Monitoring Chat Users

Nicolas Bonvin
nicolas.bonvin@epfl.ch

Computer Science

Semester Project

July 2004

Responsible
Prof. Serge Vaudenay
serge.vaudenay@epfl.ch
EPFL / LASEC

Supervisor
Philippe Oechslin
philippe.oechslin@epfl.ch
EPFL / LASEC

Table des matières

Première partie

Rapport de projet

1 Introduction

Le but de ce projet est de corriger et d'améliorer le programme *Chat Monitor* de Vincent Magnin[1], utilisé essentiellement par la police cantonale vaudoise pour surveiller les canaux de communications *IRC*. Trois étapes se partagent le travail à accomplir. La première étant le déverminage du programme qui présentait quelques faiblesses au niveau de la stabilité. La seconde partie concerne son adaptation pour l'utilisation particulière faite par les inspecteurs. L'exploration de nouvelles fonctionnalités ou de nouveaux moyens de surveillance fait l'objet de la dernière partie.

Le protocole *IRC* permet à différentes personnes de « chater » en utilisant Internet. Ce protocole étant peu connu du grand public, il est en grande partie utilisé par des utilisateurs avertis et notamment par le milieu « underground[2] ». *IRC* utilise un système proche de *FTP* pour permettre aux utilisateurs de s'échanger des fichiers : les *FSERVs*. C'est précisément par ce moyen qu'est échangé le matériel illégal.

Ce programme est destiné à surveiller uniquement les ressortissants helvétiques. Internet étant ce qu'il est, identifier de manière sûre les utilisateurs suisses est mission impossible. Il suffit à un utilisateur d'utiliser un proxy pour se jouer de la surveillance de ce programme. En outre, la seconde version de ce logiciel se destine plutôt à la police fédérale, du simple fait qu'identifier le canton d'un utilisateur suisse n'est possible que par les fournisseurs d'accès Internet. Une fois le canton du suspect identifié, la police cantonale prendra le relais.

Ce document relate dans un premier temps les étapes du portage, puis explique la manière dont a été entrepris le déverminage. Suivent des explications sur les ajouts apportés au logiciel de base, ainsi que sur les fonctionnalités retirées.

En seconde partie de document se trouve un guide d'utilisateur, permettant une prise en main rapide du logiciel.

[1] Pour plus d'informations techniques concernant les bases du programme, il est conseillé de lire le rapport réalisé par Vincent Magnin.

[2] Warez, Cracks, Gamez, Mp3, Sex, ...

1.1 Fonctionnement général

Chat Monitor commence par se connecter à un serveur *IRC*. L'utilisateur configure le programme et lui indique les canaux à surveiller. Une fois ces canaux rejoints, *Chat Monitor* enregistre et analyse tous les messages émis par les utilisateurs. Au moment où un utilisateur émet un message, le programme va tenter d'identifier sa nationalité. Pour cela, il essaie de récupérer l'*IP* de l'utilisateur. Avec cette dernière, il est possible de retrouver le pays du fournisseur d'accès à internet ayant délivré cette *IP*. S'il s'avère que cette personne est suisse, alors le programme va enregistrer le moindre de ses messages. En particulier, il va chercher à connaître les *FSERVs* auxquels l'utilisateur s'est connecté, et regarder si ce dernier en possède un.

Les utilisateurs de matériel illicite cherchent à s'échanger leur données. Pour ce faire, il leur est nécessaire d'indiquer aux autres utilisateurs du canal leur *Fserv*. Ces annonces[3] prennent la forme d'un simple message, par exemple :

```
toto > Trigger : !toto Ratio : 1 :2 Start Credit : 0 KB Desc :
HiSpeed Dedicated Server
```

En envoyant ce simple message :

```
!toto
```

sur un canal *IRC*, on se connecte au *FSERV* de l'utilisateur toto. Une fois connecté, on peut naviguer dans le système de fichiers de manière classique en utilisant les commandes `DIR`, `LS`, `CD`, ...

Le programme cherche à identifier toutes les annonces de *FSERV* et à s'y connecter, afin d'en lister le contenu.

A la fin de la session de surveillance, chaque utilisateur suisse possèdera un fichier contenant ses moindres actions. De plus, les logs bruts des canaux, des messages de chaque utilisateur, des échanges de fichiers sont également disponibles. Afin de faciliter le travail de la police, un rapport HTML peut être automatiquement généré. Pour rendre ce programme plus efficace, il est possible de lister des mots-clés qui génèreront une alerte, comme par exemple un nom de fichier connu pour être pédophile.

[3]La liste des mots-clés d'annonce de messages se trouve dans le fichier `words.txt`

2 | Portage

2.1 Langage

La première partie de ce projet consistait à porter l'ancienne version du logiciel vers *JAVA 1.4*. En effet, il était écrit en *Visual J++*, langage qui n'est plus supporté par son éditeur. Les principales difficultées sont venues du fait que ces deux langages n'utilisent pas toujours les mêmes types de base. Par exemple le type *Text* n'existe pas en *JAVA*. Il doit remplacé par *String* ou *StringBuffer* selon les cas.

Toutes les simulations de vecteurs par des tableaux ont été remplacées par de vrais vecteurs. En outre le stockage d'informations sur disque était réalisé, dans la première version, de manière générale à l'aide de fichiers sérialisés. Cette manière de procéder introduit plusieurs problèmes :

- il est impossible de pouvoir lire les données directement en ouvrant le fichier.
- le seul moyen pour récupérer ces informations est d'utiliser le programme qui a généré le fichier. Ceci peut devenir problématique lorsque le programme deviendra obsolète.

Dans un souci de transparence et de pérennité des données, la majeure partie des fichiers sérialisés ont été remplacé au profit de fichiers texte ou de fichiers *XML*.

Traduire du *J++* en *JAVA* ne fut guère compliqué, tant la syntaxe de ces deux langages est proche. Cependant l'écrasante majorité des fonctions a fait l'objet d'une réflexion pour trouver une implémentation plus efficace ou plus élégante.

2.2 Interface graphique

L'ancienne version utilisait des classes obsolètes depuis 1997 disponible dans le domaine public pour l'interface utilisateur. Il a donc fallu réécrire entièrement la partie graphique en utilisant les systèmes qu'offre *JAVA* : *SWING* et *AWT*. L'interface graphique étant intimement liée à la logique du

programme, il a fallu avoir une compréhension globale du programme (qui faisait tout de même un peu moins de 30 000 lignes de code !). En parallèle du portage de la logique du programme et de la réécriture de l'interface graphique, de nombreux efforts furent investis pour nettoyer et consolider certaines fonctions, ou totalement les réécrire.

Le design général de l'interface graphique n'a que très peu changé, afin de ne pas dérouter les utilisateurs.

2.3 Remarques

Cette partie du projet prit beaucoup plus de temps que prévu, et fut en fin de compte très frustrante. Se plonger dans un moyennement gros programme écrit par un autre n'est pas chose aisée. Il est également difficile de ne pas tomber dans le piège classique pour un informaticien : réinventer sa propre roue.
Un sentiment de frustration est apparu lorsqu'après maints efforts, le programme se vit porter de « JAVA » en ... *JAVA* !
Certes, de nombreuses améliorations ont été réalisées durant le portage, il n'en reste pas moins qu'aucune nouvelle fonctionnalité révolutionnaire n'a été ajoutée.
Effet quelque peu inattendu, cette deuxième version se montre plus gourmande en mémoire vive. Des efforts conséquents ont été entrepris pour limiter au maximum la consommation mémoire, cependant de temps à autre, l'envirronement *JAVA* est en manque crucial de resources. Il donc conseillé de suivre les indications du chapitre 8.2 lors du lancement du programme.

3 Déverminage

Comme indiqué dans les sections précedentes, de nombreuses améliorations ont été entreprises durant le portage même du logiciel. Cependant, deux problèmes rapportés par les utilisateurs de la première version ont fait l'objet d'une attention particulière.

3.1 Déconnection réseau

Plusieurs raisons font qu'il est possible que *Chat Monitor* soit déconnecté du serveur, comme une coupure de la connection Internet, l'arrêt d'un serveur *IRC*, le fait qu'un opérateur nous expulse du serveur en se rendant compte, par exemple, que c'est un robot[1], ...

Il est donc nécessaire de pouvoir s'y reconnecter et reprendre la surveillance. Deux nouvelles petites fonctionnalités ont été ajoutées pour rendre ce processus plus efficace. A chaque déconnection, *Chat Monitor* va attendre de plus en plus de temps avant d'essayer de se reconnecter. Ceci permet de contrer dans une certaine mesure le fait que certains serveurs mettent l'*IP* du déconnecté sur une liste noire temporaire.

