2Steps2Hell
presents

version 1.1

N3m3s1s (Francesco Nwokeka), Zamy (Simone Daminato)

2Steps2Hell

www.2s2h.com

Indice

1 Italiano

1.1 Introduzione

Questa è la prima release ufficiale di *BanBot*. BanBot è un bot fatto in C++ che mira ad essere semplice, efficiente e ad occupare poche risorse della macchina che lo hosta limitando (addirittura eliminando) lag non voluti del server spesso provocati da questo genere di programmi.
Si tratta di un progetto open-source, con licenza GNU-Gpl (a questo proposito, leggere la licenza inclusa, o in alternativa la potete trovare qui).
Integra al suo interno le librerie SQLite3:

> SQLite3 Copyright (C) 1994, 1995, 1996, 1999, 2000, 2001, 2002, 2004, 2005 Free Software Foundation, Inc.

Pensato soprattutto per gli amministratori di sistema e dei game server di UrT, fornisce varie facilitazioni e utili funzionalità che vedremo in dettaglio nei successivi paragrafi. Può benissimo controllare più di un server con una sola istanza, anche in presenza di ambienti chroot, dove ovviamente deve avere i permessi di accesso.

Vi invitiamo a inviarci le vostre opinioni e idee, o le vostre esigenze, per poter migliorare sempre di più il bot (fate riferimento alla sezione *Assistenza e bug report* 1.6.5).

1.2 Funzionalità

Le funzionalità sono divise in due tipologie: di sistema e di controllo.

1.2.1 Funzionalità di sistema

Riguardano le caratteristiche del bot che non influiscono direttamente sul server di gioco, ma che presentano comunque dei vantaggi.

File di configurazione: la configurazione viene fatta attraverso uno o più file, in modo da poter essere versatile e dividere parti di configurazione in file diversi;

Reload automatico delle impostazioni: ogni tanto il bot controlla i file di configurazione e ricarica automaticamente le impostazioni se queste sono state cambiate, senza bisogno di essere riavviato;

Separazione totale tra server: il bot può controllare più di un server con una sola istanza, tenendoli però completamente separati. Ognuno ha la propria configurazione, le proprie cartelle (dei file di log e di backup), il proprio database, ecc;

Installazione facile e veloce: BanBot viene fornito già compilato, basta copiare l'eseguibile sul server, configurarlo con attenzione ed avviarlo: a creare i file e le cartelle necessari ci pensa lui;

Backup: ogni giorno, alle 5.30 AM, viene fatto un backup automatico di tutti i file gestiti dal bot (come file di log del game, file di log del bot, database) in cartelle divise per giorni;

Utilizzo delle risorse: BanBot è ottimizzato per occupare meno risorse possibile. Tra le altre cose, se i server di gioco sono vuoti e/o non accade niente di interesse per il bot, si mette da solo in "stand-by", diventando quasi inesistente.

1.2.2 Funzionalità di controllo

Riguardano le caratteristiche del bot che influiscono direttamente sul gioco o sui giocatori.

Riconoscimento amministratori: il bot risponde solo ed esclusivamente agli amministratori, ovvero gli utenti registrati al bot, indipendentemente dal loro nick (possono essere anche in fake, il bot risponderà comunque).

Comandi di amministrazione: sono disponibili parecchi comandi utili, come per esempio slap, kick, ban, veto, map, ecc ...

Ban permanenti: tramite BanBot è possibile bannare definitivamente un player, tramite nick, guid e ip (a seconda del livello di sicurezza). Per ogni ban viene registrato anche chi lo ha fatto e l'eventuale motivo.

Anticheat: BanBot è in grado di rilevare da solo alcuni cheat, aiutando quindi gli amministratori a tenere il server di gioco più libero da persone non volute.

Diversi livelli di sicurezza: ci sono diversi livelli di sicurezza, con caratteristiche diverse, in modo da poter scegliere la politica di gestione che si ritiene più corretta.

Facilità d'uso: la maggior parte dei comandi sono pensati per essere di facile ed immediato utilizzo, ed accettano indistintamente sia il numero del giocatore, sia il suo nick (oppure anche solo una parte di esso).

1.3 Installazione

Per utilizzare BanBot dobbiamo avere sul server la possibilità di caricare dei file e di eseguire comandi: se manca anche solo uno di questi requisiti non sarà possibile utilizzarlo (come del resto quasi tutti i bot).

Vediamo ora come installare il bot (su sistema operativo linux): tutti i comandi indicati sono da eseguire su un terminale.

Innanzitutto, se non ce l'avete già, scaricate il bot dalla sezione apposita del sito del clan 2Steps2Hell. All'interno dell'archivio scaricato troverete questo manuale (all'interno della cartella doc), l'eseguibile già compilato del bot (dovrebbe essere compatibile con la maggior parte dei server), una copia della licenza, ed il file di configurazione (all'interno della cartella cfg).

1.3.1 Copia dei file & permessi cartelle

Estraete il contenuto dell'archivio sul vostro server, dove preferite, e controllate i permessi in modo che il file BanBot sia eseguibile. A questo scopo, della cartella dove si trova il bot, eseguite il comando

```
1 ls -l
```

tra le righe risultanti, quella che ci interessa è della cartella dove si trova BanBot, simile a questa:

```
1 drwxrwxr-x  1 user user 1644143 Sep  1 15:12 BanBot
```

ed in particolare i permessi. Per permettere al bot di fare il suo lavoro, dobbiamo avere i permessi di lettura(r), scrittura(w) ed esecuzione(x), come indicato nella prima colonna

drwxrwxr-x

oppure

drwxrwx—

Nel caso i permessi del vostro file siano diversi, eseguite il comando:

```
1 chmod -r 775 BanBot/
```

e controllate il risultato dell'operazione.

1.3.2 Configurazione per un singolo server

Vediamo come configurare BanBot per controllare un singolo server.

