Vulnerability Database User Manual

Ivan Krsul
COAST Techical Report 98-08
COAST Laboratory*
Purdue University
West Lafayette, IN 47907–1398
krsul@cs.purdue.edu

May 16, 1998

1 Introduction

The vulnerability collection at the COAST lab consists of a structured vulnerability database; a collection of several thousand vulnerability related files, mailings and articles; and a directory of vulnerability related tools that include exploit scripts, hacker tools, analysis tools, etc.

This document describes the vulnerability collection, the vulnerability database program, the vulnerability database WWW interface, and related analysis tools.

The directories relevant to the collection are:

The collection: `$VDBCOLL` = `/homes/krsul/vdbase`

The structured database: `$VDB` = `$VDBCOLL/vdb/`

Index file for the vulnerability database: `$VINDEX` = `$VDBCOLL/vdb/Vulnerabilities`

Directory for classifiers: `$CLASSD` = `$VDBCOLL/vdb_classifiers`

File that contains the schema definition for the database: `$VDBSCH` = `$VDBCOLL/vdb_field_list`

Directory for the Java GUI: `$JAVAGUI` = `$VDBCOLL/vdbjavaprogs`

Directory for the source code for the Java GUI: `$JAVAGUIDEV` = `$VDBCOLL/vdb/javaprogs_dev`

Directory for all the perl based programs: `$PERLTOOLS` = `$VDBCOLL/perl`

Directory for the MIME included files: `$MIMEINCLUDES` = `$VDBCOLL/vdb_includes`

Directory for HTTP server and `cgi-scripts`: `$HTTPD` = `$VDBCOLL/httpd`

Directory for the co-word analysis utilities: `$COWORD` = `$VDBCOLL/co-word`

Directory for the general data-analysis utilities: `$GENDSET` = `$VDBCOLL/GenDataset`

We have a collection of vulnerability-related files, tools, exploit scripts, etc. that is in `$VDBCOLL/related_stuff`. There is no documentation on this part of the collection.

The structured database is a collection or records that have a number of fields defined in the file `$VDBSCH`. The fields can be of various types, including text, list, choice list, matrix classifier and hierarchical classifiers. All fields where you can type text are considered to be multi-valued (i.e., you can add as many lines as you wish) and all the

* Portions of this work were supported by sponsors of the COA ST Laboratory.

fields where you can type text support the inclusion of arbitrary MIME parts. See Section 5 for a detailed description of the format of this file.

The fields of records are stored in individual files, using the file system as a database manager of sorts. The database directory has a subdirectory for each field in the schema, and in those directories we have a directory for each database record ID that has the field defined, and in that directory a file (called `V`) that contains the value for the field. If the field has a confidence rating then the value for that rating will be stored in a file called `R` in the same directory.

If the field is a text field and it has an included MIME part, this part will be stored in the `$MIMEINCLUDES` directory in a subdirectory that has the name of the record ID. This allows for multiple fields in the same record to point to a single MIME file that does not need to be replicated.

For example, if the record `sunsmailbug` has information on the field description, and this field has a confidence rating, and a MIME part called `MIME123456` then there will be a file called `V`, a file called `R`, and a file called `MIME123456` as shown in Figure 1.

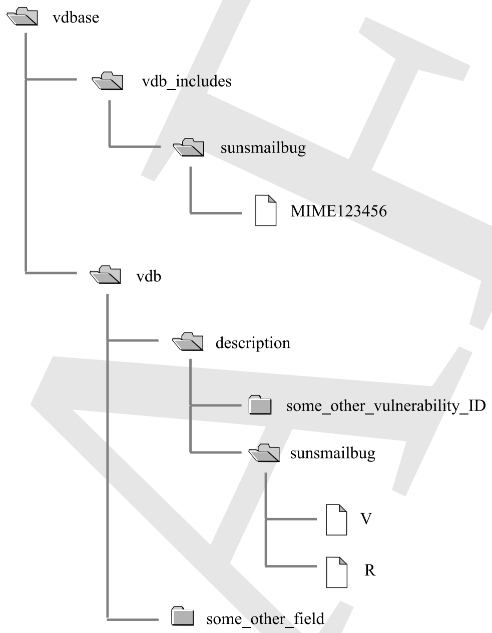

Figure 1: An example of part of the directory structure for the database for a record `sunsmailbug` that has information on the field description, a confidence rating for this field, and a MIME part.

The list of records defined in the database is in the file `$VINDEX`. Each non-comment line has the ID of a record.

2 The WWW Interface

The preferred mechanism for accessing the database is the HTML-based WWW interface for the vulnerability database. This interface provides an easy to use browsing tool that displays records and provides comprehensive search mechanisms.

2.1 Using the WWW Interface

The WWW interface requires that you point a WWW browser to `http://landover.cs.purdue.edu/cgi-bin/vdb`. Because this service is behind a firewall, you will need to be in a machine that is within the COAST internal network or a machine that is authorized to connect to the service. You will also need a user name and a password to access the database.

2.2 Details About the Internals of the Interface

As shown in Figure 2, the WWW interface requires four components that must work together to display information. The database is a collection of several thousand files. The search server keeps a copy of the database cached in memory and waits for connections from the `cgi-bin` script. The `cgi-bin` script receives requests from the user via a WWW browser, translates the request into something that the search server can understand, and relays the reply from the `cgi-bin` script to the browser.