D'une même manière, l'identité peut être mise sur une liste noire. Pour éviter cela, *Chat Monitor* rajoute un aléa aux données de connection.

3.2 Parcours d'un FSERV

Une des fonctionnalités principales de *Chat Monitor* présentait une imperfection : dans certains cas, le robot chargé de parcourir le *FSERV* se plantait.

Le fonctionnement du *thread*[2] responsable du parcours d'un *FSERV* est simple : il cherche à recréer l'arboresence des répertoires sous forme d'arbre. Certains *FSERVs* nécessitent l'envoi d'une commande spéciale, genre `C` ou encore `RULES` pour en accorder l'accès. *Chat Monitor* est capable de détecter

[1] Ce cas s'est présenté sur le canal où la majorité des tests ont été réalisés !

[2] Appelé aussi robot, ou Crawler dans le code source.

ces demandes, et d'agir en conséquence. Ensuite, le robot envoie la commande `DIR` afin de lister les répertoires et fichiers disponibles à la racine. Ces fichiers et répertoires sont les premiers fils de l'arbre. Pour chaque répertoire, il va, de la même manière, créer un arbre en lui ajoutant le parent correspondant. Lors du parcours du *FSERV* et de la création de l'arbre, *Chat Monitor* en profite pour garder une trace de tous les fichiers rencontrés. Une fois tous les fils parcourus, et lorsque l'arbre actuel n'a pas de parent[3], le robot cesse son activité. *Chat Monitor* va ensuite lancer une demande de connection au prochain *FSERV* de sa liste et un nouveau robot sera créé.

Le problème de la première version du robot est qu'il se mettait à boucler de manière infinie dans les répertoires proposés. Ce problème ne se présente que très rarement, et seulement avec certains types de *FSERVs*. Il est dû au fait que certaines réponses du *FSERV* ne sont pas comprises par *Chat Monitor*. Ne comprenant pas ces messages, il recommençait son parcours au niveau du répertoire supérieur[4], et retombait forcément dans ce même répertoire.

La norme étant assez vague, ces messages ne sont pas tous documentés. Plusieurs de ces messages ont été identifés, et traités. Ils concernent pour l'essentiel certaines propriétés des répertoires, comme le fait qu'on n'y ait pas accès. Ces répertoires sont omis par *Chat Monitor* lors du parcours du *FSERV*.

[3]On se trouve à la racine. Tous les répertoires ont été parcourus.

[4]Le noeud parent de l'arbre courant.

4 Fonctionnalités ajoutées

4.1 Système de sauvegarde

Un logiciel, même censé fonctionner de manière autonome, n'est pas exempt de bugs et peut se planter à tout moment. En particulier, un logiciel utilisant un réseau peut perdre la connection ou encore se faire déconnecter du serveur avec qui il communique. Les résultats obtenus durant le fonctionnement normal du programme ne doivent pas être perdus. C'est pourquoi un système efficace de sauvegarde doit être implémenté. La première version du logiciel utilisait des fichiers sérialisés pour les sauvegardes des données sur les utilisateurs, et des logs bruts pour les canaux. Tous les fichiers sérialisés, impossibles à relire sans le programme, ont été avantageusement remplacés par des logs au format *XML*. Ce format permet de traiter facilement de manière informatique les données recueillies.

4.1.1 Logs

Chat Monitor utilise abondamment les logs pour sauvegarder les différents résultats obtenus durant la surveillance des canaux. Le programme utilise deux sortes de logs différents selon la nature des traitements à effectuer :

Logs bruts

Ces fichiers servent à garder une trace de toutes actions effectuées ou de tous les événements rencontrés par *Chat Monitor*. En principe, aucun traitement n'est effectué sur ces fichiers. Plusieurs répertoires contiennent ce genre de logs :

dcc ces logs contiennent tous les échanges directs avec d'autres utilisateurs. On y trouve donc le contenu des *FSERVs* visités par *Chat Monitor*, ainsi que toutes les commandes utilisées pour naviguer dans le *FSERV*.

msg ces logs contiennent les messages envoyés par chaque utilisateur surveillé. Pour avoir un compte rendu des autres utilisateurs, il est nécessaire de consulter les logs en rapport avec les canaux.

room ce répertoire contient les logs de chaque canal visité par *Chat Monitor*. Tous les messages échangés d'un canal y sont stockés.

En outre deux autres fichiers contiennent des logs bruts. Ces fichiers contiennent des informations générales, et non pas des informations relatives à un utilisateur en particulier :

MonitorLog.txt ce fichier texte contient les informations relatives à la surveillance des utilisateurs effectuée par *Chat Monitor*. Ce fichier contient les annonces de *FSERVs* détectées par *Chat Monitor*. Y sont indiqués le canal, le possesseur du *FSERV*et la commande nécessaire pour s'y connecter. On trouve également des entrées relatives à la détection d'un utilisateur suisse. Des entrées y sont également ajoutées lorsque le programme détecte qu'un utilisateur utilise le *FSERV* d'un autre utilisateur.

ActionsLog.txt ce fichier garde un trace des actions effectuées par *Chat Monitor* dans le cadre de son fonctionnement normal. Par exemple on y trouve les entrées relatives aux listes de salons, à la récupération de l'IP et du pays d'un utilisateur ayant émis un message, les tentatives de connection d'un utilisateur à un *FSERV*, ...

Logs XML

Les fichiers de sauvegarde nécessitant un traitement afin d'y extraire les informations importantes ou encore lorqu'il est nécessaire d'effectuer des regroupements sont stockés sous format *XML*. Ce format de fichier a également l'avantage d'être très lisible aussi bien par une machine que par un humain notamment à l'aide d'un navigateur web.
Le traitement des fichiers *XML* a fortement été facilité par l'utilisation de *DTD2Java*[1]. Il devient possible de recréer en mémoire la structure logique d'un fichier *XML* chargé depuis un disque sans passer par les systèmes fastidieux que sont *DOM* ou *SAX*, ou à l'inverse créer un fichier *XML* à partir d'objets *JAVA*.
Chaque utilisateur qui fait l'objet d'une surveillance possède son propre fichier dans lequel ses moindres actions et messages sont stockés. La structure logique de ce fichier est la suivante :

```
<!ELEMENT chatuser (ownedfservs, visitedfservs, ulogs)>
<!ATTLIST chatuser
        uuid CDATA #IMPLIED
```

[1]Ce programme a été écrit par Xavier Perséguers. Il est disponible à l'adresse suivante : `http ://ic.epfl.ch/~ persegue/`

```
        ip CDATA #IMPLIED
        hostname CDATA #IMPLIED
        age CDATA #IMPLIED
        awayreason CDATA #IMPLIED
        channels CDATA #IMPLIED
        city CDATA #IMPLIED
        country CDATA #IMPLIED
        id CDATA #IMPLIED
        idle CDATA #IMPLIED
        nick CDATA #IMPLIED
        nickhistory CDATA #IMPLIED
        realname CDATA #IMPLIED
        server CDATA #IMPLIED
        serverinfo CDATA #IMPLIED
        sex CDATA #IMPLIED
        status CDATA #IMPLIED
        userhost CDATA #IMPLIED
        validonline CDATA #IMPLIED>

<!ELEMENT  ownedfservs (fserv)*>
<!ELEMENT  visitedfservs (fserv)*>
<!ELEMENT  fserv (owner,room,date,command,announce,files)>
<!ELEMENT  files (file)*>
<!ELEMENT  file (name,size,attribute)>
<!ELEMENT  owner (#PCDATA)>
<!ELEMENT  announce (#PCDATA)>
<!ELEMENT  command (#PCDATA)>
<!ELEMENT  name (#PCDATA)>
<!ELEMENT  size (#PCDATA)>
<!ELEMENT  attribute (#PCDATA)>
<!ELEMENT ulogs (ulog)*>
<!ELEMENT ulog (date,action,target,text)>
<!ELEMENT date (#PCDATA)>
<!ELEMENT action (#PCDATA)>
<!ELEMENT target (#PCDATA)>
<!ELEMENT text (#PCDATA)>
<!ELEMENT room (#PCDATA)>
```

Voici un cours exemple de logs résultant d'une session de surveillance de *Chat Monitor* :

