKey2: IP
X-Otrs-TicketValue2: 195.113.144.199
X-Otrs-TicketKey3: ADMIN
X-Otrs-TicketValue3: abuse@cesnet.cz

We are keeping an eye on IODEF and IDMEF incident and intrusion description formats, as these provide a more precise target identification and standard form of further distribution. However, their proliferation is yet very low, and our approach would have to stay as a fallback even if these formats managed to gain wider audience.

5 Automated incident categorization
Each incident bears its characteristic features and can be categorized as a well known type. Categorization can be managed by human intervention, however if we could achieve a reliable machine classification
beforehand, we would get a valuable clue on how to process a particular incident. Categorization is also necessary for further statistical and trend analyses.

Similar and a well studied problem is spam identification – free form mail text is analysed to decide whether message is allowed to reach the destination mailbox or whether it is malicious or unsolicited commercial message. Statistical methods, based on Naive Bayesian probability analysis which are used for the purpose of spam identification, constitute a two-way decision process.

In general, these methods generate a weighted histogram of words (or of n-tuples of words, or of larger meshes as in the case of the hidden Markov model), based on previous learning history. Histogram values undergo a statistical cleaning and the combined representative value (based on particular method, it can be some kind of average or median value) determines the spam rate of a message.

However, there is nothing inherently two-way in these methods. One of the first Bayes statistics based filters, Jason Rennie's ifile [2], supports n-way filtering. By means of several custom scripts we inserted ifile's classification into the incoming queue. The analyser output is then added as an associated header, and later it is used directly as an incident category in OTRS.

The success of statistical methods stands and falls with quality of learning. Our current work-flow guarantees that at the most one day old incidents are already reviewed and processed by human operator. To eliminate human slips, we use all tickets older than two days as the basis for building up the ifile's database.

5.1 Incident taxonomy

We use a simplistic approach to incident taxonomy. As exhaustive enumeration is not necessary, only incident types of nowadays highest proliferation have been used. As several incident types traces overlap (for example spam is a part of phishing), we declared a rule of the most fitting modus operandi – incident type which contains incident symptoms completely fits.

1. **Spam** – usual unsolicited commercial email.
2. **Bounce** – mail backscatter (usually caused by spam).
3. **Phishing** – spam is used as advertisement for a website which imitates some well known institution in order to gain its clients' personal information (bank account credentials, credit card information).
4. **Pharming** – similar to phishing. More sophisticated DNS attacks are used to cover the redirection of the client to a fraudulent site.
5. **Copyright** – copyright infringement, usually by means of peer-to-peer networks.
6. **Trojan** – malicious code on a server attempting to attack server clients and spread on (by defaced web page or active probing).
7. **Malware** – malicious code on a client workstation, for example keylogger, rootkit or malware as a part of botnet. Trojan and Malware classes partially overlap, in many cases they can be in fact the same code. However we are trying to distinguish the situation where primary function is to spread and attack another machines (Trojan), while Malware mainly collects user data, sends spam, etc.
8. **Probe** – probing servers and networks. Portscan, portsweep, SSH (or other service) scan or unsuccessful attempts to crack service.
9. **DOS** – simple or distributed. Again it partially overlaps with a probe but DOS's primary aim is denying the service, not a compromise.
10. **Crack** – generally any other compromise.
11. **Other** – anything we are not able to classify into previous categories. Meant as a fallback category, which should get reviewed regularly, and the results of which should get incorporated back into this taxonomy.
12. **Unknown** – it is not possible to clearly state the incident type from report (usually some additional clarification from the complainant is needed).

5 Incoming traffic sanitization

The world of email nowadays is widely infected with unsolicited commercial emails, backscatter bounces and various kinds of worms and viruses. Some kind of filtering of incoming mails is therefore necessary to keep amounts of messages to be handled manageable.

However, an incident handling mailbox may face expectable problems – incident report messages themselves can contain samples of spam, bounce or viruses. Usual antispam and antiviral methods fail and some kind of additional treatment is necessary.
5.1 Spam
This section is unfortunately short – we have yet to find a reliable method to distinguish spam reports and spam. The most efficient method so far consists of a manually selected (based on a vast amount of incident reports so far) subject-keyword whitelist. Messages which contain any of these words or phrases in Subject line bypass spam analysis and are allowed to enter the system directly.

The list is maintained in the form of a regular expression:

```
/abuse mail|abuse-mail|abuse of| abuse report|abuse spam|e-mail spam| multiple spam|received spam|report abuse|reported spam|reporting spam| returned spam|spam:spam abuse|spam complaint|spamcop|spam from|spam mail|spammails|spam mails|spammer| spamming|spam-rbl|stop the spam| 
ube|ube-uce|ube/uce|uce:uce|uce-ube| 
uce/ube|ube from|uce from|\[uce\]|\[spam\]|spam received|uce complaint| 
ube complaint|phish|fraud/
```

5.2 Unsolicited bounces
In the case of mail bounces (mail delivery report messages) we have achieved a significant advantage. We know we should only get bounces to messages originated by us. Therefore we are able to keep track of ticket identification numbers (which are injected into subject lines of each message sent). No bounce message (identifiable by an empty Return-Path header line) which does not contain existing ticket identifier younger than two months (to keep machine work low) anywhere in the Subject line or body is allowed to enter the system.

We face a problem here – the format of mail delivery messages [3] is specified very vaguely. There are strict requirements to some of message headers, but Subject and body of the message are completely free form. Some mail delivery agents (mainly certain qmail [4] versions) do not attach enough of the original message to keep the ticket identifier. However according to our analysis conducted on nearly seven thousand of bounce messages shows only 0.5% of such messages which is very acceptable loss ratio. Anyway, the situation with such stubborn agents has generally been improving.