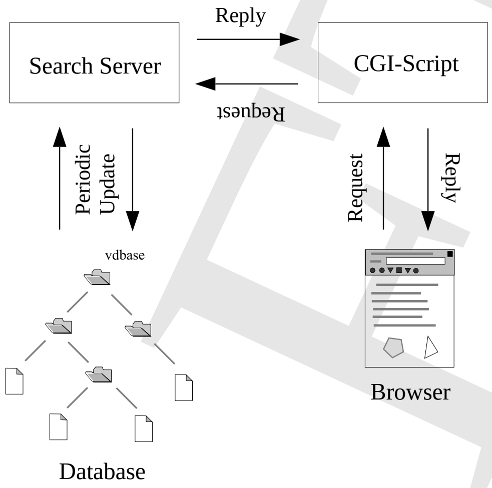

Figure 2: In the WWW interface, four components work together,

This complicated setup is necessary because we need a quick response to queries from the browser and the `cgi-bin` script cannot scan through the files in the database fast enough.

The search server is also described in Section 3.2 and has two functions. It serves as an intermediary between the `cgi-bin` script and the database, and it allows UNIX users to quickly search the database by issuing a command in a shell.

2.2.1 `cgi-bin` Script Details

The `cgi-bin` script is located in `$HTTP/cgi-bin/vdb`. This is a Perl script that waits for commands from a browser, contacts the search server for requests and returns the data provided by the search server to the browser. The script is responsible for providing the HTML code for the headers and footers of every page, but is not responsible for formatting the data provided by the search server.

2.2.2 Search Server Details (With Respect to the WWW Interface)

The search server waits on a well defined port (hard coded into the server) for connections from clients that are coming from the machine where the `cgi-bin` is located. For security reasons only users in `landover.cs.purdue.edu` are allowed to connect to the search server. **Note:** *The assumption is that DNS cannot be hijacked within the lab because we are behind a firewall. Hence, the authorization check is performed by comparing the name of the machine connecting, as returned by a reverse DNS lookup, to the hard coded name.*

The search server responds to the following commands:

`SEARCH Search String` . The search server does a simple string match search on the database and prints in raw text the search results.

`HTMLSEARCH Search String` . The search server does a simple string match search on the database and prints in formatted HTML the search results.

`HTMLPERLSEARCH Perl Regular Expression` . The search server does a perl regular expression search on the database and prints in formatted HTML the search results. Note that special characters in the regular expression are escaped because they are a problem in the web server. Hence, the search server converts these characters back to normal before using the regular expression (with one notable exception: the back tick.).

`LIST Search String` . The search server lists, in raw text, all the records whose title match the string given, if any.

`HTMLLIST Search String` . The search server lists, formatted in HTML, all the records whose title match the string given, if any.

`DUMP Record_ID` . The search server provides a raw text dump of the entire record.

`HTMLDUMP Record_ID` . The search server provides an HTML dump of the entire record. In this dump all URIs and all references to MIME parts are converted to hyper-links.

`MIMEDECODE Record_ID MIME_Part_Name` . Given a record ID and the name of a MIME part, the search server fetches the MIME part, decodes it and returns the result with an associated MIME type header.

`TABLEDUMP` . This command dumps a copy of the database as a table of classifiers and features. The table is CSV. Non-existent values are printed as a single star (`*`). Dumps all CCLASS classifiers as a single value, all LIST classifiers as multiple columns, one for each possible value of the classifier, and TEXT fields as either hasData or `*`.

3 The Java Interface

We encourage you to familiarize yourself with the source code of the interface and its inner workings and, if necessary, to improve it as you see fit. The code is in the `$JAVAGUIDEV` directory. The perl programs that are used as support are in the `$PERLTOOLS` directory.

One *very important* piece of information that you should be aware of: The Java GUI implements record locking to allow multiple people access to the database at the same time. If the program crashes before it released the locks on the records you were editing (which happens when you save or when you exit), it is possible that the next time you use the program you may have to clean the locks by hand. The program will tell you how to do that, but be sure that the lock that you clean by had is yours! It is possible that someone else may be editing that record. You can do that by checking the ownership of the lock-file it created.

3.1 Running the GUI Interface

There is a shell script that sets the Java classpath and runs the GUI interface. We recommend that to run the interface by creating the following alias: "`alias vdbJava $JAVAGUI/runvdbgui`."

Please, don not run the database using another command! The script makes sure that your `umask` is set to 007 and hence the files created by the database will have the correct permissions. The script also makes sure that your classpath contains all the packages needed to run the system. If your `umask` is not 007 then it is possible that the files you create using the Java GUI will not be readable or writable by anyone else. This has the potential for breaking lots of things.

3.2 Searching in the Database from UNIX

The `$PERLTOOLS` directory contains a program called `pattern_match.pl` that takes as an argument a series of words and searches the vulnerability database for matches on those keywords. This perl program loads the database index and calls the `fgrep` program to search every file called `V` in the `$VDB` directory. Searching the database this way can bring any machine to its knees (every search opens thousands of files) so we do not recommend that you use it unless you have no other choice.