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<chatuser
        channels=""
        ip="68.14.68.99"
        awayreason="N/A"
        sex="N/A"
        hostname="ip68-14-68-99.ri.ri.cox.net"
        age="N/A"
        idle="N/A"
        nickhistory=""
        id="jaybrd54!~jaybrd411@ip68-14-68-99.ri.ri.cox.net"
        validonline=""
        server="N/A"
        status="N/A"
        country="CH"
        city="N/A"
        uuid="1050"
        userhost="~jaybrd411"
        realname=""
        nick="jaybrd54"
        serverinfo="N/A"
```

```
>
  <ownedfservs>
  </ownedfservs>
  <visitedfservs>
    <fserv>
      <owner>1050</owner>
      <room>Mp3passion</room>
      <date>Wed Jun 09 00:20:22 GMT 2004</date>
      <command>!Pobo5</command>
      <announce></announce>
      <files>
        <file>
          <name></name>
          <size></size>
          <attribute></attribute>
        </file>
      </files>
    </fserv>
  </visitedfservs>
  <ulogs>
    <ulog>
      <date>Wed Jun 09 00:20:22 GMT 2004</date>
      <action>SENDROOM</action>
      <target>Mp3passion</target>
      <text>!Pobo5 311 - Transistor - 13 - Color.mp3</text>
    </ulog>
    <ulog>
      <date>Wed Jun 09 00:20:31 GMT 2004</date>
      <action>SENDROOM</action>
      <target>Mp3passion</target>
      <text>!BUCman Enigma - Voyageur-Advance (2003) - 02 - Voyageur.mp3</text>
    </ulog>
  </ulogs>
</chatuser>
```

Une option intéressante de cette nouvelle implémentation du système de logs est qu'il est possible de spécifier le nombre d'entrée par fichier. Ce nombre atteint, un nouveau fichier sera créé. Lors de la génération du rapport *HTML*, *Chat Monitor* ne charge pas tous les fichiers d'un utilisateur en mémoire afin de ne pas consommer trop de mémoire. Le rapport est construit petit à petit de manière séquentielle. Seul un fichier à la fois est chargé en mémoire. Cette opération est relancée pour chaque utilisateur faisant l'objet d'une surveillance. Cette précaution est nécessaire du fait que ce format est particulièrement verbeux et par conséquent les fichiers de logs volumineux.

4.1.2 Sauvegarde automatique

Chat Monitor lance en arrière-plan un *thread* responsable de sauvegarder l'état courant sur le disque toutes les minutes. Les logs bruts contenus dans les répertoires dcc, room et msg sont quant à eux sauvegardés en temps réel. Seuls les logs au format *XML*, ainsi que les fichiers `ActionLog.txt`, `generatedAlarms.txt` et `MonitorLog.txt` sont contrôlés par le *thread*.

Ce dernier possède encore une autre fonction importante : vider les *Widgets* de l'interface *JAVA*. Après plusieurs tests, il s'est avéré que les composants *jTextArea* et *jList* consomment énormément de mémoire. En effet, lors du parcours d'un *FSERV* contenant un nombre important de fichiers, ces composants prenaient tellement de mémoire que cela provoquait systématiquement des erreurs du type `java.lang.OutOfMemoryError`. Le programme était irrécupérable. Le *thread* a donc comme mission de vider régulièrement ces composantes.

4.1.3 Sauvegarde des préférences

La sauvegarde des préférences (*cf.* Fig. 9.1) est effectuée non plus sous forme de fichier sérialisé, mais au format *XML*. Les préférences peuvent donc être éditées directement à la main, sans devoir passer par l'interface de configuration. La structure logique du fichier est la suivante :

```
<!ELEMENT Presets (Preset)*>
<!ELEMENT Preset (
        name,
        sessionname,
        server,
        port,
        login,
        nickname,
        password,
        ident,
        realname,
        age,
        protocol,
        sex,
        city,
        status,
        log,
        reconnect,
        monitor,
        swissdetect,
        follow,
        fservdetect,
        fservuse
)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT sessionname (#PCDATA)>
<!ELEMENT server (#PCDATA)>
<!ELEMENT port (#PCDATA)>
<!ELEMENT login (#PCDATA)>
<!ELEMENT nickname (#PCDATA)>
<!ELEMENT password (#PCDATA)>
<!ELEMENT ident (#PCDATA)>
<!ELEMENT realname (#PCDATA)>
<!ELEMENT age (#PCDATA)>
<!ELEMENT protocol (#PCDATA)>
<!ELEMENT sex (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT status (#PCDATA)>
<!ELEMENT log (#PCDATA)>
```

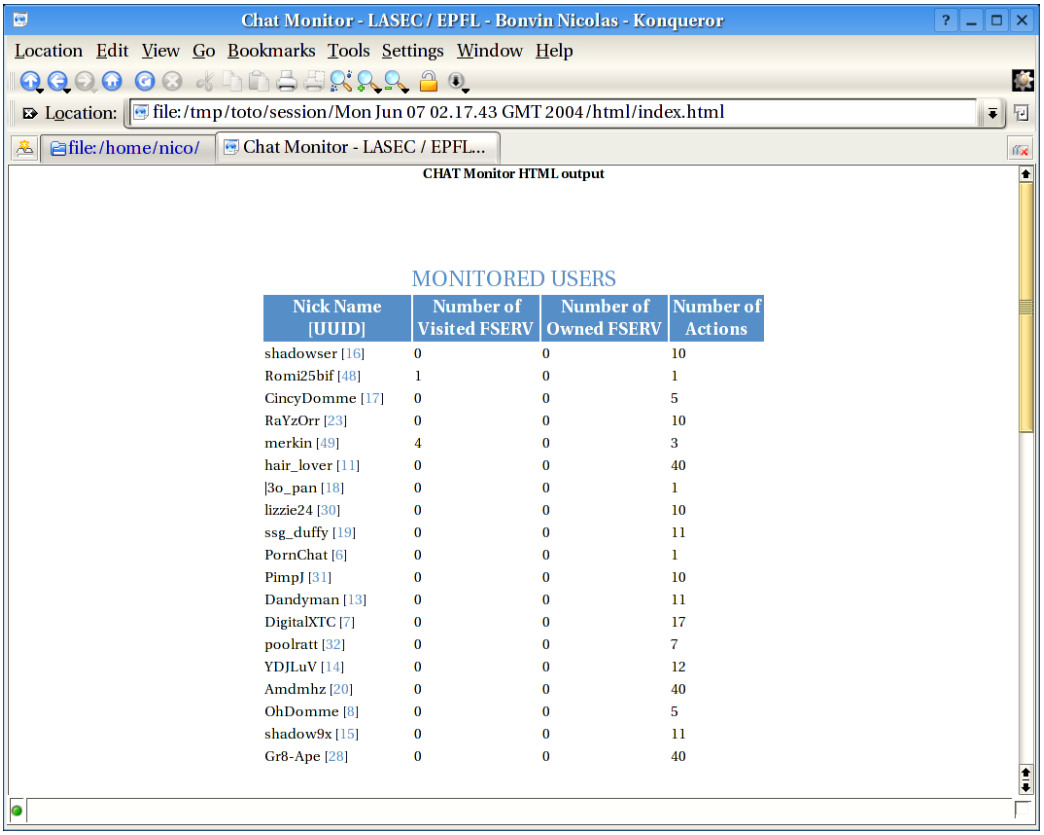

FIG. 4.1 – Fichier `index.html` - Utilisateurs surveillés.