Per questo scopo è sufficiente modificare il file di configurazione (cfg/BanBot.cfg) inserendo i dati corretti; consigliamo di usare sempre dove possibile dei percorsi assoluti e non relativi (per esempio, "/home/user/doc" invece di "doc").

GENERAL_BOTLOG : indica il file dove salvare il log generale del bot (dove generalemente sono contenuti notifiche ed errori).

GENERAL_BACKUP_PATH : indica la cartella dove fare il backup del file indicato in GENERAL_BOTLOG.

SERVER_NAME : un nome arbitrario da dare a server (potete scrivere quello che volete, non influisce sul funzionamento del bot).

IP : ip del server di gioco da controllare (ip visto dai player, non inserite indirizzi locali come "localhost" o "127.0.0.1").

PORT : porta del server di gioco da controllare.

RCON_PASSWORD : password rcon impostata sul server di gioco.

GAME_LOGFILE : file di log del server di gioco (solitamente si chiama "games.log" e si trova nella cartella ".q3a/q3ut4/").

BOT_LOGFILE : indica il file dove salvare il log del bot relativo a questo server di gioco.

BACKUP_DIR : la cartella dove eseguire il backup di tutti i file relativi a questo server.

DATABASE_DIR : la cartella dove creare il database necessario al bot.

STRICT_LEVEL : indica il livello di sicurezza e restrizioni da applicare di default (per maggiori dettagli a questo proposito, vedere la sezione apposita).

Tutti i file e cartelle inesistenti verranno creati automaticamente dal bot (in caso di errori e problemi, controllate le impostazioni e i permessi), ad eccezione di "GAME_LOGFILE".

1.3.3 Configurazione per più server

Ci sono principalmente tre metodi per configurare BanBot in modo da controllare più server.

In generale, si possono inserire tutti i parametri di configurazione direttamente nel file di configurazione primario (o generale), oppure utilizzare anche dei file di configurazione secondari, contenenti tutte le impostazioni o solo una parte, ognuno riguardante un singolo server. Questo secondo sistema si presenta estremamente utile soprattutto in caso di più utenti distinti, che così possono modificare le impostazioni riguardanti il loro server (per esempio la password rcon se la cambiano) senza poter vedere nè tantomeno modificare quelle degli altri.

Metodo 1: tutto nel file di configurazione principale

Per ogni server che desideriamo configurare, creiamo un blocco contenente tutte le impostazioni, dove ogni blocco è delimitato da parentesi graffe ("{" e "}"). Per le descrizioni delle impostazioni, riferirsi alla sezione 1.3.2. Vediamo un esempio:

```
1 GENERAL_BOTLOG=/home/admin/BanBot/logs/BanBot.log
2 GENERAL_BACKUP_PATH=/home/admin/BanBot/backup/
3 {
```

```
4       SERVER_NAME = server1
5       IP = 81.27.6.123
6       PORT = 27960
7       RCON_PASSWORD = yeah!
8       GAME_LOGFILE = /home/user1/.q3a/q3ut4/games.log
9       BOT_LOGFILE = /home/user1/BotLog.log
10      BACKUP_DIR = /home/user1/old_logs/
11      DATABASE_DIR = /home/user1/database/
12      STRICT_LEVEL = 0
13  }
14
15  {
16      SERVER_NAME = server2
17      IP = 81.27.6.123
18      PORT = 27961
19      RCON_PASSWORD = rconpass
20      GAME_LOGFILE = /home/user2/.q3a/q3ut4/public.log
21      BOT_LOGFILE = /home/user2/BotLog_public.log
22      BACKUP_DIR = /home/user2/backup/
23      DATABASE_DIR = /home/user2/database/
24      STRICT_LEVEL = 1
25  }
26
27  {
28      ...
29  }
```

Le restrizioni a cui sono soggette le configurazioni sono:

- i server devono avere nomi diversi.
- le cartelle di backup devono essere diverse (sia quelle dei singoli server, sia quella generale del bot).
- ogni server deve avere un diverso database.
- ogni file di log del server di gioco deve essere il log di un solo server (avere due server di gioco che salvano il log sullo stesso file, oltre ad essere inutile, crea parecchi problemi).

Metodo 2: tutto diviso in file di configurazione secondari

Vediamo ora l'utilizzo dei file di configurazione secondari: in pratica, creiamo un file per ogni server con dentro le sue impostazioni, e nel file di configurazione generale diciamo a BanBot dove andare a prendere i dati. Per fare questo utilizziamo la direttiva:

CONFIG_FILE : utilizzabile solo all'interno del file di configurazione principale all'interno del blocco di un server, indica al bot il file dove andare a cercare le impostazioni.

Anche nel file di configurazione secondario tutte le impostazioni vanno racchiuse in un blocco delimitato dalle parentesi graffe. Vediamo un esempio:

File di configurazione principale

```
GENERAL_BOTLOG=/home/admin/BanBot/logs/BanBot.log
GENERAL_BACKUP_PATH=/home/admin/BanBot/backup/
{
    CONFIG_FILE = /home/user1/BanBot.cfg
}
{
    CONFIG_FILE = /home/user2/BanBot.cfg
}
```

File di configurazione secondario: /home/user1/BanBot.cfg

```
{
    SERVER_NAME = server1
    IP = 81.27.6.123
    PORT = 27960
    RCON_PASSWORD = yeah!
    GAME_LOGFILE = /home/user1/.q3a/q3ut4/games.log
    BOT_LOGFILE = /home/user1/BotLog.log
    BACKUP_DIR = /home/user1/old_logs/
    DATABASE_DIR = /home/user1/database/
    STRICT_LEVEL = 0
}
```

File di configurazione secondario: /home/user2/BanBot.cfg

```
{
    SERVER_NAME = server2
    IP = 81.27.6.123
    PORT = 27961
    RCON_PASSWORD = rconpass
    GAME_LOGFILE = .q3a/q3ut4/public.log
    BOT_LOGFILE = BotLog_public.log
    BACKUP_DIR = backup/
    DATABASE_DIR = database/
    STRICT_LEVEL = 1
}
```

Come si può notare, sui file secondari è possibile utilizzare percorsi sia assoluti (server1) che relativi (server2). In particolare, per i percorsi relativi nei file secondari, BanBot considera come punto di partenza la cartella del file di configurazione secondario, ovvero se abbiamo:

/cartella1/cartella2/file_configurazione_secondario.cfg

un eventuale indirizzo relativo nel file secondario sarà tradotto così:

/cartella1/cartella2/indirizzo_relativo

Quindi, prendendo per esempio la proprietà GAME_LOGFILE di *server2*, BanBot la tradurrà in

/home/user2/.q3a/q3ut4/public.log

Metodo 3: misto

Il terzo metodo consiste in una combinazione tra i due sistemi. In pratica si possono mettere per ogni server alcuni parametri nel file di configurazione principale, ed alcuni in un file di configurazione secondario.