The same directory contains two other programs that are particularly useful for searching the database The first is a program called `searchServer.pl` and it essentially loads the entire database to memory and does a pattern search using perl. The server will check to see if new records have been added or if records have been modified every thirty seconds and will load/reload as needed.

The second program is called `searchClient.ps` and it contacts the server and gives it a string to search for and displays whatever the search server returns.

The match is similar to `fgrep` in that the entire text passed has to be matched. Unlike `fgrep`, however, the search string is altered a little bit before its used in the server:

```
  $line = "" if length($line) > 200; # We only want reasonable entries
  $line =~ s/^(\s)*(\S*.*\S)(\s)*$/$2/; # Remove leading and trailing
spaces
  $line = "" if !defined($line);
  $line =~ s/[^a-zA-z0-9\@\_\-\!\%\'\:\.]/ /g; # Special characters
are not allowed
  $line =~ s/\s\s+/ /g; # Replace multiple spaces with single spaces
  $line =~ s/^(\s)*(\S*.*\S)(\s)*$/$2/; # Remove leading and trailing
spaces
```

In its output the search server prints the results with very simple filling so that what you see in the screen is not likely to be in the same format as what you see in the database.

To run the search server `cd` to `$PERLTOOLS` and type:

```
% ./searchServer.pl -p 6768
```

where the `-p` option tells it which port to use. To run the client, `cd` to `$PERLTOOLS` and type:

```
% ./searchClient.pl -p6768 -k "String to search for"
```

where the `-p` option tells it which port the server should use.

Be warned that the server is not a full-blown daemon and should not be run in the background of an obscured window. Create a separate `xterm` window for it and look at it periodically to make sure that no errors are being ignored.

3.3 Using the Database

Running the Java interface will present you a the main screen as shown in Figure 3. Double-clicking on an item on the list of vulnerabilities will display the contents of that record.

The file menu, shown in Figure 4, has options for saving your changes to the database, exiting the database, printing the record as pure text, and exporting the record as a multi-part MIME file.

The view menu, shown in Figure 5, has options for displaying information regarding the field rating system, the classifiers used for the database, etc. Of particular importance is the option for indicating to the GUI interface to eliminate from the record display selected fields. This is particularly useful when fields such as patches and exploit scripts clutter the screen and the user wishes to view records without displaying these fields. When printing records the interface will also only print the fields as indicated by this menu.

The edit menu, shown in Figure 4, has options for editing the current record and adding new records to the database. When you create a new record, a dialog is presented to the user requesting a record ID and title for the new vulnerability. Once this information is presented a new blank record is created for that vulnerability.

When editing a record the GUI interface will open a window, shown in Figure 6, that contains fields and pop-up menus for entering data. Fields that have classifiers are marked by including the name of the classifier in parenthesis under the field name and you can display the classifier by clicking on the name of the field. If the field has a classifier and is a text field then and the you are not required to enter data that matches the classifiers. However, the GUI will complain about it and we strongly recommend that you do stick with well defined choices.

Some fields in the database have associated confidence ratings that give users an idea of how reliable is the data for that particular field. The rating system is as follows:

Value of 0: Item has not been rated. Users will generally make no assumptions about the information in this field. Items with a rating of 0 should not be trusted or used to justify any results.

Value of 1: Item is likely to be a guess or speculation.

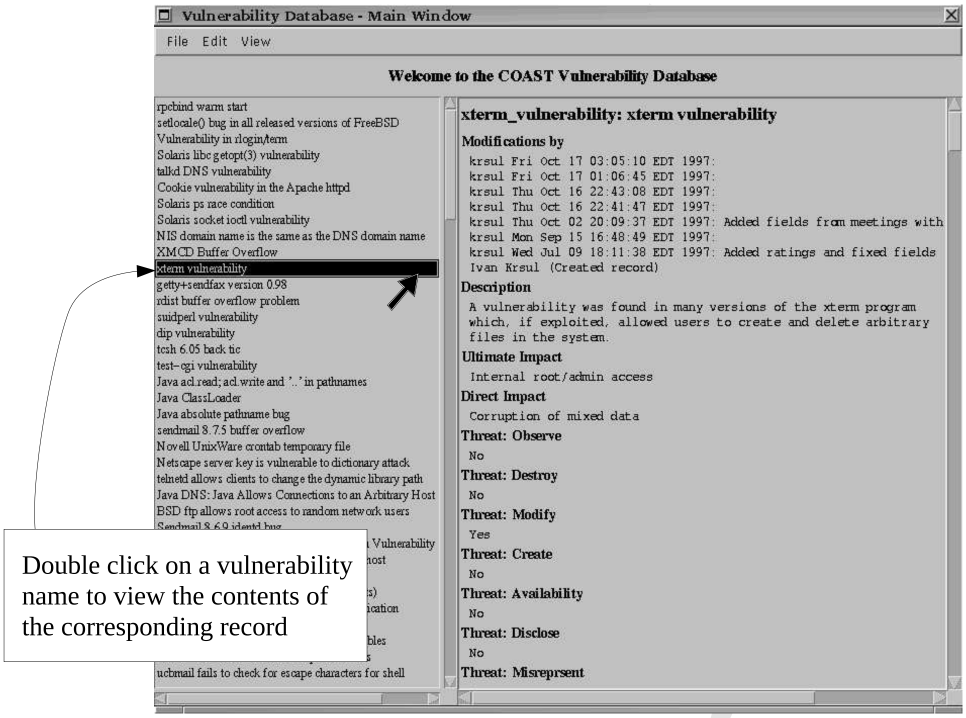

Figure 3: The main screen of the database. Double-clicking on an item on the list of vulnerabilities will display the contents of that record.