```
<!ELEMENT reconnect (#PCDATA)>
<!ELEMENT monitor (#PCDATA)>
<!ELEMENT swissdetect (#PCDATA)>
<!ELEMENT follow (#PCDATA)>
<!ELEMENT fservdetect (#PCDATA)>
<!ELEMENT fservuse (#PCDATA)>
```

4.2 Rapport HTML

Un des buts de ce projet était de rendre cet outil autonome et simple d'utilisation. D'une manière idéale on aimerait pouvoir le lancer et l'oublier dans son coin. Peu de personnes prennent du plaisir à consulter régulièrement des fichiers de logs bien garnis.

Chat Monitor se propose de rendre cette tâche un peu moins fastidieuse en proposant une vue d'ensemble des fichiers de logs importants. Il suffit

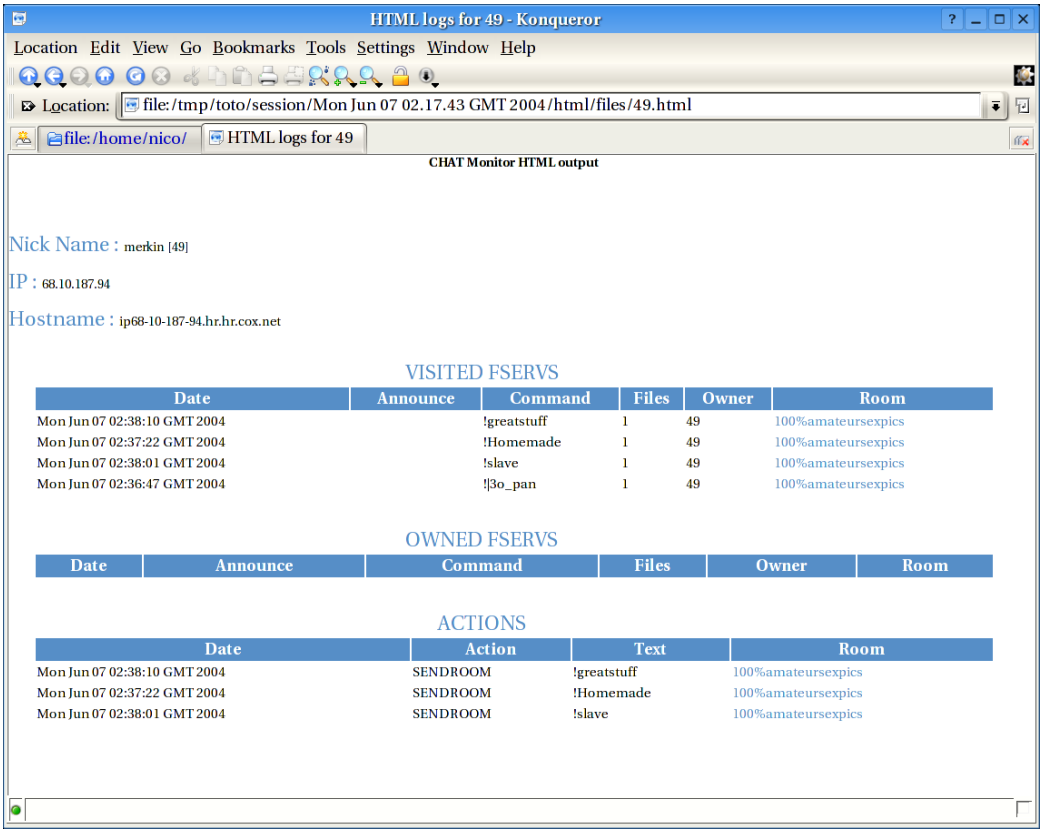

FIG. 4.2 – Détails d'un utilisateur surveillé.

de cliquer sur l'option « Generate HTML output » du menu « Logs » (*cf.* Fig. 8.1) pour qu'apparaisse une boîte de dialogue qui permet de choisir le répertoire de logs que l'on veut traiter. Cette fonctionnalité ne crée pas forcément le rendu *HTML* de la session en cours.

Chat Monitor parcourt tous les fichiers de logs *XML* de chaque utilisateur et en récupère différentes informations comme le nombre d'actions effectuées par l'utilisateur, les *FSERVs* qu'il a visité, ou encore les *FSERVs* qu'il détient. Ces informations permettent de créer un fichier `index.html` situé dans le répertoire `html` de la session choisie. A partir de cette vue d'ensemble (*cf.* Fig. 4.1) de tous les utilisateurs faisant l'objet d'une surveillance, il est possible en suivant les ancres *HTML* appropriées de voir les logs complets d'un utilisateur précis (*cf.* Fig. 4.2), toujours sous forme de fichier *HTML*.

D'autres informations importantes se trouvent dans le fichier `index.html`.

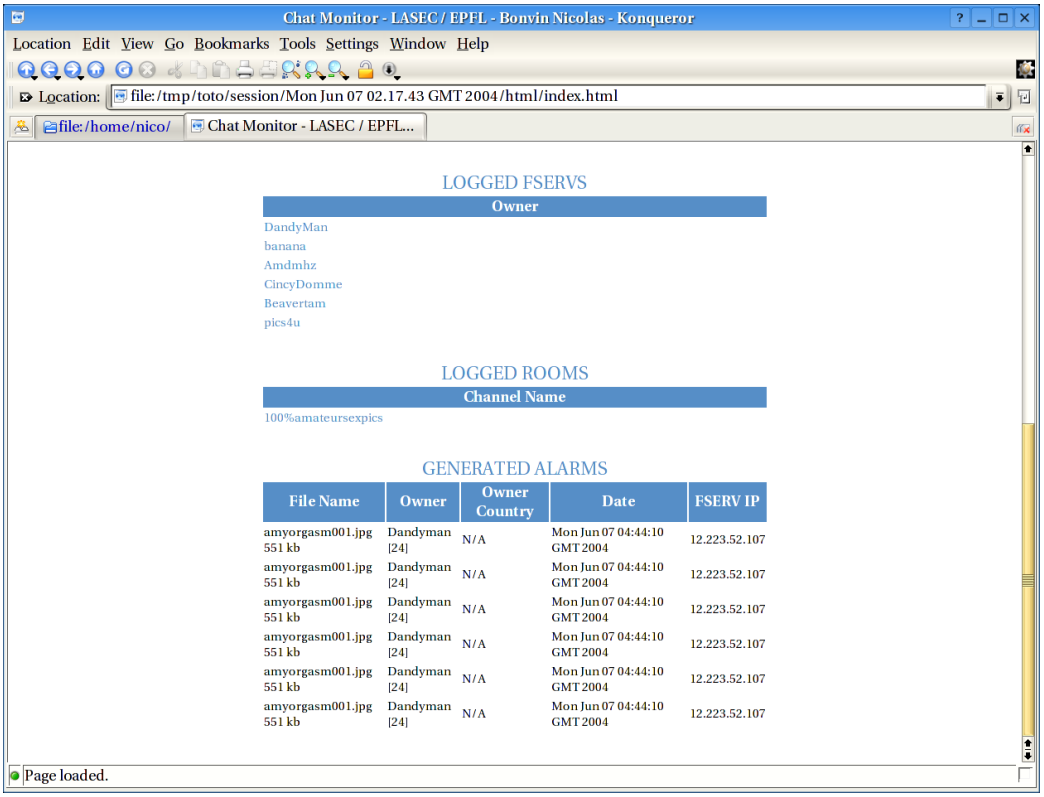

FIG. 4.3 – Fichier `index.html` - liste des *FSERVs* visités, des canaux surveillés et des alertes générées par *Chat Monitor*.

comme les *FSERVs* dont le contenu a été listé par *Chat Monitor* (*cf.* Fig. 4.3), les canaux surveillés ou encore toutes les alarmes levées lors du parcours des différents *FSERVs*.

Lors de la génération des logs *HTML*, *Chat Monitor* remplace toutes les occurences d'utilisateurs connus ou de *FSERVs* visités par un lien direct vers les fichiers de logs bruts. Ceci permet de recouper les informations de manière plus aisée.

5 Fonctionnalités retirées

La première version du programme avait des objectifs légèrement différents de ceux de la version actuelle. Quelques fonctionnalités ont de fait été retirées ou n'ont fait l'objet d'aucun effort durant ce projet.

5.1 Multi-protocoles

Ce logiciel avait comme objectif de pouvoir surveiller plusieurs protocoles de *CHAT* comme *Caramail* ou *IRC* de manière transparente en utilisant à bon escient un système de classes abstraites. Un système de couche similaire à celui bien connu de *TCP-IP* avait été developpé.

Les objectifs de cette version étaient légèrement différents : seul *IRC* était important pour la police. Le système de couche a été conservé, afin de ne pas réduire à néant le travail déjà fait. Cependant, seul le protocole *IRC* a été amélioré.

5.2 Logiciel de chat

Vincent Magnin s'était largement basé sur un logiciel de *CHAT* développé par ses soins pour créer *Chat Monitor*. Ce dernier possède donc les fonctionnalités classiques d'un logiciel de ce genre. Du fait des besoins réels requis, certaines fonctions de *CHAT* n'ont pas été portées, notamment au niveau de l'interface graphique.

6 Conclusion

6.1 Généralités

Un programme qui plante n'est guère utilisable en production, même si ce dernier possède des fonctionnalités époustouflantes. C'est pourquoi de nombreux efforts ont été mis en oeuvre afin de rendre *Chat Monitor* plus stable. De plus, il faut être attentif au fait que l'utilisateur final n'est pas un technicien, d'autant plus que les rapports générés sont destinés à être présentés à un juge. Il est donc nécessaire de pouvoir présenter l'information sous forme lisible et compréhensible. Les rapports au format *HTML* jouent donc un rôle important.

Dans plusieurs domaines en informatique, une personne mal-intentionnée peut utiliser de manière efficace certains moyens comme le chiffrement pour se protéger. Dans le cas présent, ceci n'est pas toujours possible. Certes, les utilisateurs de matériel illégal utilisent des canaux privés protégés par mot de passe, mais à un moment donné, il vont chercher à récupérer plus de matériel, à s'échanger davantage. Il leur est donc nécessaire de se connecter à des serveurs publics afin d'enrôler de nouvelles personnes. C'est à ce moment précis qu'ils sont vulnérables, et qu'un logiciel comme *Chat Monitor* prend tout son sens.

Pour lutter de manière efficace contre ces personnes, il est judicieux d'utiliser *Chat Monitor* à un niveau fédéral, car il est seulement possible d'identifier le pays d'un utilisateur. Déterminer son canton nécessite l'aide des fournisseurs d'accès à Internet. Lorsqu'un suspect est identifé, il suffit alors à la police fédérale de transmettre son dossier à la police cantonale correspondante.

6.2 Notes personelles

Ce projet m'a permis de me familiariser avec *IRC*, *DCC* ou encore le système de *FSERVs*. Cependant, la vraie leçon de ce projet est qu'il est bien plus difficile que prévu de porter un logiciel, même de taille modeste, d'autant plus lorsque l'on n'en est pas l'auteur. Il n'est pas évident d'avoir une vision globale et de s'y retrouver parmi la multitude de fichiers sources.

Deuxième partie

Guide d'utilisation

7 Introduction

Chat Monitor est un programme servant à surveiller les canaux IRC. Il se focalise essentiellement sur les utilisateurs suisses en enregistrant tous leurs messages, les FSERVs visités et détenus. Des alarmes sur des noms de fichiers connus comme étant du matériel illégal facilitent les remontées d'alertes. Il permet également de générer des rapports sous format *HTML* aisés à lire. *Chat Monitor* est essentiellement destiné aux forces de police suisses.
Une première version de *Chat Monitor* avait été réalisée par Vincent Magnin. Cette deuxième version est en grande partie une réécriture de la première, avec en prime quelques améliorations intéressantes comme la levée d'alertes ou les rapports *HTML* facilement compréhensibles. Cette version réponds à des objectifs mieux définis que la première version.

8 Installation

8.1 Configuration requise

Chat Monitor utilise un environnement d'exécution *JAVA*[1]. Il faut donc vous procurer le *Java Runtime Environement (JRE) 1.4* ou supérieur sur le site de Sun MicroSystems[2].

8.2 Lancement du programme

Pour lancer *Chat Monitor* il suffit de taper la commande suivante dans un terminal :