Questo torna utile se vogliamo che l'utente possa cambiare la password rcon e magari il file di log sulla configurazione, ma non altri parametri come la cartella di backup. Vediamo un esempio:

File di configurazione principale

```
1  GENERAL_BOTLOG=/home/admin/BanBot/logs/BanBot.log
2  GENERAL_BACKUP_PATH=/home/admin/BanBot/backup/
3  {
4      SERVER_NAME = server1
5      IP = 81.27.6.123
6      PORT = 27960
7      BOT_LOGFILE = /home/user1/BotLog.log
8      BACKUP_DIR = /home/user1/old_logs/
9      DATABASE_DIR = /home/user1/database/
10     CONFIG_FILE = /home/user1/BanBot.cfg
11 }
12
13 {
14     SERVER_NAME = server2
15     IP = 81.27.6.123
16     PORT = 27961
17     BOT_LOGFILE = /home/user2/BotLog_public.log
18     BACKUP_DIR = /home/user2/backup/
19     DATABASE_DIR = /home/user2/database/
20     CONFIG_FILE = /home/user2/BanBot.cfg
21 }
```

File di configurazione secondario: /home/user1/BanBot.cfg

```
1  {
2      RCON_PASSWORD = yeah!
3      GAME_LOGFILE = .q3a/q3ut4/games.log
4      STRICT_LEVEL = 0
5  }
```

File di configurazione secondario: /home/user2/BanBot.cfg

```
1 {
2     RCON_PASSWORD = rconpass
3     GAME_LOGFILE = .q3a/q3ut4/public.log
4     STRICT_LEVEL = 1
5 }
```

In questa maniera l'utente ha il suo file di configurazione dove può cambiarsi alcuni parametri, senza vedere gli altri.

Da notare che devono però essere presenti sempre tutti i parametri, le impostazioni incomplete o non considerate valide verranno ignorate. Altra considerazione importante: evitate di inserire due volte una parametro all'interno di uno stesso server, perchè in questo caso l'ultimo inserimento sovrascriverà i precedenti. Inoltre, i parametri inseriti sui file di configurazione secondari vengono esaminati per ultimi, sovrascrivendo eventuali precedenti.

1.3.4 Configurazione intelligente del game server

Il più delle volte, anche per il regolamento dei tornei, non si vuole che ci siano bot attivi quando il server è privato.
Sicome il bot controlla il server seguendo solo il file di log indicato, si può facilmente disattivare il bot cambiando nome al file di log del game server all'interno della cfg che lo imposta come privato.

Per fare un esempio, nella cfg *public.cfg* che imposta il server come pubblico, si potrebbe impostare la variabile

set g_log "games.log"

mentre nella cfg *pcw.cfg*, che imposta il server privato per le clan war, si potrebbe mettere

set g_log "pcw.log"

In questo modo il bot non farebbe nulla se il server viene messo privato.

1.3.5 Lancio del bot

Dopo la configurazione, siamo pronti ad avviarlo. Andiamo nella cartella del bot, ed eseguiamo il comando:

screen

che ci aprirà un nuovo terminale che *non si chiuderà* quando chiuderemo la connessione. A questo punto, lanciamo il bot:

./BanBot

Controlliamo velocemente che sia tutto a posto (opzioni caricate, nessun messaggio di errore nel creare le cartelle, ecc.) e nel caso correggiamo le impostazioni e/o i permessi delle cartelle, come già visto sopra.

Bene, il bot è avviato!

Ora non ci resta che entrare sui server di gioco e scrivere in chat il comando

!iamgod

per prenderne il controllo: il bot ci risponderà con un *"Welcome, my master!"*. Questa operazione è da fare (ed è possibile farla) solo quando si fa controllare al bot un nuovo server. In caso di riavvio del bot, gli amministratori resteranno salvati.

N.B.: In virtù del fatto che il bot non deve interferire con le partite nei vari server, ha un sistema di avvio "soft". In pratica il bot si accorge e riconosce i giocatori man mano che questi inviano al server le proprie informazioni (per capirci, la riga ClientUserInfo dei log), e quindi gli eventuali giocatori già in gioco verranno visti dal bot solo quando cambieranno nick, cambieranno armi e/o cambieranno team, oppure al caricamento della mappa successiva. Quindi a bot appena avviato è perfettamente normale che alcuni giocatori (se non tutti) non riuscite a kickarli, slapparli, ecc...

1.4 Sicurezza: strict level

Lo strict level è un parametro che permette di scegliere il livello di sicurezza e delle restrizioni che il bot applicherà sul nostro server.
Ogni livello ha le seu caratteristiche, ed è sempre più restrittivo man mano che sale.

1.4.1 Strict level 0

Livello di base, non è attivo nessun controllo anticheat.
Semplicemente il bot si limita ad eseguire i comandi dati dagli amministratori, e a tenere fuori le persone bannate.

Il ban per guid è permanente, mentre quello per ip e per nick dura solo un'ora. Nel caso entri una persona con un nick bannato da più di un'ora, il bot si limita ad avvisare gli admin in gioco con un messaggio.