Value of 2: Item is not likely to be correct and limited trust should be put on it.

Value of 3: Item is likely to be only partially correct, may contain errors, may be incomplete, etc.

Value of 4: Item seems to be correct but has not been verified by a trusted party. The operator that entered this information, to the best of his knowledge, believes the information to be accurate.

Values of 5: Item is correct and has been verified by a trusted entity. The operator has evidence that the item is correct and can guarantee, with a high probability, that the item contains accurate and complete information.

When entering data you should be specially careful to enter the appropriate rating for the data that you are entering. Leaving that rating at its default value of zero will cause the data that you are entering to be ignored in automatic processes.

All text fields where you can type information can have MIME parts inserted within the text[1] MIME parts are manipulated by using the following keyboard commands while in the text field:

`<control-i>`**:** Insert textual mime part. Opens a dialog, as shown in Figure 7, that allows the user to type or paste text into the field and insert it as a MIME part.
Important Note: *The editor is not smart enough, nor it should be, to notice that you have inserted a MIME part and that it should remove the corresponding file if you decide to discard your changes to the record. Hence, if you add a MIME part and then discard your changes to the record you will have a MIME part file in the* `$MIMEINCLUDES` *directory that will not be referenced by any record. Hence, delete the MIME parts created manually before discarding your changes to the record if you want the MIME parts to be discarded too!*

`<control-d>`**:** Delete MIME part. This option deletes the MIME part where the cursor is located. The MIME include directive is removed from the text and the MIME part file is deleted from the file system.

[1] Fields that have associated classifiers can also have MIME parts. However, we don't recommend that this be done as there are utilities will not work correctly in this case.

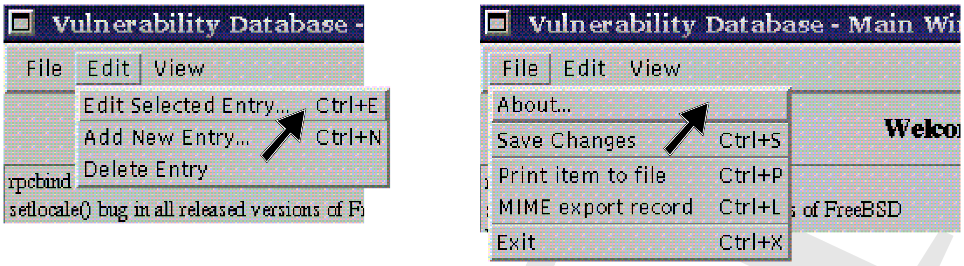

Figure 4: The file menu has options for saving your changes to the database, exiting the database, printing the record as pure text, and exporting the record as a multi-part MIME file. The edit menu has options for editing the current record and adding new records to the database

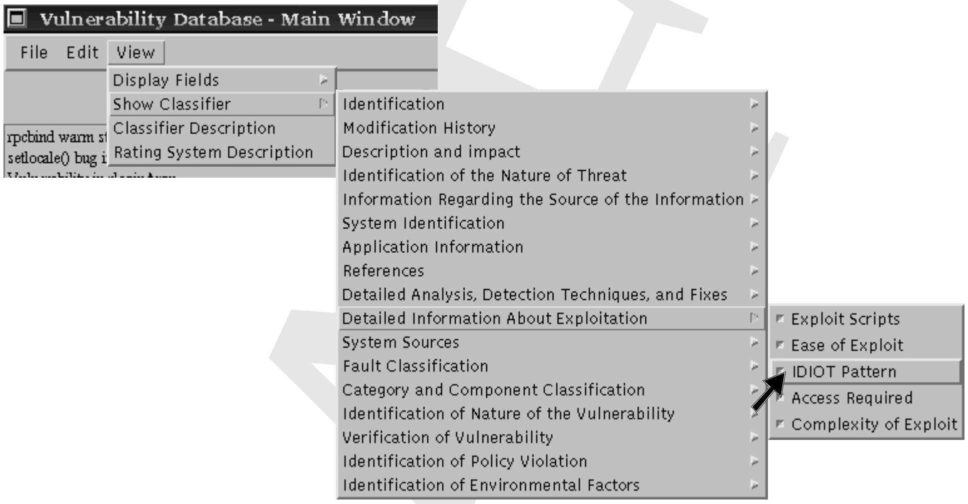

Figure 5: The view menu has options for displaying information regarding the field rating system, the classifiers used for the database, and for limiting the fields that are displayed or printed for a record.

`<control-e>`**:** Edit MIME part. If the MIME part is editable then this command allows the user to edit the part in a special MIME part editor as shown in Figure 7.

`<control-v>`**:** View the MIME part. Displays the content of the MIME part in a special window.

`<control-x>`**:** Export MIME part. *Not implemented yet!* Allows the user to export this part to a multi-part MIME file that can be viewed with an external viewer or that can be send via email.