```
java -Xmx128m -jar ChatMonitor.jar
```

L'option `-Xmx128m`[3] spécifie la taille maximale de mémoire que l'environnement *JAVA* peut allouer, ici 128 Mo.

8.3 Création des répertoires

Au lancement d'une session de surveillance, *Chat Monitor* va créer de manière automatique une arborescence de répertoire à l'endroit même où le programme est lancé. Le répertoire racine session contiendra une liste de répertoire correspondant à autant de séances de surveillance. A l'intérieur de chacun de ces répertoires, on trouvera une liste de fichiers et de répertoires :

dcc ce répertoire contient les logs bruts de tous les échanges directs avec d'autres utlisateurs. Dans ces logs, on trouve donc le contenu des

[1] Il est important de télécharger l'environement de *Sun MicroSystems*, et ne pas utiliser celui fourni par défaut par *Microsoft Windows*.

[2] `http ://java.sun.com`

[3] Cette option n'est pas requise. Par défaut, la machine virtuelle *JAVA* alloue 64 Mo.

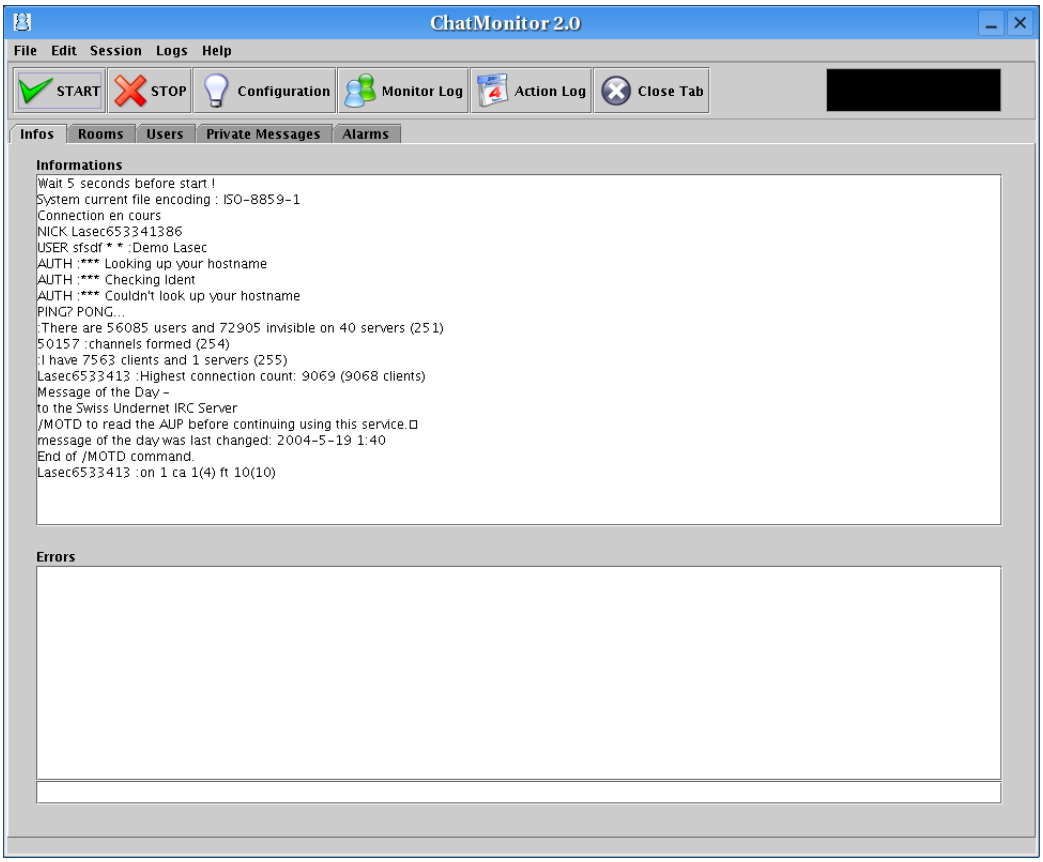

FIG. 8.1 – Fenêtre principale de *Chat Monitor*.

FSERVs visités par *Chat Monitor*, ainsi que toutes les commandes utilisées pour naviguer dans le *FSERV*.

file ce répertoire contient les fichiers que l'utilisateur de *Chat Monitor* a accepté de télécharger. Lors d'une utilisation normale du programme, ce répertoire restera vide, puisque *Chat Monitor* ne télécharge aucun fichier de lui-même.

fserv ce répertoire contient la liste des fichiers des *FSERVs* visités par *Chat Monitor*. Dans certains cas, *Chat Monitor* termine la visite d'un *FSERV* de manière incorrecte[4]. Dans ce cas, aucun fichier ne sera ajouté dans ce répertoire. Bien entendu les logs bruts seront disponibles.

html ce répertoire contient le rapport *HTML* généré automatiquement. Il contient deux sous-répertoires : css et files. Ce dernier contient les fichiers *HTML* de chaque utilisateur surveillé.

msg ce répertoire contient les logs bruts des messages envoyés par chaque utilisateur surveillé.

room ce répertoire contient les logs bruts de chaque canal visité par *Chat Monitor*.

Au niveau de la racine, on trouve également quelques fichiers intéressants :

***.xml** ces fichiers contiennent toutes les informations de chaque utilisateur surveillé.

actions.xml ce fichier contient les actions effectuées par les utilisateurs.

ActionsLog.txt ce fichier contient un résumé des actions effectuées par *Chat Monitor*.

MonitorLog.txt ce fichier contient un résumé de la surveillance effectuée par *Chat Monitor*.

generatedAlarms.txt ce fichier contient un résumé des alertes levées par *Chat Monitor*.

users.db ce fichier serialisé est utilisé par *Chat Monitor* pour identifier les utilisateurs. Il ne contient pas d'informations directement exploitables.

[4]Le détenteur du *FSERV* vient de nous expulser, une coupure du réseau, ...

9 Description de l'interface

9.1 Configuration

Avant de pouvoir vous connecter à un serveur *IRC*, il est nécessaire de configurer *Chat Monitor*. Pour cela cliquez sur l'icone « Configuration » ou choisissez l'option « Configuration » du menu « Edit » (*cf.* Fig. 8.1).

Voici un brève description des options disponibles dans cette fenêtre (*cf.* Fig. 9.1) :

Presets Liste des préselections disponibles.

Protocol Liste des protocoles disponibles. Seul *IRC* est fonctionnel dans la version 2.0.

Server Serveur *IRC* auquel on veut se connecter.

Login Login utilisé pour se connecter au serveur *IRC*.

Pseudo Pseudonyme utilisé dans les canaux *IRC*.

Password Mot de passe pour se connecter à un serveur *IRC* privé et protégé (Optionnel).

Ident Identité pour se connecter à un serveur *IRC*. Ceci est utilisé en commun avec « Real Name ».

Real Name Identité réelle de la personne désirant dialoguer sur *IRC*.

Age Age de la personne. Pas utile pour *IRC*.

Gender Genre de la personne. Pas utile pour *IRC*.

City Ville de la personne. Pas utile pour *IRC*.

Status Statut de la personne. Pas utile pour *IRC*.

Logs Salons/PVs/DDCs Option permettant d'enregistrer les messages et les actions des utilisateurs (logs).

Reconnection Automatique Option permettant de se reconnecter au cas où la connection avec le serveur *IRC* serait coupée.

Surveillance Active Option permettant de surveiller les utilisateurs.

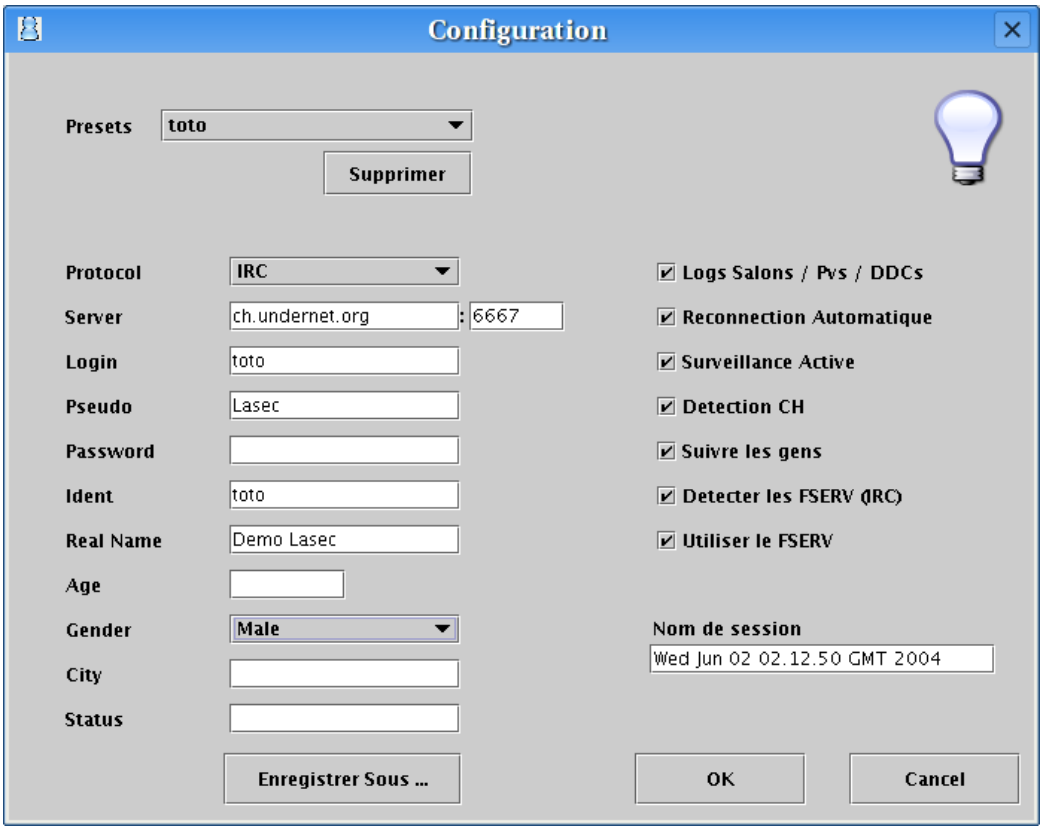

FIG. 9.1 – Fenêtre de configuration.

Détection CH Option permettant de répérer et suivre les utilisateurs helvétiques.

Suivre les gens Option permettant de suivre un utilisateur sur plusieurs canaux.

Détecter les FSERV(IRC) Option permettant d'identifier les annonces de *FSERVs* .

Utiliser le FSERV Option permettant au programme de se connecter automatiquement aux *FSERVs* rencontrés et d'en faire la liste des fichiers.

Nom de session Ce nom est utilisé pour créer le répertoire qui contiendra toutes les informations de surveillance recueillies.

9.2 Interface principale

L'interface principale est composée d'une rangée de boutons permettant d'accèder plus rapidement aux options intéressantes, de menus déroulants, ainsi que de plusieurs onglets.

Onglets

Infos Ce panel comporte deux champs texte. Le premier renseigne l'utilisateur sur le bon déroulement du programme. Le second rends compte des différentes erreurs rencontrées durant l'exécution (*cf.* Fig. 8.1).

Rooms Ce panel sert à rechercher les canaux de discussions, selon plusieurs critères (*cf.* Fig. 9.2) :

Max Users nombre maximal d'utilisateurs dans le salon

Min Users nombre minimal d'utilisateurs dans le salon

Max Minutes canaux plus vieux que le nombre de minutes spécifiés

Min Minutes canaux plus récents que le nombre de minutes spécifiés

Topic-Set Max canaux avec des sujets plus vieux que le nombre de minutes spécifiés

Topic-Set Min canaux avec des sujets plus récents que le nombre de minutes spécifiés

Il est possible d'appliquer un filtre sur les résultats obtenus en cliquant sur le bouton « Filter ». Les boutons « join » et « Enter the Room » permettent de rejoindre un canal.

Users Ce panel permet d'afficher et de rechercher des utilisateurs (*cf.* Fig. 9.2).

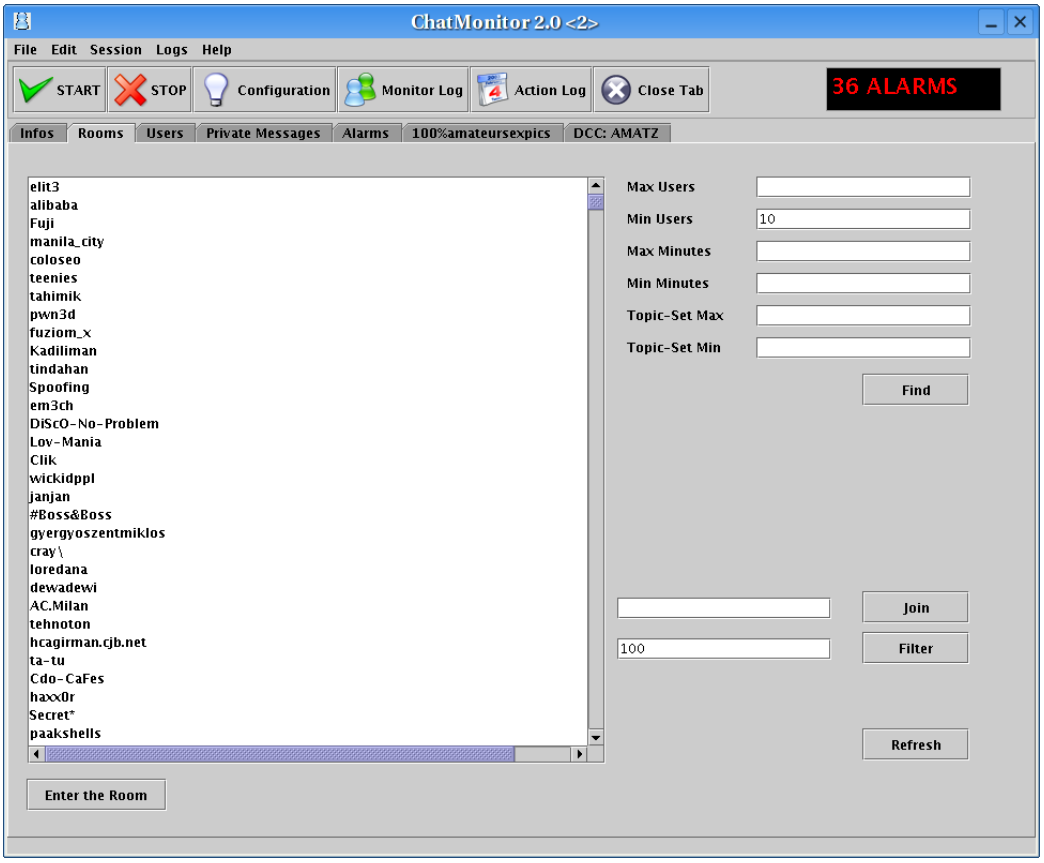

FIG. 9.2 – Recherche de canaux selon divers critères.

Private Messages Ce panel contient tous les messages privés envoyés par d'autres utilisateurs des canaux auxquels *Chat Monitor* est connecté (*cf.* Fig. 9.4). En grande partie, ces messages sont envoyés par des robots lorsqu'on se connecte à un canal. Il suffit de cliquer sur la liste à gauche pour voir le contenu du message s'afficher à droite.

Alarms Ce panel affiche les alarmes générées par le programme (*cf.* Fig. 10.1). En cliquant sur la liste de gauche, un rapide aperçu du fichier ayant généré l'alerte est affiché dans la partie de droite. Plusieurs informations y figurent : le nom du fichier incriminé ainsi que sa taille, le nom de son possesseur, son *UUID*, son pays probale, et l'*IP* du *FSERV* contenant le fichier. Le nombre d'alarmes levées est également affichée en haut à droite.

Boutons

La rangée de boutons au-dessus des onglets permet un accès rapide aux fonctionnalités intéressantes du logiciel :

Start ce bouton permet de démarrer une session de surveillance, après avoir préalablement configuré *Chat Monitor*. C'est à ce moment que *Chat Monitor* se connecte sur un serveur *IRC*.

Stop ce bouton permet de se déconnecter du serveur *IRC*.

Configuration ce bouton ouvre la fenêtre qui permet de configurer *Chat Monitor*.

Monitor Log ce bouton ouvre une fenêtre affichant les logs courants de la session de surveillance. Le contenu de cette fenêtre est sauvé à intervalles réguliers dans le fichier `MonitorLog.txt`.

Action Log ce bouton ouvre une fenêtre affichant les actions courantes effectuées par *Chat Monitor*. Le contenu de cette fenêtre est sauvé à intervalles réguliers dans le fichier `ActionLog.txt`.

Close Tab ce bouton ferme l'onglet actif.

Menus déroulants

Toutes les fonctionnalités de *Chat Monitor* sont diponibles à partir des menus déroulants :

File / Exit termine la session et ferme le logiciel.

Edit / Configuration ouvre la fenêtre qui permet de configurer *Chat Monitor*.

Logs / Actions ouvre une fenêtre affichant les actions courantes effectuées par *Chat Monitor*. Le contenu de cette fenêtre est sauvé à intervalles réguliers dans le fichier `ActionLog.txt`.

Logs / Surveillance ouvre une fenêtre affichant les logs courants de la session de surveillance. Le contenu de cette fenêtre est sauvé à intervalles réguliers dans le fichier `MonitorLog.txt`.

Logs / Save Log force la sauvegarde de tous les logs sur le disque dur.

Logs / Generate HTML outpout ouvre une fenêtre permettant de choisir le répertoire contenant la session à traiter. Un message de confirmation apparaît lorsque l'opération est achevée.

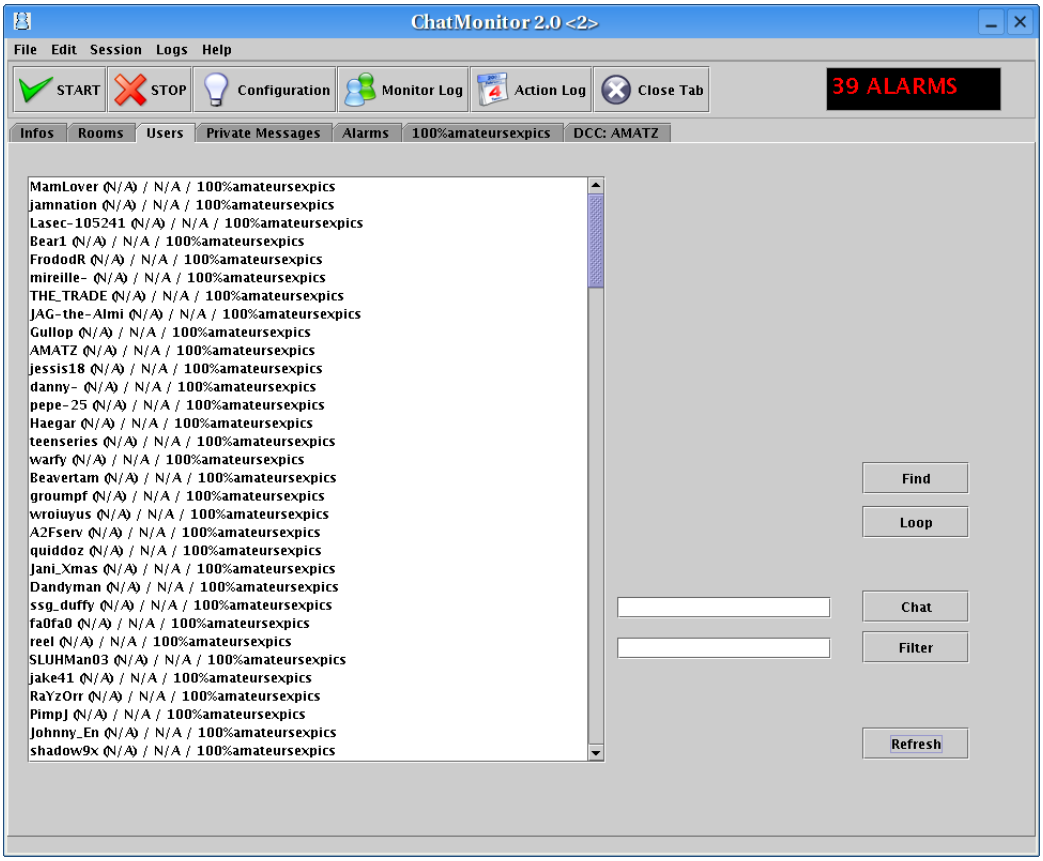

FIG. 9.3 – Liste des utilisateurs d'un serveur *IRC*.

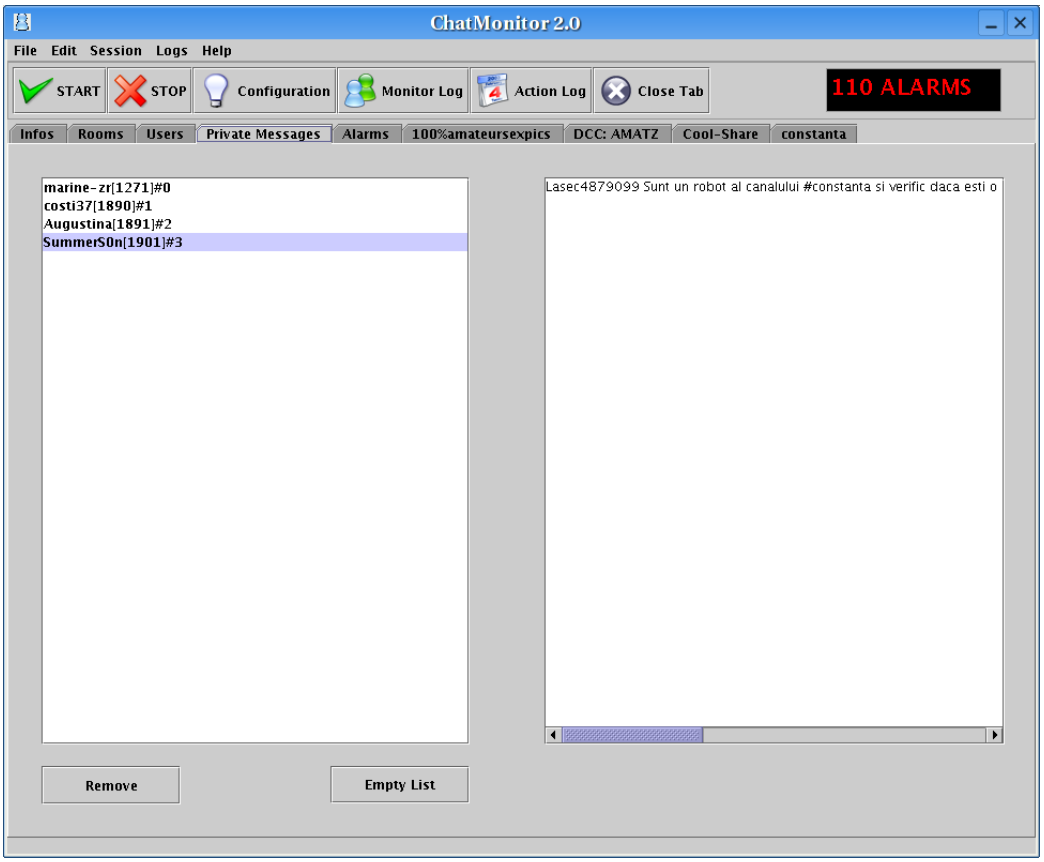

FIG. 9.4 – Liste des messages privés envoyés par d'autres utilisateurs.

10 Fonctionnalités

10.1 Alarmes

Une des caractéristiques intéressantes de *Chat Monitor* est qu'il est possible de lister des noms de fichiers connus pour être du matérial illégal. Lors du parcours des *FSERVs* , *Chat Monitor* analyse chaque nom de fichier rencontré et le compare avec la liste fournie par l'utilisateur. Si un de ces fichiers corresponds, alors une alerte est levée. Le compteur d'alarme (en haut à gauche, *cf.* Fig. 10.1) s'incrémente, et une alarme est ajoutée dans l'onglet correspondant.
Chat Monitor charge le contenu du fichier `alarms.txt` à chaque démarrage. C'est dans ce dernier qu'il faut inclure les noms de fichiers à filtrer.
Par exemple, prenons le contenu suivant pour `alarms.txt` :

```
preteen
teen.jpg
old.png
01.jpg
```

Prenons la première ligne, *Chat Monitor* va générer une alerte pour les fichiers suivants :

- `preteen.png`
- `imabeautifulpreteen.jpg`
- `cutepreteen_withothergirls.mpg`
- ...

On remarque que tous les noms de fichiers contenant la chaîne de caractères « preteen » lèvent une alerte. La dernière ligne du fichier va lever une alarme pour ce genre de fichiers :

- `01.jpg`
- `kimy01.jpg`
- `naked01.jpgoutdoor.png`
- ...

Plus les entrées dans `alarms.txt` sont génériques, plus le nombre d'alerte sera élevé. Une entrée comme `softpornwithnadiainasmallbed19990404_21.jpg` générera une alarme plus ciblée.

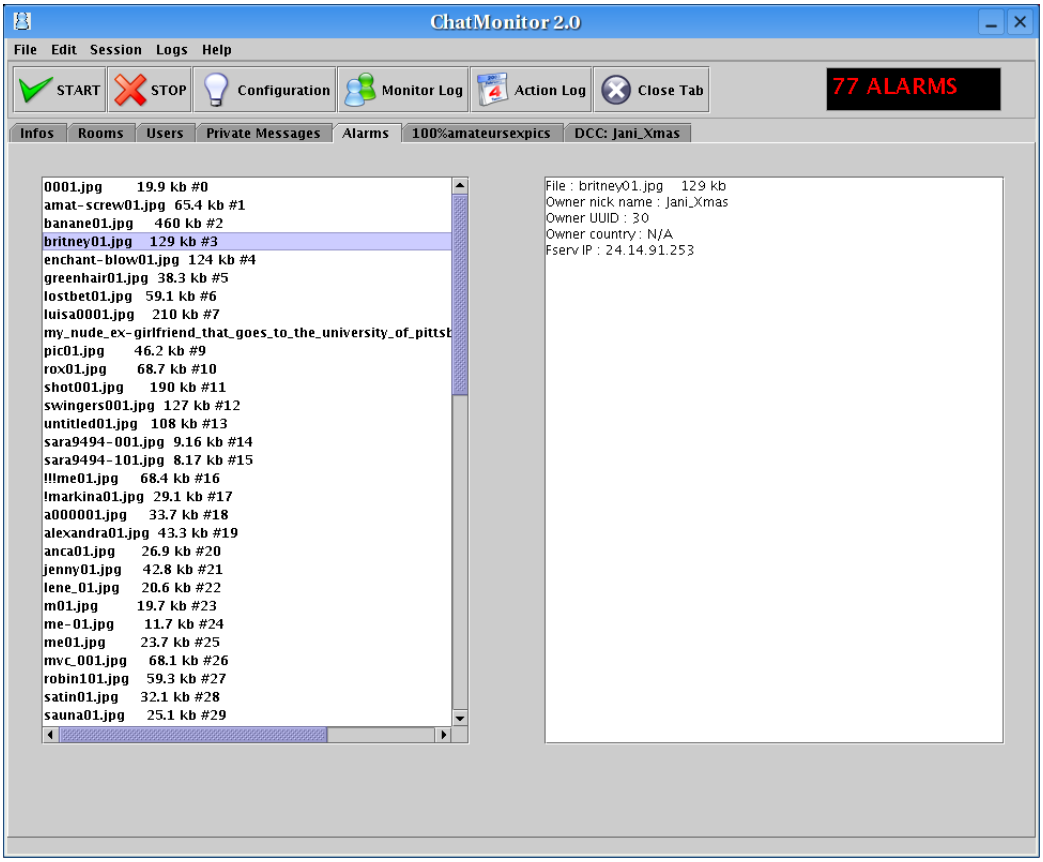

FIG. 10.1 – Liste des alarmes levées par *Chat Monitor*.

Le fichier `alarms.txt` doit se trouver à l'endroit même où on lance le programme. Il peut être modifié à l'aide d'un simple éditeur de texte.

Les alarmes levées sont visibles dans l'onglet « Alarms » et sont sauvegardées à intervalles réguliers dans le fichier `generatedAlarms.txt` situés dans le répertoire courant de la session de surveillance.

10.2 Rapport HTML

Une autre fonctionnalité intéressante de *Chat Monitor* est le fait qu'il est capable de générer un rapport facilement compréhensible à partir des logs bruts et des logs *XML*. Pour utiliser cette fonctionnalité, il suffit de sélectionner le menu « Logs » et de cliquer sur l'entrée « Generate HTML output ». Une fenêtre s'ouvre et vous permet de spécifier le répertoire dont

vous voulez un rapport. Un message vous informe losrque cette opération est terminée.

Chat Monitor génère un fichier `index.html` dans le répertoire session/*répertoire_session_choisie*/html. Ce fichier donne une vue globale de la surveillance effectuée durant la session.

11 | Exemple d'utilisation

Voici un court exemple d'une session de surveillance classique.

11.1 Démarrage de la session

Une fois la configuration terminée, il est possible de démarrer la surveillance en cliquant sur le bouton « Start » de l'interface principale (*cf.* Fig. 8.1) ou en sélectionnant l'option « Start » du menu « Session ». Des informations concernant la connection s'affichent dans le premier champ texte de l'onglet « Info ».

11.2 Choix des canaux

La connection au serveur *IRC* établie, il suffit de cliquer sur l'onglet « rooms » afin de chercher une liste de canaux susceptibles de nous intéresser. Par exemple, dans le champs « Min Users », on entre la valeur « 10 », puis pour lancer la recherche, on clique sur le bouton « Find ».
Après quelques secondes, de nombreux canaux apparaissent. Il est possible d'appliquer un filtre afin de restreindre les canaux disponibles. Pour cela, on entre par exemple « 100 » dans le champs situé près du bouton « Filter », puis on clique sur ce dernier. Le nombre de canaux disponibles s'est fortement réduit. Par exemple pour accéder au canal « 100%Amateursexpics », on le sélectionne à l'aide de la souris, puis on clique sur le bouton « Enter the Room ».
En cliquant sur l'onglet nouvellement créé, on accède aux discussions en cours sur ce canal.

11.3 Génération du rapport HTML

Après un certain temps, on voudra probablement avoir un compte-rendu de la session de serveillance. Pour cela, il suffit de choisir l'option « Generate

HTML output » du menu « logs ». Une fois le message de confirmation affiché, il faut ouvrir le fichier `index.html` du répertoire `html`.