1.4.2 Strict level 1

Equivalente del livello precedente per quanto riguarda i ban, ma a questo livello sono attivi i controlli anticheat di base.
In caso di cheat sicuro, banna automaticamente. Nei (rari) casi dubbi, avvisa gli admin presenti nel server, con un messaggio privato, del possibile cheater, senza fare nulla.
Gli admin decideranno loro se bannarlo o meno.

1.4.3 Strict level 2

A questo livello abbiamo i controlli anticheat di base attivati, ed il bot è molto più selettivo: i ban, sia per guid, che per nick, che per ip, sono eterni, non hanno più limiti di tempo.
Inoltre, anche nei casi dubbi per cheat, non avvisa gli admin, banna e butta fuori direttamente.

Una nota a questo punto: a causa del ban perenne dei nick, potrebbe essere buttata fuori anche gente che non ha colpe, magari perchè si è messa un nick che era stato bannato. Per i livelli di sicurezza superiori allo *strict 1* il bot potrebbe mietere anche alcune vittime innocenti, sia per i casi dubbi, sia per i ban permanenti: quindi qualsiasi responsabilità è vostra, pensateci bene.

1.5 Comandi del bot

Andiamo ora a vedere comando per comando a cosa serve e come si usa. Innanzitutto, tutti i comandi si utilizzando tramite "say", ovvero la chat pubblica del gioco; inoltre il bot, per qualsiasi comando, risponde solo agli amministratori (ad eccezione del comando speciale *!iamgod*): Nella spiegazione dei comandi, i simboli minore e maggiore ('<"' e ">") indicano un parametro obbligatorio, mentre le parentesi quadre ("[" e "]") ne indicano uno opzionale.

1.5.1 !admins

Utilizzo:

!admins

mostra un elenco dei giocatori attualmente in game che sono registrati come admin al bot.

1.5.2 !ban

Utilizzo:

!ban <nick/numero> [motivo]

banna permmanentemente un player attualmente in gioco.
Accetta un numero (l'id del giocatore, si vede con il comando "clientlist" sul terminale), o in alternativa il nick o una sua parte (che ovviamente deve identificare un solo giocatore).
Per esempio, se vogliamo bannare questo giocatore (dove 9 è il numero del giocatore) :

9 [AlFd]Prot

Possiamo usare uno di questi comandi:

```
1 !ban 9
2 !ban 9 non lo voglio tra i piedi
3 !ban Prot
4 !ban [AlFd]
5 !ban ]prot fuori dalle palle!
```

abbiamo usato 2 motivi di esempio, sulle righe 2 e 5. Da notare che nel caso ci sia per esempio un giocatore chiamato "Protect", il comando alla riga 3 non funzionerà: il bot risponderà con un "giocatore non trovato o nick non univoco", perchè appunto ci sono due giocatori il cui nick contiene "Prot". In questi casi usate un'altra parte del nick, o scrivete tutto il nick, o alla peggio, usate il numero del giocatore.

1.5.3 !deop

Utilizzo:

!deop <id>

toglie lo stato di amministratore al player identificato dal bot con quell'id. Per vedere quale id viene assegnato ad un giocatore, utilizzare il comando *!findop*.

1.5.4 !find

Utilizzo:

!find <nick>

ricerca il nick inserito tra i bannati, e ci mostra il risultato della ricerca. Principalmente serve per permettere di vedere il numero associato dal bot ad una persona bannata, per poterlo sbannare. Una risposta di esempio potrebbe essere:

!find Z

```
1 Ricerca esatta: none.
2 Ricerca:
3 3 [2s2h]Zamy by [2s2h]n3m3s1s,
4 10 Zinko by [2s2h]Zamy rompicoglioni.
```

Nella risposta ci sono 4 colonne: id, nick, admin che ha effettuato il ban, e l'eventuale motivo. L'id è il numero che ci serve per alcuni comandi particolari. Da notare che BanBot tiene in memoria solo l'ultimo nick utilizzato dal giocatore.

1.5.5 !findop

Utilizzo:

!findop <nick>

ricerca il nick inserito tra gli admin, e ci mostra il risultato della ricerca. Principalmente serve per permettere di vedere il numero associato dal bot ad un amministratore, per poterlo deoppare. Una risposta di esempio potrebbe essere:

!findop Z

```
1 Ricerca esatta: none.
2 Ricerca:
3 3 [2s2h]Zamy,
4 10 Zenna.
```

Nella risposta ci sono due colonne: id e nick. L'id è il numero che ci serve per alcuni comandi particolari.

1.5.6 !force

Utilizzo:

!force <red/blue/spect> <nick/numero>

cambia team al giocatore indicato: si può spostare nel team rosso, blu o negli spettatori.

1.5.7 !help

Utilizzo:

!help

stampa una serie di messaggi con la lista dei principali comandi e il loro utilizzo.

1.5.8 !kick

Utilizzo:

!ban <nick/numero>

butta fuori dal server il giocatore indicato.

1.5.9 !iamgod

Utilizzo:

!iamgod

registra il giocatore che esegue il comando come amministratore. Funziona solo se non ci sono admin registrati al bot.

1.5.10 !map

Utilizzo:

!map <nome mappa>

cambia la mappa sul server: deve essere indicato il nome completo della mappa (il nome del fle della mappa).

1.5.11 !mute

Utilizzo:

!mute <nick/numero/all/ALL>

muta o smuta la persona indicata, se invece del nick o del numero viene scritto come parametro *all* oppure *ALL*, muta/smuta tutti ad eccezione di sè stessi.

1.5.12 !nextmap

Utilizzo:

!nextmap <nome mappa>

cambia la mappa successiva sul server: deve essere indicato il nome completo della mappa (il nome del fle della mappa).

1.5.13 !nuke

Utilizzo:

!nuke <nick/numero>

esegue un nuke sul giocatore indicato (numero o nick). Da notare che in UrbanTerror, se nel server è attivato il match mode, il nuke non funziona.

1.5.14 !op

Utilizzo:

!op <nick/numero>

registra il giocatore indicato come amministratore. Da notare che verrà registrato col suo nick attuale, quindi è consigliato mettersi il nick "ufficiale" prima di fare questa operazione.

1.5.15 !slap

Utilizzo:

!slap <nick/numero> [2-5]

slappa il giocatore indicato, una volta sola se il parametro opzionale non è indicato, o fino ad un massimo di 5.

Nota bene: se si richiede un numero di slap superiore a 5, il comando verrà ignorato.

1.5.16 !status

Utilizzo:

!status

visualizza la versione e lo stato del bot (numero di admins, di bannati, il livello strict attualmente attivo, ecc).

1.5.17 !strict

Utilizzo:

!strict <OFF/off/0/1/2>

cambia il livelo di strict attivo su questo server: *off* e *0* sono equivalenti.

1.5.18 !unban

Utilizzo:

!unban <id>

sbanna il player identificato dal bot con quell'id. Per vedere con quale id è stato bannato un giocatore, utilizzare il comando *!find*.

1.5.19 !veto

Utilizzo:

!veto

annulla la votazione in corso.

1.6 Risoluzione dei problemi e assistenza

1.6.1 Lentezza del bot

Alcune volte, soprattutto nel caso vengano controllati più server con una sola istanza, il bot ci mette qualche secondo a rispondere ai comandi: questo perchè il bot non utilizza più thread, e può quindi eseguire una sola operazione alla volta; per esempio, potrebbe dover finire di stampare la risposta al comando *!help* su un server, prima di rispondere ad un comando dato sul successivo (lo stesso discorso vale anche all'interno di uno stesso server). Questo problema verrà risolto in futuro.

1.6.2 Il bot all'inizio non funziona bene/non riconosce i giocatori

Come già detto, il bot ha una partenza "soft", e riconosce i giocatori man mano che reinviano le proprie informazioni al server. Se questo crea problemi, basta fare un reload della mappa, e il problema sarà subito risolto.

1.6.3 Il bot non risponde più ai comandi

In questo caso, possono esserci molti motivi: per prima cosa controlllate se non risponde solo a voi oppure anche agli altri admin. Se non risponde solo a voi vuol dire che per qualche motivo avete cambiato guid, e dovrete farvi registrare nuovamente al bot da un altro admin (e cancellate anche la vecchia registrazione).

Se invece non risponde a nessuno, possono esserci tre motivazioni:

- il bot è crashato.
- è stata cambiata la password rcon.
- è stato cambiato il file di log del server di gioco.

1.6.4 Il bot non funziona/non risponde al comando *!iamgod*

Se non è la prima volta che avviate il bot e ci sono già degli admin registrati, è normale che non risponda al comando *!iamgod*, dovete usare il comando !op. In caso di problemi, per esempio l'unico admin registrato ha cambiato guid, potete resettare il bot eliminando il database.

Negli altri casi, controllate le impostazioni: è molto probabile ce ci sia qualcosa di sbagliato. Leggete i messaggi di avvio del bot per capire meglio dov'è il problema.

1.6.5 Assistenza e segnalazione bug

Per assistenza e la segnalazione dei bug, abbiamo creato una sezione apposta nel nostro sito, dove si trova il bugtracker: qui, dopo la vostra registrazione, potrete aprire dei ticket per richiedere aiuto, segnalare dei bug, o semplicemente per darci la vostra opinione o nuove idee.
Usate **solo** il bugtracker, eventuali richieste di aiuto/assistenza sul nostro forum o per email, verranno ignorate.

2 English

2.1 Introduction

This is the first official release for *BanBot*. BanBot is a bot made in C++ that aims to be simple, efficient and to consume as few resources as possible of the hosting machine limiting (even eliminating) unwanted lags of the gameserver usually caused by programs of this type.
BanBot is an open-source project with GNU-Gpl license (to know more read the license file included in the project or you can visit this link here).
It integrates the SQlite3 libraries:

> SQLite3 Copyright (C) 1994, 1995, 1996, 1999, 2000, 2001, 2002, 2004, 2005 Free Software Foundation, Inc.

Developed for system and Urban Terror game server administrators, BanBot provides various functions that we'll see in depth later on.

It can easily control more than one server with one single instance even in chrooted systems, where it obviously MUST have the correct permissions.

We invite you to send us your ideas, impressions and wishlists so that we can make a better bot (checkout the *Assistance and bug report* 2.6.5).

2.2 Functions

The functions provided are divided in two types: system and control.

2.2.1 System Functions

These functions concern the bots features that don't directly influence the game server, but that bring some benefits.

Configuration file/s: the configuration is done with the help of one or more files so that it becomes easier work to do and more ordered;

Automatic settings reload: every now and then, the bot checks its configuration files and loads automatically any new settings without having to be restarted;

Server separation: the bot can control more than one server with one single instance keeping them completley separated. Every server has its own configuration files, folders (backup and database), database etc;

Fast and easy installation: BanBot comes precompiled, all you have to do is copy the executable on your server, configure it well and launch it. Don't worry about file and directory creation, BanBot does that by itself;

Backup: every day at 5:30AM an automatic backup of all files is made by the bot (game log, bot log, database) in different folders ordered by date;

Resource usage: BanBot is optimized to use little resources as possible. Among other things, if the game servers are empty and/or nothing interesting happens, BanBot automatically goes in "standby" becoming almost non existant.

2.2.2 Control functionality

These functions concern the bots features that influence directly on the game and players.

Administrator recognition: the bot answers only to the admins or the players registrated to the bot regardless of their nicks (yes, you can play with "fakes" and the bot will recognize you)

Administration commands: many useful commands are provided like : slap, kick, ban, veto, map etc ...

Permanent bans: through BanBot it is possible to definatley ban a player by nick, guid and ip (depends on the level of security chosen). Every ban given registers also who did it and eventualy, if specified, why.

Anticheat: BanBot is capable of detecting by itself some cheats, helping administrators keep the game server free from unwanted clients

Differen security levels: there are different security levels with different features letting administrators choose the way they want to administrate their server

Easy to use: the majority of the bots commands are designed to be easy and quick to use and accept both number and client nick (or a portion of it)

2.3 Installation

In order to use BanBot you must have the possibility to load files and execute commands on your server: if only one of these prerequisites is missing, you will not be able to use it (like all other bots).

Let's see how to control the bot (on linux operating systems): all the commands indicated are to be executed on a terminal.

First of all, if you don't have the bot, download it from the 2steps2hell webpage 2Steps2Hell. Inside the archive you'll find this manual(inside doc), the bot executable already compiled for the specific architecture, a copy of the license and the configuration file(inside cfg).

2.3.1 Copy of files & folder permissions

Extract the content of the archive to your server, wherever you prefer, and check the permissions so that the file "BanBot" is executable. At this point, from the bot folder, execute the following:

```
1 ls -l
```

from the result of this command we'll be looking for the line corrisponding to the output of the executable (should look like this):

```
1 drwxrwxr-x  1 user user 1644143 Sep  1 15:12 BanBot
```

To let the bot do it's job, the executable MUST have the permission to read(r), write(w) and execute(x) as indicated in the first column.

drwxrwxr-x

or

drwxrwx---

If the permissions of your file are different, execute the following command:

```
1 chmod -r 775 BanBot/
```

and check the result of the operation.

2.3.2 Configuration for a single server

Let's take a look on how to configure BanBot for a single server.

For this purpose it's sufficient to modify the configuration file (cfg/BanBot.cfg) inserting the correct data. We suggest you use, where possible, absolute paths and not relative ones (example: use "/home/user/doc" instead of "doc").

GENERAL_BOTLOG : indicates the file where to save the bots general log (bot notifications and errors)

GENERAL_BACKUP_PATH : indicates the folder where to make a backup of the file indicated by GENERAL_BOTLOG.

SERVER_NAME : a name to give to your server(you can write what you want, it doesn't influence the operations of the bot)

IP : ip of the server to survey (ip seen by the players, don't use local ip's like "localhost" or "127.0.0.1").

PORT : game server port to survey.

RCON_PASSWORD : rcon password of the game server.

GAME_LOGFILE : game server log file (usally named "games.log" and is found inside ".q3a/q3ut4/").

BOT_LOGFILE : indicates the file where to save the bots log related to the current server under surveys

BACKUP_DIR : folder where to backup server and bot files.

DATABASE_DIR : folder where the bot should create it's database.

STRICT_LEVEL : indicates the level of security restriction to apply by default (for more information see specified section).

All files and folders that don't exist will be created automatically by the bot (in case of errors, check your setup and directory permissions), except for "GAME_LOGFILE".

2.3.3 Configuration for more than one server

There are three ways for configuring BanBot so that it can control more than one server.

Usually you can insert all the configuration parameters directly into the primary configuration file otherwise you can use "secondary" files that have complete or partial information regarding different servers. This second method is extremely useful especially when having more than one user (a server each in case you're a sysadmin hosting game servers) letting the users modify the bot settings regarding their server (for example their rcon password) without compromising the other users configuration.

Method 1: all in one configuration file

For every server you want to configure you'll have to create a block containing all the settings, where every block is delimited by brackets("{" and "}"). For the settings descriptions, see 2.3.2. Let's see an example:

```
1 GENERAL_BOTLOG=/home/admin/BanBot/logs/BanBot.log
2 GENERAL_BACKUP_PATH=/home/admin/BanBot/backup/
3 {
4     SERVER_NAME = server1
5     IP = 81.27.6.123
6     PORT = 27960
7     RCON_PASSWORD = yeah!
8     GAME_LOGFILE = /home/user1/.q3a/q3ut4/games.log
9     BOT_LOGFILE = /home/user1/BotLog.log
```

```
10      BACKUP_DIR = /home/user1/old_logs/
11      DATABASE_DIR = /home/user1/database/
12      STRICT_LEVEL = 0
13  }
14
15  {
16      SERVER_NAME = server2
17      IP = 81.27.6.123
18      PORT = 27961
19      RCON_PASSWORD = rconpass
20      GAME_LOGFILE = /home/user2/.q3a/q3ut4/public.log
21      BOT_LOGFILE = /home/user2/BotLog_public.log
22      BACKUP_DIR = /home/user2/backup/
23      DATABASE_DIR = /home/user2/database/
24      STRICT_LEVEL = 1
25  }
26
27  {
28      ...
29  }
```

The restrictions under which the settings go are:

- the servers MUST have different names.
- the backup folders MUST be different (both single server ones and bot general backup one).
- every server MUST have a different database.
- every gameserver logfile must be the logfile of a SINGLE server (having two servers write on the same logfile is useless and creates problems).

Method 2: use of secondary configuration files

Let's see how to use secondary configuration files: all you have to do is create a file for every server with it's settings. Done this all you'll have to do is tell BanBot where to get the server settings data from. To do this use:

CONFIG_FILE : usable only inside the settings block of a server inside the main configuration file. It tells the bot from which file to load it's settings from.

Also in the secondary configuration file, all settings must be delimited by brackets. Let's see an example:

Main configuration file

```
1  GENERAL_BOTLOG=/home/admin/BanBot/logs/BanBot.log
2  GENERAL_BACKUP_PATH=/home/admin/BanBot/backup/
3  {
4      CONFIG_FILE = /home/user1/BanBot.cfg
5  }
```

```
6 {
7     CONFIG_FILE = /home/user2/BanBot.cfg
8 }
```

Secondary configuration file: /home/user1/BanBot.cfg

```
1  {
2      SERVER_NAME = server1
3      IP = 81.27.6.123
4      PORT = 27960
5      RCON_PASSWORD = yeah!
6      GAME_LOGFILE = /home/user1/.q3a/q3ut4/games.log
7      BOT_LOGFILE = /home/user1/BotLog.log
8      BACKUP_DIR = /home/user1/old_logs/
9      DATABASE_DIR = /home/user1/database/
10     STRICT_LEVEL = 0
11 }
```

Secondary configuration file: /home/user2/BanBot.cfg

```
1  {
2      SERVER_NAME = server2
3      IP = 81.27.6.123
4      PORT = 27961
5      RCON_PASSWORD = rconpass
6      GAME_LOGFILE = .q3a/q3ut4/public.log
7      BOT_LOGFILE = BotLog_public.log
8      BACKUP_DIR = backup/
9      DATABASE_DIR = database/
10     STRICT_LEVEL = 1
11 }
```

As you can see, you can use both absolute (server1) or relative (server2) paths. For the relative paths in the secondary configuration files, BanBot considers the secondary files folder as starting point. For example, if you have:

/folder1/folder2/secondaryFile1.cfg

the corrisponding relative path in the secondary file will be something like this:

/folder1/folder2/relative_path

Looking at the setting GAME_LOGFILE of *server2* BanBot will translate it to

/home/user2/.q3a/q3ut4/public.log

Method 3: mix

The third method consists in a combination of the previous two. With this method, you can set some parameters for every server in the primary configuration file and then set the others in the secondary configuration file.

This is useful if we want to limit the settins of the user letting him/her change rcon password and log file, but not the rest like for example the backup folder or the database folder. Let's see an example:

Main configuration file

```
GENERAL_BOTLOG=/home/admin/BanBot/logs/BanBot.log
GENERAL_BACKUP_PATH=/home/admin/BanBot/backup/
{
    SERVER_NAME = server1
    IP = 81.27.6.123
    PORT = 27960
    BOT_LOGFILE = /home/user1/BotLog.log
    BACKUP_DIR = /home/user1/old_logs/
    DATABASE_DIR = /home/user1/database/
    CONFIG_FILE = /home/user1/BanBot.cfg
}

{
    SERVER_NAME = server2
    IP = 81.27.6.123
    PORT = 27961
    BOT_LOGFILE = /home/user2/BotLog_public.log
    BACKUP_DIR = /home/user2/backup/
    DATABASE_DIR = /home/user2/database/
    CONFIG_FILE = /home/user2/BanBot.cfg
}
```

Secondary configuration file: /home/user1/BanBot.cfg

```
{
    RCON_PASSWORD = yeah!
    GAME_LOGFILE = .q3a/q3ut4/games.log
    STRICT_LEVEL = 0
}
```

Secondary configuration file: /home/user2/BanBot.cfg

```
{
    RCON_PASSWORD = rconpass
    GAME_LOGFILE = .q3a/q3ut4/public.log
    STRICT_LEVEL = 1
}
```

With this method the user has his/her own configuration file where he/she can change some settings without seeing the others.

Take notice that all parameters must be set at all times. Incomplere settings or invalid ones will not be taken in consideration and ignored.

Another important consideration: do not insert twice the same parameter inside the same server configuration. In this case, the last parameter inserted will overwrite the one before. Plus the settings on secondary server configuration files are the last to be examined therefore overwriting the previous.

2.3.4 Intelligent configuration of the game server

Most of the times, even for tournament rules, you don't want to have the bot active when setting the server in "private" mode.
The bot surveys the server following the file indicated in it's settings. You can easily "deactivate" the bot changing the name of the log file in use for the server when in "private" mode.

To make an example, the config *public.cfg* sets the server as public. You could set the variable

set g_log "games.log"

whilst in the config *pcw.cfg* that sets the server as "private", you could set

set g_log "pcw.log"

This way the bot won't take action if the server is in "private" mode.

2.3.5 Launching the bot

After having configured correctl the bot, you're ready to launch it. Go to the folder containing the bot executable and digit (from a terminal):

screen

this will open a new terminal that *won't terminate* when we close our connection to the server. At this point, launch the bot:

./BanBot

Take a look to check that all went well (options loaded, no error messagges ecc) and in case correct those errors with folder and directory permissions like seen previously.

Good, the bot is running!

Now all you have to do is enter the game server and write the following in chat

!iamgod

to get control over the bot. The bot will answer with a *"Welcome, my master!"*. This operation is to be done (and is possible) only when making the bot survey a new server. If the bot is shut down and restarted, the admins will remain saved.

NOTICE: The bot must not interfere with the ongoing matches on the server. To do this, it has a "soft" startup. The bot recognizes players as the enter the game or change their info. So if you restart the bot in the middle of a match, the players in game will be taken notice of by the bot only if they change nick, team or on the loading of the next map. So it's perfectly normal if you can't slap, kick etc some players.

2.4 Security: strict level

The strict level is a parameter that lets you choose the level of security and restrictions that the bot will apply on the server.
Every level has it's own caratteristics. The higher the level, the more restrictive.

2.4.1 Strict level 0

Base level. No anticheat function is activated.
The bot simply listens for comands given by the administrators and keeps already banned player out. A ban via guid is permanent whilst a ban via nick or ip lasts only one hour. In case someone enters the server with a nick that has been banned for more than one hour, the bot will tell an admin in game about this.

2.4.2 Strict level 1

Same rules for ban as strict level 0 but for this level the base anticheat detection is active.
If the bot detects a cheat, it will ban automatically. In rare cases where the bot is not secure about a cheat, it will advise and admin about the possible cheater without taking action.
The admins will decide weather to ban or not.

2.4.3 Strict level 2

With this level you have the base anticheat checks and the bot is way more selective: all bans are without time limit
Also, when in doubt cases, the bot doesn't warn admins but bans and throws out directly the player.

A note regarding this last point: because of permanent ban, blameless people could be thrown out of the server because named the same way a banned user is. For all strict levels > *strict 1* the bot might make some innocent "victims" both for doubt cases of permanent bans. So think well about what level of security to use. All responsability is yours.

2.5 Bot commands

Let's take a look at the bots commands, at what they are used for and what they do. First of all you need to know that all commands are used through "say". That would be the public chat on the game. All bot responses are visible only to the admin that gave the command (except for the special command *!iamgod*). In the explanation of the commands, the symbols smaller and greater ("<" and ">") indicate an obbligatory parameter whilst the square brackets ("[" and "]") indicate an optional parameter.

2.5.1 !admins

Use:

> !admins

shows a list of players in game that are registered as admins to the bot

2.5.2 !ban

Use:

> !ban <nick/number> [motive of ban]

bans permanently a player in game.
This command accepts a number (the id of the player visible through the command "clientlist" on the game console) or in alternative a portion of the players nick (that must identify a unique player).
For example, if you want to ban this player (where his player number is 9) :

> 9 [AlFd]Prot

you can use the following commands:

```
1 !ban 9
2 !ban 9 don't want him around
3 !ban Prot
4 !ban [AlFd]
5 !ban ]prot get lost!
```

Commands 2 and 5 were done with the "motive" option. In case a player is named "Protect", command 3 won't work and the bot will answer with a "player not found or nick not unique". This because there are two players with "Prot" in their nicks. In these cases use another part of the nick or write the whole nick or use the player number.

2.5.3 !deop

Use:

> !deop <id>

take op status from the player identified by the bot with the id given. To see which id the bot has given to a determined admin use the command *!findop*.

2.5.4 !find

Use:

!find <nick>

searches for the given nick amongst the banned players and shows the results. This is used to search for the number associated to that player by the bot so that you can unban him/her.

A possible answer could be the following:

!find Z

```
1 Exact search: none.
2 Ricerca:
3 3 [2s2h]Zamy by [2s2h]n3m3s1s,
4 10 Zinko by [2s2h]Zamy test test test.
```

In the answer there are 4 columns: id, nick, admin that applied the ban and an eventual motive. The id is the number that you'll be needing for particular commands.
Notice that BanBot remember only the last nick used by the player.

2.5.5 !findop

Use:

!findop <nick>

searches the given nick amongst the admins and shows the result of the search. This is used to see what id the bot has given the selected admin. A possible answer might be the following:

!findop Z

```
1 Exact search: none.
2 Search:
3 3 [2s2h]Zamy,
4 10 Zenna.
```

In the answer there are two columns: id and nick. You'll need the id for particular commands.

2.5.6 !force

Use:

!force <red/blue/spect> <nick/number>

changes the team of the given player: you can move the player to the blu or red team. Or even make him/her spectate the match.

2.5.7 !help

Use:

!help

shows a list with the bots main commands and their use

2.5.8 !kick

Use:

!kick <nick/number>

kicks given player from server

2.5.9 !iamgod

use:

!iamgod

registers the player that gives the command as admin. Works only if there are no admins registered to the bot.

2.5.10 !map

Use:

!map <map name>

change the current map on the server. The exact name of the map must be inserted (ex: "!map ut4_casa").

2.5.11 !mute

Use:

!mute <nick/number/all/ALL>

mutes and unmutes the given player. If instead of the nick, the parameter *all* or *ALL* is given, the bot mutes everyone execpt the admin that gave the command.

2.5.12 !nextmap

Use:

!nextmap <map name>

changes the server's next map. You must indicate the complete name of the map (map filename).

2.5.13 !nuke

Use:

!nuke <nick/number>

nukes the given player. If matchmode is enabled, the nuke won't work.

2.5.14 !op

Use:

> !op <nick/number>

registers the given player to the bot as an admin.The player will be registered with the given nick so it's advised to use your official nick before giving this command

2.5.15 !slap

Use:

> !slap <nick/number> [2-5]

slaps the given player. Once if no optional parameter is given otherwise slaps up to 5 times.

NOTE: if you pass an optional slap count ¿ 5, the command will be ignored

2.5.16 !status

Use:

> !status

show version and the status of the bot (number of admins, banned player, strict level ecc)

2.5.17 !strict

Use:

> !strict <OFF/off/0/1/2>

changes strict level used by the bot on the server: *off* and *0* are equivalent

2.5.18 !unban

Use:

> !unban <id>

unbans player corrisponding to given id. To see which id to use in the command, use *!find*.

2.5.19 !veto

Use:

> !veto

cancels current vote.

2.6 Problem resolution and assistance

2.6.1 The bot is slow sometimes

Sometimes, especially when more than one serer is under control by the bot with a single instance, the bot might take a few seconds to respond to commands : this because the bot is not multi-thread yet and can execute only one command at a time. for example: the bot might still need to finish printing the *!help* on a server before answering your command. Same thing for two comands given on the same server. This will be solved in the next releases of BanBot.

2.6.2 The bot doesn't recognize players

As already said, the bot has a "soft" startup and recognizes players as they log into the server and send their information. If this causes problems just reload the map.

2.6.3 The bot doesn't respond to my commands

In this case there may be more than one cause: - check if the bot doesn't respond only to you or also to other admins if the bot doesn't respond only to you then for some reason you changed guid and you'll have to register to the bot again (make another admin do this for you) and then cancel your old registration. If the bot doesn't answer to any admin then there can be three other causes:

- the bot has craashed.
- the rcon password has changed.
- the game log file has been changed.

2.6.4 The bot doesn't respond to the command *!iamgod*

If it's not the first time you launch the bot and there are already some admins registered to it, then it is normal it doesn't respond to *!iamgod*. You have to use the command !op. In case of problems, for example a single admin registered that has changed guid, you can reset the bot by deleting its database.

In other cases, check you settigns: it's very probable that you got something wrong. Read the startup messagges to understand better where the problem is.

2.6.5 Assistance and bug reporting

For assistance and bug reporting, we've created a section on our website where you'll find a bugtracker. After having registered, you can open tickets to ask for help, report bugs or simply give us your opinion about the bot or new ideas for feature you'd like to see implemented. Use **ONLY** the bugtracker for **BUGS or FEATURES**. Any help/assistance questions via our forum or mail.