`<control-m>`**:** View part with an external viewer. If the MIME part is not a textual part then it cannot be viewed using the `control-v` command. This command saves exports the part as a temporary file and calls an external MIME viewer to display the part.

`<control-f>`**:** MIME encode a file. This command opens a file dialog box and lets the user select an external file that must be MIME encoded and saved to a MIME part for the record. Once the file is selected, the interface will attempt to guess the MIME type and will open a dialog box, shown in Figure 8, to confirm that the type selected is indeed correct. If it is not, then select the correct type and proceed with the conversion.

As shown in Figure 9 MIME parts are highlighted in the main window and can be viewed by double clicking on the name of the included part. *Bug Note:* Under some window managers in UNIX, a double click is defined as two

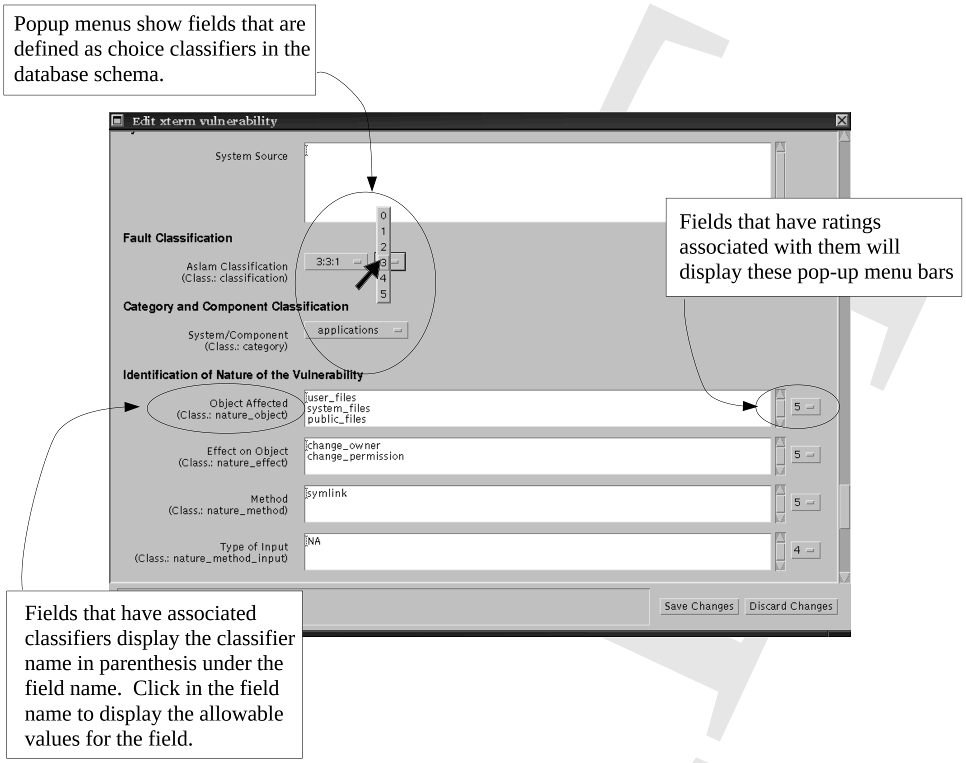

Figure 6: The Edit Screen in the Java Interface

successive clicks on the mouse with the *middle* button. With other window managers the double click is done with the left button.

4 Classifiers

Classifiers are stored as files in the directory `$CLASSD`. Depending on whether the classifier is a straight list or a choice list,the first line of the classifier includes the type. Classifiers can have comments and all blank lines in classifiers are ignored.

For example, a standard choice list for yes/no fields is the `yes_no` classifier. It typically has the following contents:

```
#cclass

# Choice list for a yes/no choice.
# This choice list also allows the
# user to specify that a yes/no answer
# is not appropriate or that a value is
# not know for the record.

yes#Yes
no#No
NA#Does not apply
?#Unknown
```

An example of a list classifier is the `system` classifier that defines the kinds of operating systems that are allowed. A reduced example of this classifier is:

```
#list
# This classifier is used to define
# operating systems and has the
```

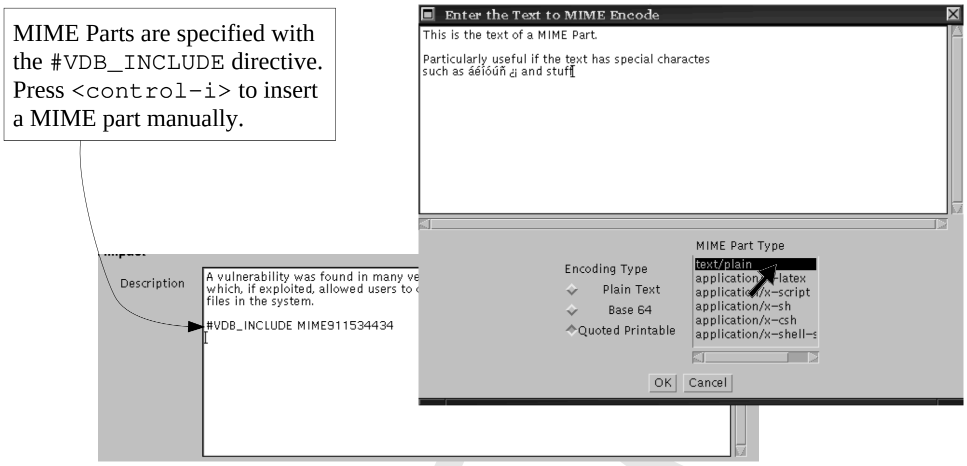

Figure 7: Inserting MIME parts into a field in a record of the database

```
# following form:
#
#  Operating system code#Operating system name
#
Solaris#SUN Solaris
SunOS#SUN OS
DOS#Microsoft DOS
Windows95#Microsoft Windows 95
Caldera#Caldera
Goah#NEC's Goah
NA#Does not apply
```

5 Detailed Description of the Schema File

The vulnerability database schema file (`$VDBSCH`) describes the fields allowable in the database. It is a text file where comments are indicated by starting a line with the character `#`. Every non-comment line defines a field or a separating header.

Every field specification is composed of the following parts separated by the `#` character: Field ID, short name, long name, field type, field height. Field IDs should be unique. The short name is used to display identify the field in editors and reports. The long name is used to provide descriptive information about the field to the user on data entry and reports. The field type and number of lines tells the Java interface how to represent this field in the data entry editor.

The field types are one of `text`, `secsep`, `cclass`, and `list`[2]:

`text`: Fields of type `text` are general text fields that can contain any textual information. These fields can have included MIME parts that are specified by putting in a line an include directive of the form `#VBD_INCLUDE file_name` where the file name corresponds to a MIME part file in the appropriate directory under the `$ MIMEINCLUDES` directory.

`secsep`: Fields of this type are not real data fields but used to specify that the line in the schema is a section separator.

`cclass`: These are choice lists. The user is presented with a list of choices and can select a single item from the list.

[2]The database supports fields of type `hclass` and `matrix` but these are not used anymore. Look at the code to find out how to use these!

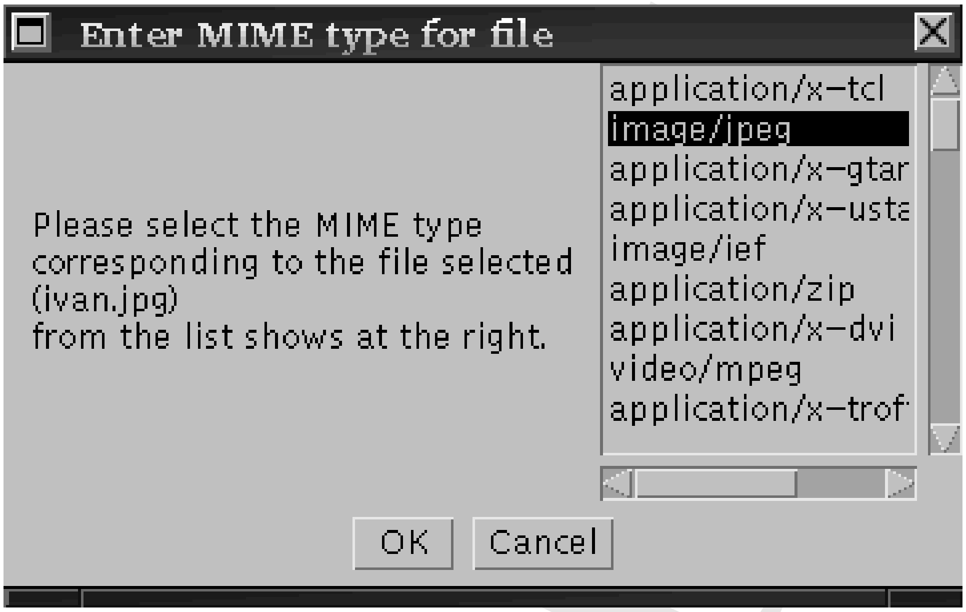

Figure 8: When MIME encoding a file, the interface will attempt to guess the MIME type and will ask the user to confirm that the type selected is indeed correct

`list`: These fields are lists where the user can select multiple elements from the list.

Fields can have options that are indicated by following the field type by a question mark and the options desired. As of this release, the options defined are:

`c`: Every field can have a *confidence interval*. These are ratings that are given to the field that indicate the level of confidence that the data entry operator has on the correctness of the value.

The following text is an example of the schema file for the database. If represents the database as it was defined on October 19, 1997.

IMPORTANT NOTE: The schema file does not support continuation characters. In this example, however, we have added two backslashes (\\) to split lines that were to long to display in this document.

```
# One line per field. Each line has the vulnerability field ID,
# a title, a long [er] description of what the description
# is, a field type, and the recommended editor height
#
# Field types can be one of:
#       text                          (A text field.)
#       class:classifier_file_name    (Choice list classifier)
#       list:list_file_name           (List of allowed values)
#
# Please be ware that the field title is hard-coded
# in all the programs and it should never change or be removed.
#
# March/97 - Ivan Krsul - krsul@cs.purdue.edu
#

# Identification
*#Identification#*#secsep#1
title#Title#Title of the vulnerability#text#2

# Information about modification
*#Modification History#*#secsep#1
modifications#Modifications by#Person(s) that have modified this
record, \\
  the date of modification, and the modifications made#text#4
```

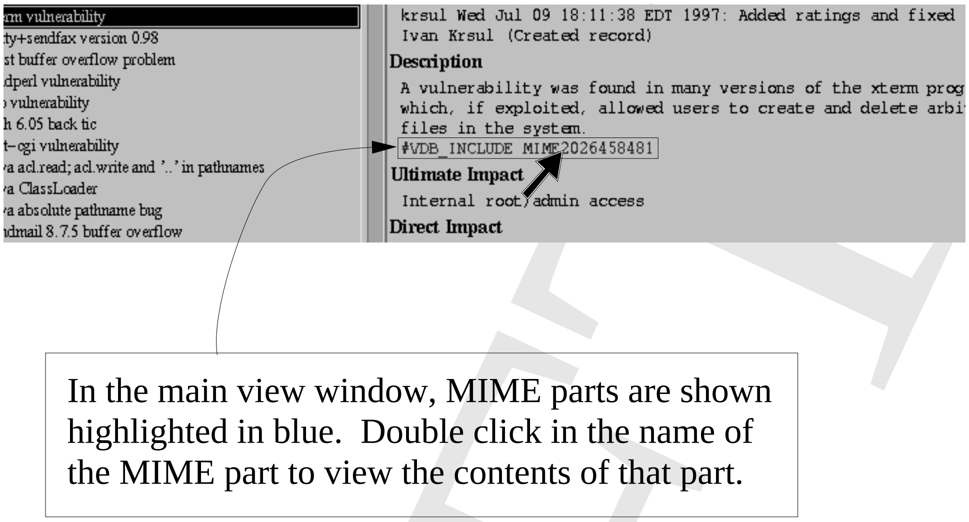

Figure 9: MIME parts are highlighted in the main window and can be viewed by double clicking on the name of the included part.

```
#Description and impact
*#Description and impact#*#secsep#1
desc#Description#Description of the vulnerability#text#15
indirect_impact#Ultimate Impact#Ultimate consequences of an attack \\
  exploiting the vulnerabilty by a threat
agent#cclass:indirect_impact#1
direct_impact#Direct Impact#Rather the the ultimate impact of the \\
  vulnerability, the direct or immediate impact#cclass:direc_impact#1
impact_verbatim#Impact Text#Textual description of the impact of
exploiting \\
  the vulnerability#text#7

#Threat
*#Identification of the Nature of Threat#*#secsep#1
thac_observe#Threat: Observe#The vulnerability can result in a user
observing \\
  objects, data, etc., in violation of expected policy#cclass:yes_no#1
thac_destroy#Threat: Destroy#The vulnerability can result in a user
destroying \\
  objects, data, etc., in violation of expected policy#cclass:yes_no#1
thac_modify#Threat: Modify#The vulnerability can result in a user
modifying \\
  objects, data, etc., in violation of expected policy#cclass:yes_no#1
thac_create#Threat: Create#The vulnerability can result in a user
creating \\
  objects in violation of expected policy#cclass:yes_no#1
thac_cavail#Threat: Availability#The vulnerability can result in the
change of \\
  availability of  the system#cclass:yes_no#1
thac_disclose#Threat: Disclose#The vulnerability can result in the
disclosure of \\
  information in violation of expected policy#cclass:yes_no#1
thac_misrep#Threat: Misreprsent#The vulnerability can result in \\
  misrepresentation of information#cclass:yes_no#1
thac_repudiate#Threat: Repudiate#The vulnerability can result in
repudiation of \\
  information#cclass:yes_no#1
thac_integrity#Threat: integrity#The vulnerability can result in
change of \\
  integrity of the system#cclass:yes_no#1
thac_conf#Threat: Confidentiality#The vulnerability can result in the
loss of \\
  confidentiality of information#cclass:yes_no#1

# Information about the source of the information
*#Information Regarding the Source of the Information#*#secsep#1
source_addres#Source Detail#Detailed information on the source of the
```

```
\\
  information. The WWW address, email address, books, etc. where the
\\
  information was gathered from.#text#4

# System identification
*#System Identification#*#secsep#1
system#System(s)#System(s) vulnerable#list:system?c#4
system_version#System Version#System Version#text?c#4
system_vendor#System Vendor#System Vendor#list:vendor?c#4
system_verbatim#Misc System#Additional textual description of
system#text?c#4
os_type#Type of OS#Type of operating systems affected#cclass:os_type#1

# Application information
*#Application Information#*#secsep#1
app#Application#Application that contains the
vulnerability#list:application?c#4
app_version#Application Version#Application Version#text?c#4
app_verbatim#Verbose application#Long description of applications
that contain \\
  vulnerabilities#text?c#5

# References
*#References#*#secsep#1
advisory#Advisory/ies#Advisory/ies that warn/describe about the \\
  vulnerability.#text#4
reference#References#References to the vulnerability in literature or
in the \\
  net#text#4
related_docs#Related Docs.#Documents that describe the vulnerability,
related \\
  to the vulnerability or that are useful in the analysis of the \\
  vulnerability#text#10

# Detailed analysis, detection techniques and fixes
*#Detailed Analysis, Detection Techniques, and Fixes#*#secsep#1
analysis#Analysis#A detailed analysis of the vulnerability#text?c#15
core_vulner#Core Vulnerability#If the vulnerability is in a piece of
code, the \\
  smallest piece of code that still has the vulnerability#text?c#15
detection#Detection#Method of detecting that the vulnerability is
being \\
  exploited#text?c#10
fix#Fix#A fix that can be used to eliminate the
vulnerability.#text?c#10
test#Test#Method that can be used to detect whether the vulnerability
is present \\
  in a system#text?c#10
workaround#Workaround#A temporary workaround for the vulnerability.
Used until \\
  a patch can be applied.#text?c#10
patch#Patch(es)#A patch or a series of patches that can be used to
eliminate the \\
  vulnerability.#text?c#15

# Detailed information about exploitation
*#Detailed Information About Exploitation#*#secsep#1
exploit#Exploit Scripts#Reference to exploit scripts or
programs#text?c#15
ease_of_exploit#Ease of Exploit#How easy is it to exploit the \\
  vulnerability#list:ease_of_exploit#3
idiot#IDIOT Pattern#IDIOT Pattern used to detect the exploitation of
the \\
  vulnerability.#text#15
access_required#Access Required#What access is required for the \\
  exploitation#cclass:access_required#1
complexity_of_exploit#Complexity of Exploit#How complex is the
exploitation of the \\
  vulnerability#cclass:complexity_of_exploit#1

# Source code and pointers to source code for the systems that
contain the \\
  vulnerabilities.
*#System Sources#*#secsep#1
system_source#System Source#Source code or a pointer to the source
code for the \\
  system that contains the vulnerability#text#7

# Classifications and features
```

```
*#Fault Classification#*#secsep#1
class#Aslam Classification#Aslam Classification. See the
documentation for the \\
  possible values and an explanation.#cclass:classification?c#3

*#Category and Component Classification#*#secsep#1
category#System/Component#To what system or component does the
vulnerability belong \\
  to#cclass:category#1

# Nature of vulnerability
*#Identification of Nature of the Vulnerability#*#secsep#1
nature_object#Object Affected#The object fundamentally affected by
the \\
  vulnerability#list:nature_object?c#3
nature_effect#Effect on Object#The effect that the vulnerability has
on the \\
  object#list:nature_effect?c#3
nature_method#Method#The method or means by which the object is \\
  affected#list:nature_method?c#3
nature_method_input#Type of Input#The type of input, if any, that
leads to the \\
  effect#list:nature_method_input?c#3

# Verification of vulnerabilty.
# Although the verif field accepts any value, it is unlikely that it
# will ever be used as an enumeration.  Rather it is likely to be used
# as a boolean that indicates if this vulnerability was verified.
# However, it may be useful for humans reading the database to add the
# full name of the person or persons that verified the vulnerability.
*#Verification of Vulnerability#*#secsep#1
verif#Verified by#Person or entity that verified the vulnerability.
Verification \\
  should imply that the vulnerability is know to exist and had been
exploited or \\
  verified by the person named.#text#3

#
# Policy features
*#Identification of Policy Violation#*#secsep#1
policyvio#Expected Policy Violated#Expected Policy Violated by the
Vulnerability.  \\
  These policies need not be formally specified and are the
expectation that users \\
  feel have been violated.#text?c#5

#
# The following fields are used to indicate future features that
would be desirable
# for this record of the database
*#Identification of Environmental Factors#*#secsep#1
environment#Environment Features#What environmental conditions
contribute to the \\
  vulnerability?  What assumptions are made about the environment
that don't hold?  \\
  What about the environment makes this vulnerability
possible?#text#10
features#Other Features#What other characteristics and features are
relevant for the \\
  understanding of the vulnerability?#text#10
```

6 MIME Support in the Database

To support the inclusion of all kinds of data in the database, all of the text fields that do not contain classifiers can contain pointers to MIME encoded fields.

7 Considerations on the Inner Workings of the Java Interface

- The source code for the database is has sufficient comments to be easily understood. The `docs` directory in the `$JAVAGUIDEV` contains the Javadocs documentation for all the classes of the database GUI. The main Java file is `VulnerabilityDatabase.java`.

- In addition to all the source code developed at COAST, the database uses the PerlTools classes for implementing pattern matching[3]. The MIME routines were developed on top of the Cryptix-Java V2.2 sources[4], however, we only used the MIME packages of that distribution and hence there is no crypto code in our program.
- All the site-dependencies have been centralized in a single class called `vdbGlobals.java`.
- The Java GUI uses two perl programs to MIME encode large files and export records to a multipart MIME file. These are the `encodeMIMEFile.pl` and `generateMIMEMultiPart.pl`. These perl programs are not system specific and hence should be completely portable. They receive from the Java GUI all the information they need to perform their task.

8 Related Analysis Tools

There are two directories that contain utilities that are useful for generating data sets from the database such that these can be used with data analysis tools.

In the `$COWORD` directory are all the tools that perform co-word analysis on the database. The document `$COWORD/README` contains a description of the tools and gives examples of how to use them.

In the `$GENDSET` directory are all the tools necessary for generating, from the database, datasets that can be used in the LNKNet, Mineset, and Xgobi program. The document `$GENDSET/$README` contains a description of the tools.

[3] See `http://www.oroinc.com/`

[4] See `http://www.systemics.com/docs/cryptix/`