5.3 Viruses
All mail is handled and sanitized for viewing by OTRS. OTRS is a web based application, so security precautions before rendering arbitrary email content into a browser are necessary. The content is completely stripped of scripts and HTML tags, thus mere viewing is secure. The only risk remaining is for the operator to open mail attachments directly, however this can be addressed by a policy or necessary tools (antivirus, anti-malware) can be installed on operator workstations, should the used platform need it.

6 Lifetime and bulk tests

6.1 Lifetime checks
Incident reports handed downstream to responsible security teams or administrators are usually handled on a timely basis, however not all teams have the same expectations, human resources and priorities for particular incident responses. Also, possible human error should be considered. A higher level team must therefore take care of reports during their whole lifetime, ask for updates, take actions when there is no response, and inform the claimant properly.

Human or technical errors are likely to occur even within the CSIRT team itself.

We have developed a set of modules for monitoring open tickets timeline. A ticket, which does not get proper treatment within expected timeframe (2 days in case of downstream team, 30 minutes in case of first-tier local operators) is raised and other members of the team can be informed.

OTRS supports regular check of tickets for some conditions and changing them accordingly but the time can be checked only in relation to the ticket creation, not its update. However the time of the last update is internally stored by OTRS. We have thus created an auxiliary script (running as one of OTRS's cron scripts), which goes through open tickets, checks the time of their last update, and tickets exceeding some timeframe change the state. Timed-out tickets are thus not rotting in the queue until somebody accidentally spots them.

While developing the script, we had to step aside from the usual OTRS ways and combine a direct access to the database with the internal object model. We execute usual SQL statements over the relational repository, which gives us a list of affected ticket identifiers. We then use this list to instantiate real OTRS ticket objects, and use their methods for a full featured manipulation. This ensures that all auxiliary structures are updated accordingly along with history messages.
6.2 Excessive number of reports
We consider some incident reports solely as informational. However, a higher number of common incidents reports on one particular IP address from various sources may foreshadow a more serious problem going on, so seriousness of such incidents should be re-evaluated by human operator.

Again, based on previous work and principles, we created a module for checking unusual amount of incidents from one IP address and sending email notifications if a certain threshold is exceeded.

Results usually correlate with data from the CESNET IDS [5] system.

7 Statistics
Reliable incident source authority identification and automatic incident classification gave us interesting data source for further statistical analysis to be able to compare the incident solving hit rate of our members and constituency, and to review incident type proportion rate trends.

OTRS has some basic statistical module, however its functionality is limited to basic time/state/queue based counts. As the basic data model of OTRS is nicely transparent, fetching more complex data is just a case of straightforward use of conveniently crafted SQL queries. Again, we used our own module with subsequent processing of results and formatting them into a visually and factually convenient output. We were also able to add some data from other sources (annotate institutions with their whole names instead of RIPE shortcuts) or apply some more visually convenient elements.

<table>
<thead>
<tr>
<th>Network</th>
<th>Institution</th>
<th>Incidents</th>
<th>Unknown</th>
<th>Fixed</th>
<th>Warning</th>
<th>IDS</th>
<th>Bcc</th>
</tr>
</thead>
<tbody>
<tr>
<td>CZ</td>
<td>NICE</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>NICE</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>NICE</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>NICE</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>NICE</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>NICE</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>NICE</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>NICE</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>NICE</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>NICE</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>NICE</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
</tbody>
</table>

8 Architecture
Our mail setup accepts mails for certs@, abuse@ and postmasters@ addresses from main CESNET domains.

OTRS is able to accept mails by piping it to its auxiliary bin/PostMaster.pl script or by POP3 polling. We have used the former method, mainly because of its flexibility. During the initial deployment, the mail was dispatched by Postfix directly into this script through alias file. Currently we are using the Maildrop[6] mail delivery agent as a wrapper and caller for metadata extraction, incident categorization and antispam/antivirus/anti-backscatter modules.

Incoming mail is accepted and processed by the usual Postfix setup.

Also a backup mailbox where all incoming and outgoing mail is copied in real time has been set up. We used the usual alias record method for incoming mail and OTRS capacity to duplicate all outgoing mail for outgoing mail:

```perl
$self->{'SendmailBcc'} = 'backup@example.cz';
```

Usefulness of Maildrop shows up in connection with OTRS special headers handling. OTRS understands a definite set of mail headers the content of which can modify its behaviour - choose a particular queue or add some metadata. OTRS itself has a way to classify and define specific actions on mails, but this support is limited, which makes using of the real delivery agent a natural choice.

9 Code
All code and patches to OTRS are released under the GPL license on the CESNET FTP server:


10 Conclusions
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 mid-sized teams. Even the most advanced projects include nontrivial management or
programming requirements.

Our OTRS ticketing system installation currently holds around 3800 tickets, not counting spam and unsolicited bounces. The OTRS interface is used by five core team members as well as six Monitoring centre operators to manage incident reports for several hundreds of assigned network ranges.

Automated metadata extraction and IP address identification through network range sieving works well – the service on CESNET networks and later on across the whole Czech Republic address range shows the need for a manual review of the data for less than 3 % of incident reports only.

The error rate of the statistical incident type deduction also remains similarly low, under 1 %.

Our handmade whitelist worsens the efficiency of the antispam filter; however it is the price to pay for lowering the false positives rate to nearly zero.

Detection of unsolicited bounces also works flawlessly. We are not aware of any loss of valid delivery message on our side.

Timeout robots and excessive incident number detectors help us mitigate human errors and pinpoint possible anomalies in time.

Statistical tools have shown as an immense source of information and as a way to visualize efficiency of particular downstream organizations in combating the electronic crime.

According to the configuration and development experience as well as users’ observations, the work invested into the customizations and the code is paying off, and the course set has worked well so far.

References: