Lampiran A: Konfigurasi Squidguard

```
logdir
/home/geraldi/Desktop/Skripsi/Work/SquidGuard/squidGuar
d-1.5-beta/test
dbhome
/home/geraldi/Desktop/Skripsi/Work/SquidGuard/squidGuar
d-1.5-beta/test

src admins {
    ip 172.16.240.224/27
    # range 172.16.240.213 - 172.16.240.254
}

src students {
    ip 10.0.0.0/8
    # range 10.0.0.0 - 10.255.255.254
}

dest blacklist {
    domainlist testblacklist/domains
    urllist    testblacklist/urls
    redirect
http://info.foo.bar/BLOCKED?clientaddr=%a&clientname=%n
&clientuser=%i&clientgroup=%s&targetgroup=%t&url=%u
}

acl {
    admins {
        pass all
    }

    students {
        pass !blacklist none
        redirect
http://info.foo.bar/REDIRECTED?clientaddr=%a&clientname
=%n&clientuser=%i&clientgroup=%s&targetgroup=%t&url=%u
    }

    default {
        pass none
        redirect
http://info.foo.bar/BLOCKED?clientaddr=%a&clientname=%n
&clientuser=%i&clientgroup=%s&targetgroup=%t&url=%u
    }
}
```

Lampiran B: Bloom-m2.h

```
#include <fcntl.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

#ifndef _BLOOM_MURMURHASH2
#define _BLOOM_MURMURHASH2
unsigned int murmurhash2(const void * key, int len, const unsigned
int seed)
{
 // 'm' and 'r' are mixing constants generated offline.
 // They're not really 'magic', they just happen to work well.

 const unsigned int m = 0x5bd1e995;
 const int r = 24;

 // Initialize the hash to a 'random' value

 unsigned int h = seed ^ len;

 // Mix 4 bytes at a time into the hash

 const unsigned char * data = (const unsigned char *)key;

 while(len >= 4)
 {
  unsigned int k = *(unsigned int *)data;

  k *= m;
  k ^= k >> r;
  k *= m;

  h *= m;
  h ^= k;

  data += 4;
  len -= 4;
 }

 // Handle the last few bytes of the input array

 switch(len)
 {
 case 3: h ^= data[2] << 16;
 case 2: h ^= data[1] << 8;
 case 1: h ^= data[0];
         h *= m;
 };

 // Do a few final mixes of the hash to ensure the last few
 // bytes are well-incorporated.
```

```
 h ^= h >> 13;
 h *= m;
 h ^= h >> 15;

 return h;
}
#endif //_BLOOM_MURMURHASH2


#ifndef _BLOOM_H
#define _BLOOM_H

/**
************************************************************************
*********
 * Structure to keep track of one bloom filter.  Caller needs to
 * allocate this and pass it to the functions below. First call for
 * every struct must be to bloom_init().
 *
 */
struct bloom
{
  // These fields are part of the public interface of this
structure.
  // Client code may read these values if desired. Client code MUST
NOT
  // modify any of these.
  int entries;
  double error;
  int bits;
  int bytes;
  int hashes;

  // Fields below are private to the implementation. These may go
away or
  // change incompatibly at any moment. Client code MUST NOT access
or rely
  // on these.
  double bpe;
  unsigned char * bf;
  int ready;
};

static int bloom_check_add(struct bloom * bloom,
                           const void * buffer, int len, int add)
{
  if (bloom->ready == 0) {
    (void)printf("bloom at %p not initialized!\n", (void *)bloom);
    return -1;
  }

  int hits = 0;
  register unsigned int a = murmurhash2(buffer, len, 0x9747b28c);
  register unsigned int b = murmurhash2(buffer, len, a);
  register unsigned int x;
  register unsigned int i;
  register unsigned int byte;
  register unsigned int mask;
  register unsigned char c;
```

```
  for (i = 0; i < bloom->hashes; i++) {
    x = (a + i*b) % bloom->bits;
    byte = x >> 3;
    c = bloom->bf[byte];        // expensive memory access
    mask = 1 << (x % 8);

    if (c & mask) {
      hits++;
    } else {
      if (add) {
        bloom->bf[byte] = c | mask;
      }
    }
  }

  if (hits == bloom->hashes) {
    return 1;                   // 1 == element already in (or
collision)
  }

  return 0;
}

/**
****************************************************************
*********
 * Initialize the bloom filter for use.
 *
 * The filter is initialized with a bit field and number of hash
functions
 * according to the computations from the wikipedia entry:
 *     http://en.wikipedia.org/wiki/Bloom_filter
 *
 * Optimal number of bits is:
 *     bits = (entries * ln(error)) / ln(2)^2
 *
 * Optimal number of hash functions is:
 *     hashes = bpe * ln(2)
 *
 * Parameters:
 * -----------
 *     bloom   - Pointer to an allocated struct bloom (see above).
 *       entries - The expected number of entries which will be
inserted.
 *     error   - Probability of collision (as long as entries are
not
 *               exceeded).
 *
 * Return:
 * -------
 *     0 - on success
 *     1 - on failure
 *
 */
int bloom_init(struct bloom * bloom, int entries, double error)
{
  bloom->ready = 0;
```

```
  if (entries < 1 || error == 0) {
    return 1;
  }

  bloom->entries = entries;
  bloom->error = error;

  double num = log(bloom->error);
  double denom = 0.480453013918201; // ln(2)^2
  bloom->bpe = -(num / denom);

  double dentries = (double)entries;
  bloom->bits = (int)(dentries * bloom->bpe);

  if (bloom->bits % 8) {
    bloom->bytes = (bloom->bits / 8) + 1;
  } else {
    bloom->bytes = bloom->bits / 8;
  }

  bloom->hashes = (int)ceil(0.693147180559945 * bloom->bpe);   //
ln(2)

  bloom->bf = (unsigned char *)calloc(bloom->bytes, sizeof(unsigned
char));
  if (bloom->bf == NULL) {
    return 1;
  }

  bloom->ready = 1;
  return 0;
}


/**
*****************************************************************
*********
 * Check if the given element is in the bloom filter. Remember this
may
 * return false positive if a collision occured.
 *
 * Parameters:
 * -----------
 *     bloom  - Pointer to an allocated struct bloom (see above).
 *     buffer - Pointer to buffer containing element to check.
 *     len    - Size of 'buffer'.
 *
 * Return:
 * -------
 *     0 - element is not present
 *     1 - element is present (or false positive due to collision)
 *    -1 - bloom not initialized
 *
 */
int bloom_check(struct bloom * bloom, const void * buffer, int len)
{
  return bloom_check_add(bloom, buffer, len, 0);
}
```

```
/**
*****************************************************************
*********
 * Add the given element to the bloom filter.
 * The return code indicates if the element (or a collision) was
already in,
 * so for the common check+add use case, no need to call check
separately.
 *
 * Parameters:
 * -----------
 *     bloom  - Pointer to an allocated struct bloom (see above).
 *     buffer - Pointer to buffer containing element to add.
 *     len    - Size of 'buffer'.
 *
 * Return:
 * -------
 *     0 - element was not present and was added
 *     1 - element (or a collision) had already been added previously
 *    -1 - bloom not initialized
 *
 */
int bloom_add(struct bloom * bloom, const void * buffer, int len)
{
  return bloom_check_add(bloom, buffer, len, 1);
}


/**
*****************************************************************
*********
 * Print (to stdout) info about this bloom filter. Debugging aid.
 *
 */
void bloom_print(struct bloom * bloom, char *s)
{
  (void)sprintf(s,   "\nbloom   at   %p\n\t->entries\t=   %d\n\t-
>error\t\t= %f\n\t\
->bits\t\t= %d\n\t->bits per elem\t= %f\n\t->bytes\t\t= %d\n\t\
->hash functions= %d\n",
  (void *)bloom, bloom->entries, bloom->error,
  bloom->bits, bloom->bpe, bloom->bytes, bloom->hashes);
}


/**
*****************************************************************
*********
 * Deallocate internal storage.
 *
 * Upon return, the bloom struct is no longer usable. You may call
bloom_init
 * again on the same struct to reinitialize it again.
 *
 * Parameters:
 * -----------
 *     bloom  - Pointer to an allocated struct bloom (see above).
 *
```

```
 * Return: none
 *
 */
void bloom_free(struct bloom * bloom)
{
  if (bloom->ready) {
    free(bloom->bf);
  }
  bloom->ready = 0;
}
#endif
```

Lampiran C: main.c Squidguard awal

```
/*
  By accepting this notice, you agree to be bound by the following
  agreements:

  This software product, squidGuard, is copyrighted (C) 1998-2009
  by Christine Kronberg, Shalla Secure Services. All rights
reserved.

  This program is free software; you can redistribute it and/or
modify it
  under the terms of the GNU General Public License (version 2) as
  published by the Free Software Foundation.  It is distributed in
the
  hope that it will be useful, but WITHOUT ANY WARRANTY; without
even the
  implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
  PURPOSE.   See the GNU General Public License (GPL) for more
details.

  You should have received a copy of the GNU General Public License
  (GPL) along with this program.
*/

#include "sg.h"
#ifdef USE_SYSLOG
#include <syslog.h>
#endif

struct Setting *lastSetting = NULL;
struct Setting *Setting = NULL;                          /* linked
list, Calloc */

struct Source *lastSource = NULL;
struct Source *Source = NULL;                       /* linked list,
Calloc */

struct Destination *lastDest = NULL;
struct Destination *Dest = NULL;                    /* linked list,
Calloc */

struct sgRewrite *lastRewrite = NULL;
struct sgRewrite *Rewrite = NULL;                   /* linked list,
Calloc */
struct sgRegExp *lastRewriteRegExec = NULL;

struct Time *lastTime = NULL;
struct Time *Time = NULL;                           /* linked list,
Calloc */

struct LogFileStat *globalErrorLog = NULL;
struct LogFile *globalLogFile = NULL;

struct LogFileStat *lastLogFileStat;
struct LogFileStat *LogFileStat;                    /* linked list,
Calloc */

struct TimeElement *lastTimeElement = NULL;
```

```
struct TimeElement *TimeElement = NULL;

struct Acl *lastAcl = NULL;
struct Acl *defaultAcl = NULL;
struct Acl *Acl = NULL;                                    /* linked
list, Calloc */
struct AclDest *lastAclDest = NULL;

struct sgRegExp *lastRegExpDest;

struct Source *lastActiveSource;

char **globalArgv ;
char **globalEnvp ;
int globalDebugTimeDelta = 0;
int globalDebug = 0;
int globalPid = 0;
int globalUpdate = 0;
int passthrough = 0;
int showBar = 0;   /* Do not display the progress bar. */
char *globalCreateDb = NULL;
int failsafe_mode = 0;
int sig_hup = 0;
int sig_alrm = 0;
int sgtime = 0;
char *globalLogDir = NULL;
int globalSyslog = 0;


#if __STDC__
int main(int    argc,
         char   **argv,
  char   **envp)
#else
int main(argc, argv, envp)
     int argc;
     char *argv[];
     char *envp[];
#endif
{
  int ch;
  struct SquidInfo squidInfo;
  struct Source *src;
  struct Acl *acl;
  struct timeval start_time,ready_time,stop_time;
  char buf[MAX_BUF];
  char *redirect,tmp[MAX_BUF];
  char *configFile = NULL;
  time_t t;
#if HAVE_SIGACTION
  struct sigaction act;
#endif
  gettimeofday(&start_time, NULL);
  progname = argv[0];
  globalPid = getpid();
#ifdef USE_SYSLOG
openlog("squidGuard", LOG_PID | LOG_NDELAY | LOG_CONS, LOG_LOCAL2);
#endif
  while ((ch = getopt(argc, argv, "hbduPC:t:c:v")) != EOF)
```

```
    switch (ch) {
    case 'd':
       globalDebug = 1;
      break;
    case 'c':
      configFile = optarg;
      break;
    case 'b':
      showBar = 1;
    case 'C':
      globalCreateDb = optarg;
      break;
    case 'P':
      passthrough = 1;
      break;
    case 'u':
      globalUpdate = 1;
      break;
    case 'v':
      fprintf(stderr,        "SquidGuard:        %s        %s\n",
VERSION,db_version(NULL,NULL,NULL));
      exit(0);
      break;
    case 't':
      if((t = iso2sec(optarg)) == -1){
 fprintf(stderr,"-t   dateformat   error,   should   be   yyyy-mm-
ddTHH:MM:SS\n");
 exit(0);
      }
      if(t < 0){
 fprintf(stderr,"-t date have to after 1970-01-01T01:00:00\n");
 exit(0);
      }
      sgLogDebug("DEBUG: squidGuard emulating date %s", niso(t));
      globalDebugTimeDelta = t - start_time.tv_sec;
      start_time.tv_sec = start_time.tv_sec + globalDebugTimeDelta;
      break;
    case '?':
    case 'h':
    default:
      usage();
    }
  globalArgv = argv;
  globalEnvp = envp;
  sgSetGlobalErrorLogFile();
  sgReadConfig(configFile);
  sgSetGlobalErrorLogFile();
  sgLogNotice("INFO: squidGuard %s started (%d.%03d)",
      VERSION, start_time.tv_sec, start_time.tv_usec/1000);
  if(globalUpdate || globalCreateDb != NULL){
    sgLogNotice("INFO: db update done");
    gettimeofday(&stop_time, NULL);
    stop_time.tv_sec = stop_time.tv_sec + globalDebugTimeDelta;
    sgLogNotice("INFO:              squidGuard              stopped
(%d.%03d)",stop_time.tv_sec,stop_time.tv_usec/1000);
#ifdef USE_SYSLOG
    closelog ();
#endif
    exit(0);
```

```
  }
  sgTimeElementSortEvents();
  sgTimeNextEvent();
#if HAVE_SIGACTION
#ifndef SA_NODEFER
#define SA_NODEFER 0
#endif
  act.sa_handler = sgHandlerSigHUP;
  act.sa_flags = SA_NODEFER | SA_RESTART;
  sigaction(SIGHUP, &act, NULL);
#else
#if HAVE_SIGNAL
  signal(SIGHUP, sgHandlerSigHUP);
#else
#endif
#endif
  gettimeofday(&ready_time, NULL);
  ready_time.tv_sec = ready_time.tv_sec + globalDebugTimeDelta;
  sgLogNotice("INFO: squidGuard ready for requests (%d.%03d)",
      ready_time.tv_sec, ready_time.tv_usec/1000);
  tmp[MAX_BUF-1] = '\0';
  while(1) {
    while(fgets(buf, MAX_BUF, stdin) != NULL){
      if(sig_hup) {
 sgReloadConfig();
      }
      if(failsafe_mode) {
 puts("");
 fflush(stdout);
 if(sig_hup){
         sgReloadConfig();
 }
 continue;
      }
      if(parseLine(buf,&squidInfo) != 1){
 sgLogError("ERROR: Error parsing squid line: %s",buf);
 puts("");
      }
        else {
 src = Source;
 for(;;){
   strncpy(tmp,squidInfo.src,MAX_BUF-1);
         tmp[MAX_BUF-1] = 0;   /* force null termination */
   globalLogFile = NULL;
   src                  =                    sgFindSource(src,
tmp,squidInfo.ident,squidInfo.srcDomain);
   acl = sgAclCheckSource(src);
   if((redirect = sgAclAccess(src,acl,&squidInfo)) == NULL){
     if(src == NULL || src->cont_search == 0){
       puts("");
       break;
     } else
       if(src->next != NULL){
  src = src->next;
  continue;
       } else {
  puts("");
  break;
       }
```

```
    } else {
      if(squidInfo.srcDomain[0] == '\0'){
        squidInfo.srcDomain[0] = '-';
        squidInfo.srcDomain[1] = '\0';
      }
      if(squidInfo.ident[0] == '\0'){
        squidInfo.ident[0] = '-';
        squidInfo.ident[1] = '\0';
      }
      fprintf(stdout,"%s %s/%s %s %s\n",redirect,squidInfo.src,
       squidInfo.srcDomain,squidInfo.ident,
       squidInfo.method);
            /*      sgLogDebug("DEBUG:      %s      %s/%s      %s
%s\n",redirect,squidInfo.src,squidInfo.srcDomain,squidInfo.ident,s
quidInfo.method);  */
      break;
    }
  } /*for(;;)*/
       }
       fflush(stdout);
       if(sig_hup)
         sgReloadConfig();
     }
#if !HAVE_SIGACTION
#if HAVE_SIGNAL
     if(errno != EINTR){
       gettimeofday(&stop_time, NULL);
       stop_time.tv_sec = stop_time.tv_sec + globalDebugTimeDelta;
       sgLogNotice("INFO:            squidGuard            stopped
(%d.%03d)",stop_time.tv_sec,stop_time.tv_usec/1000);
#ifdef USE_SYSLOG
       closelog ();
#endif
       exit(2);
     }
#endif
#else
     gettimeofday(&stop_time, NULL);
     stop_time.tv_sec = stop_time.tv_sec + globalDebugTimeDelta;
     sgLogNotice("INFO:            squidGuard            stopped
(%d.%03d)",stop_time.tv_sec,stop_time.tv_usec/1000);
#ifdef USE_SYSLOG
     closelog ();
#endif
     exit(0);
#endif
  }
  exit(0);
}

#if __STDC__
void usage()
#else
void usage()
#endif
{
  fprintf(stderr,
   "Usage: squidGuard [-u] [-C block] [-t time] [-c file] [-v] [-
d] [-P]\n");
```

```
  fprintf(stderr, "Options:\n");
  fprintf(stderr, "  -v          : show version number\n");
  fprintf(stderr, "  -d          : all errors to stderr\n");
  fprintf(stderr, "  -b          : switch on the progress bar when
updating the blacklists\n");
  fprintf(stderr, "  -c file     : load alternate configfile\n");
  fprintf(stderr, "   -t time      : specify startup time in the
format: yyyy-mm-ddTHH:MM:SS\n");
  fprintf(stderr, "   -u           : update .db files from .diff
files\n");
  fprintf(stderr, "   -C file|all : create new .db files from
urls/domain files\n");
  fprintf(stderr, "                specified in \"file\".\n");
  fprintf(stderr, "   -P          : do not go into emergency mode
when an error with the \n");
  fprintf(stderr, "                blacklists is encountered.\n");

#ifdef USE_SYSLOG
  closelog ();
#endif
  exit(1);
}
```

Lampiran D: main.c Squidguard dengan Sisipan Bloom Filter

```
/*
  By accepting this notice, you agree to be bound by the following
  agreements:

  This software product, squidGuard, is copyrighted (C) 1998-2009
  by Christine Kronberg, Shalla Secure Services. All rights
reserved.

  This program is free software; you can redistribute it and/or
modify it
  under the terms of the GNU General Public License (version 2) as
  published by the Free Software Foundation.  It is distributed in
the
  hope that it will be useful, but WITHOUT ANY WARRANTY; without
even the
  implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR
  PURPOSE.   See the GNU General Public License (GPL) for more
details.

  You should have received a copy of the GNU General Public License
  (GPL) along with this program.
*/

#include "bloom-m2.h"
#include "sg.h"
#ifdef USE_SYSLOG
#include <syslog.h>
#endif

struct Setting *lastSetting = NULL;
struct Setting *Setting = NULL;   /* linked list, Calloc */

struct Source *lastSource = NULL;
struct Source *Source = NULL;    /* linked list, Calloc */

struct Destination *lastDest = NULL;
struct Destination *Dest = NULL;  /* linked list, Calloc */

struct sgRewrite *lastRewrite = NULL;
struct sgRewrite *Rewrite = NULL;  /* linked list, Calloc */
struct sgRegExp *lastRewriteRegExec = NULL;

struct Time *lastTime = NULL;
struct Time *Time = NULL;     /* linked list, Calloc */

struct LogFileStat *globalErrorLog = NULL;
struct LogFile *globalLogFile = NULL;

struct LogFileStat *lastLogFileStat;
struct LogFileStat *LogFileStat;  /* linked list, Calloc */

struct TimeElement *lastTimeElement = NULL;
struct TimeElement *TimeElement = NULL;

struct Acl *lastAcl = NULL;
struct Acl *defaultAcl = NULL;
struct Acl *Acl = NULL;       /* linked list, Calloc */
```

```
struct AclDest *lastAclDest = NULL;

struct sgRegExp *lastRegExpDest;

struct Source *lastActiveSource;

char **globalArgv ;
char **globalEnvp ;
int globalDebugTimeDelta = 0;
int globalDebug = 0;
int globalPid = 0;
int globalUpdate = 0;
int passthrough = 0;
int showBar = 0;   /* Do not display the progress bar. */
char *globalCreateDb = NULL;
int failsafe_mode = 0;
int sig_hup = 0;
int sig_alrm = 0;
int sgtime = 0;
char *globalLogDir = NULL;
int globalSyslog = 0;


#if __STDC__
int main(int    argc,
         char   **argv,
  char   **envp)
#else
int main(argc, argv, envp)
     int argc;
     char *argv[];
     char *envp[];
#endif
{
 int ch;
 struct SquidInfo squidInfo;
 struct Source *src;
 struct Acl *acl;
 struct timeval start_time,ready_time,stop_time;
 char buf[MAX_BUF];
 char *redirect,tmp[MAX_BUF];
 char *configFile = NULL;
 time_t t;
 #if HAVE_SIGACTION
   struct sigaction act;
 #endif
  gettimeofday(&start_time, NULL);
  progname = argv[0];
  globalPid = getpid();

 #ifdef USE_SYSLOG
  openlog("squidGuard",   LOG_PID   |   LOG_NDELAY   |   LOG_CONS,
LOG_LOCAL2);
 #endif

  while ((ch = getopt(argc, argv, "hbduPC:t:c:v")) != EOF) {
    switch (ch) {
     case 'd':
      globalDebug = 1;
```

```
        break;
    case 'c':
          configFile = optarg;
        break;
        case 'b':
         showBar = 1;
        case 'C':
         globalCreateDb = optarg;
        break;
        case 'P':
         passthrough = 1;
        break;
    case 'u':
     globalUpdate = 1;
        break;
        case 'v':
         fprintf(stderr,       "SquidGuard:       %s       %s\n",
VERSION,db_version(NULL,NULL,NULL));
         exit(0);
        break;
        case 't':
         if((t = iso2sec(optarg)) == -1){
      fprintf(stderr,"-t  dateformat  error,  should  be  yyyy-mm-
ddTHH:MM:SS\n");
      exit(0);
          }
          if(t < 0){
      fprintf(stderr,"-t    date    have    to    after    1970-01-
01T01:00:00\n");
      exit(0);
          }
          sgLogDebug("DEBUG:   squidGuard   emulating   date   %s",
niso(t));
          globalDebugTimeDelta = t - start_time.tv_sec;
          start_time.tv_sec       =       start_time.tv_sec       +
globalDebugTimeDelta;
        break;
      case '?':
      case 'h':
      default:
          usage();
     }
    }

  globalArgv = argv;
  globalEnvp = envp;

  sgSetGlobalErrorLogFile();
  sgReadConfig(configFile);
  sgSetGlobalErrorLogFile();

  sgLogNotice("INFO: squidGuard %s started (%d.%06d)",
    VERSION, start_time.tv_sec, start_time.tv_usec);

  if(globalUpdate || globalCreateDb != NULL){
    sgLogNotice("INFO: db update done");
    gettimeofday(&stop_time, NULL);
    stop_time.tv_sec = stop_time.tv_sec + globalDebugTimeDelta;
    sgLogNotice("INFO: squidGuard stopped (%d.%06d)",
```

```
    stop_time.tv_sec,stop_time.tv_usec);
 #ifdef USE_SYSLOG
    closelog ();
 #endif
   exit(0);
 }

 sgTimeElementSortEvents();
 sgTimeNextEvent();

#if HAVE_SIGACTION
 #ifndef SA_NODEFER
  #define SA_NODEFER 0
 #endif
  act.sa_handler = sgHandlerSigHUP;
  act.sa_flags = SA_NODEFER | SA_RESTART;
  sigaction(SIGHUP, &act, NULL);
#else
 #if HAVE_SIGNAL
   signal(SIGHUP, sgHandlerSigHUP);
 #else
 #endif
#endif

 gettimeofday(&ready_time, NULL);
 ready_time.tv_sec = ready_time.tv_sec + globalDebugTimeDelta;
 sgLogNotice("INFO: squidGuard ready for requests (%d.%06d)",
   ready_time.tv_sec, ready_time.tv_usec);

 tmp[MAX_BUF-1] = '\0';

 /* Instantiate Bloom Filter */
 struct bloom sgBloom;

 int maxFilterEntries = 1000000;
 double filterFPR = .001;    //desired false positive rate

 if (bloom_init(&sgBloom, maxFilterEntries, filterFPR) == 1) {
   fprintf(stderr,"Filter initiation failed!\n");
 exit(0);
 }

 sgLogNotice("Gerald - Filter instantiation done!");
 char bloomInfo[MAX_BUF-1];
 bloom_print(&sgBloom, bloomInfo);
 sgLogNotice(bloomInfo);
 /* Instantiate Filter - END */

 while(1) {
   while(fgets(buf, MAX_BUF, stdin) != NULL){
       if(sig_hup) {
   sgReloadConfig();
       }
       if(failsafe_mode) {
   puts("");
   fflush(stdout);
   if(sig_hup){
            sgReloadConfig();
   }
```

```
      continue;
      }

          if(parseLine(buf,&squidInfo) != 1){
      sgLogError("ERROR: Error parsing squid line: %s",buf);
      puts("");
          } else {
      src = Source;
      for(;;){
        strncpy(tmp,squidInfo.src,MAX_BUF-1);
              tmp[MAX_BUF-1] = 0;   /* force null termination */
        globalLogFile = NULL;
        src                    =                    sgFindSource(src,
tmp,squidInfo.ident,squidInfo.srcDomain);
        acl = sgAclCheckSource(src);
        if((redirect = sgAclAccess(src,acl,&squidInfo)) == NULL){
          if(src == NULL || src->cont_search == 0){
           //fprintf(stdout, "puts because (src == NULL || src-
>cont_search == 0)\n");
              puts("");
              break;
          } else if(src->next != NULL){
         src = src->next;
         continue;
             } else {
              //fprintf(stdout, "puts because 'else{}'\n");
         puts("");
         break;
             }
        } else {
          if(squidInfo.srcDomain[0] == '\0'){
              squidInfo.srcDomain[0] = '-';
              squidInfo.srcDomain[1] = '\0';
          }
          if(squidInfo.ident[0] == '\0'){
              squidInfo.ident[0] = '-';
              squidInfo.ident[1] = '\0';
          }

          int tcheck = 0, tadd = 0;
          char *isbl = NULL;

          //fprintf(stdout, "------------\n");

          /* Is redirection a blacklist? */
          isbl = strstr(redirect,"targetgroup=blacklist");
          if (isbl == NULL) {
           //fprintf(stdout, "Request is not a blacklist. ");
           //fprintf(stdout, "Checking source IP %s len:%d...\n",
squidInfo.src, strlen(squidInfo.src));

        /* Is source already in filter? */
        tcheck     =     bloom_check(&sgBloom,     squidInfo.src,
strlen(squidInfo.src));
        if (tcheck == 1) {
         //fprintf(stdout, "Source IP MAY be in the filter. Giving
request a pass...\n");
          puts(""); /* Pass request */
          break;
```

```
        } else if (tcheck == 0) {
         /* Add source to filter */
         //fprintf(stdout, "Source IP is not in the filter. Adding
source ip to filter...\n");

         tadd      =      bloom_add(&sgBloom,      squidInfo.src,
strlen(squidInfo.src));
         if (tadd == 0) {
          //fprintf(stdout,   "Adding   successful.   Redirecting
request...\n");
         } else if (tadd == 1) {
          //fprintf(stdout, "Adding canceled due to collision.
Redirecting request...\n");
         } else {
          //fprintf(stdout/*stderr*/,"Something went wrong with
the filter (on adding)...\n");
           exit(0);
         }
         /* Add source to filter - END */
        } else {
           fprintf(stderr,"Something went wrong with the filter
(on checking)...\n");
         exit(0);
        }
        /* Is source already in filter? - END */
       } else {
        //fprintf(stdout,  "Request  is  a  blacklist.  Blocking
request...\n");
       }
       /* Is redirection a blacklist? - END */

         fprintf(stdout,"%s %s/%s %s %s\n",redirect,squidInfo.src,
        squidInfo.srcDomain,squidInfo.ident, squidInfo.method);
         /* Redirect request */
              /*     sgLogDebug("DEBUG:     %s     %s/%s     %s
%s\n",redirect,squidInfo.src,

 squidInfo.srcDomain,squidInfo.ident,squidInfo.method);  */
         break;
       }
    } /*for(;;)*/
       }
       fflush(stdout);

       if(sig_hup)
        sgReloadConfig();
    }

  /* Destroy bloom filter */
  bloom_free(&sgBloom);
  sgLogNotice("Gerald - Filter freed from memory");
  /* Destroy bloom filter - END*/

  #if !HAVE_SIGACTION
   #if HAVE_SIGNAL
       if(errno != EINTR){
          gettimeofday(&stop_time, NULL);
          stop_time.tv_sec       =       stop_time.tv_sec       +
globalDebugTimeDelta;
```

```
            sgLogNotice("INFO: squidGuard stopped (%d.%06d)",
             stop_time.tv_sec,stop_time.tv_usec);
            /* To help count runtime */
            fprintf(stdout,  "ready_  time  (%d.%06d),  stop_time
(%d.%06d)",
             (int)ready_time.tv_sec, (int)ready_time.tv_usec,
             (int)stop_time.tv_sec, (int)stop_time.tv_usec);
            /* To help count runtime - END */
      #ifdef USE_SYSLOG
             closelog ();
      #endif
            exit(2);
        }
    #endif
   #else
      gettimeofday(&stop_time, NULL);
      stop_time.tv_sec = stop_time.tv_sec + globalDebugTimeDelta;
      sgLogNotice("INFO:            squidGuard            stopped
(%d.%06d)",stop_time.tv_sec,stop_time.tv_usec);
    /* To help count runtime */
        fprintf(stdout,  "ready_  time  (%d.%06d),  stop_time
(%d.%06d)",
         (int)ready_time.tv_sec, (int)ready_time.tv_usec,
         (int)stop_time.tv_sec, (int)stop_time.tv_usec);
    /* To help count runtime - END */
    #ifdef USE_SYSLOG
       closelog ();
    #endif
      exit(0);
   #endif
   }

   exit(0);
}

#if __STDC__
void usage()
#else
void usage()
#endif
{
 fprintf(stderr,
  "Usage: squidGuard [-u] [-C block] [-t time] [-c file] [-v] [-d]
[-P]\n");
   fprintf(stderr, "Options:\n");
   fprintf(stderr, "  -v          : show version number\n");
   fprintf(stderr, "  -d          : all errors to stderr\n");
   fprintf(stderr, "  -b          : switch on the progress bar when
updating the blacklists\n");
   fprintf(stderr, "  -c file     : load alternate configfile\n");
   fprintf(stderr, "  -t time      : specify startup time in the
format: yyyy-mm-ddTHH:MM:SS\n");
   fprintf(stderr, "  -u           : update .db files from .diff
files\n");
   fprintf(stderr, "  -C file|all  : create new .db files from
urls/domain files\n");
   fprintf(stderr, "               specified in \"file\".\n");
   fprintf(stderr, "  -P          : do not go into emergency mode
when an error with the blacklists is encountered.\n");
```

```
  fprintf(stderr, "");

 #ifdef USE_SYSLOG
   closelog ();
 #endif

  exit(1);
}
```

Lampiran E: Makefile Squidguard

```
SHELL=/bin/sh
.SUFFIXES:
.SUFFIXES: .c .o .pl .pm .pod .html .man


CC = gcc
CPP = gcc -E
LEX = flex
PERL = /usr/bin/perl
YACC = bison -y
PERL = /usr/bin/perl
INSTALL= /usr/bin/install -c
INSTALL_DATA = ${INSTALL} -m 644
INSTALL_PROGRAM = ${INSTALL}
MKDIR = ../mkinstalldirs

RM = rm -f

CFLAGS = -g -O2 -I/usr/local/BerkeleyDB.4.8/include -I/usr/include
CPPFLAGS=  -I/usr/local/BerkeleyDB.4.8/include -I/usr/include
LDFLAGS=  -L/usr/local/BerkeleyDB.4.8/lib -L/usr/lib
LIBS = -lpthread -ldb -lm
DEFS = -DHAVE_CONFIG_H

INCLUDES= -I.. -I. -I$(srcdir)

COMPILE = $(CC) $(INCLUDES) $(CPPFLAGS) $(DEFS) $(CFLAGS)
LINK    = $(CC) $(LDFLAGS) -o $@

top_srcdir = ..
srcdir     = .


prefix = /usr/local
exec_prefix = ${prefix}
bindir = $(exec_prefix)/bin
logdir = /usr/local/squidGuard/log
cfgdir = /usr/local/squidGuard
infodir= $(prefix)/info

OBJS = main.o sgLog.o sgDb.o HTParse.o sgDiv.o sgFree.o y.tab.o
lex.yy.o

all::
 @echo making $@ in `basename \`pwd\``

all:: squidGuard
 @echo making $? in `basename \`pwd\``

squidGuard: $(OBJS)
 $(LINK) $(OBJS) $(LIBS)

conf: y.tab.o lex.yy.o
 $(COMPILE) -o conf y.tab.o lex.yy.o

main.o:main.c sg.h
 $(COMPILE) -c main.c
```

```
sgLog.o: sgLog.c sg.h
 $(COMPILE) -c sgLog.c

HTParse.o: HTParse.c wwwsys.h HTEscape.h
 $(COMPILE) -c HTParse.c

sgDiv.o: sgDiv.c sg.h sgEx.h
 $(COMPILE) -c sgDiv.c

sgFree.o: sgFree.c sg.h sgEx.h
 $(COMPILE) -c sgFree.c

sgDb.o: sgDb.c sg.h
 $(COMPILE) -c sgDb.c

lex.yy.o: lex.yy.c y.tab.h sg.h
 $(COMPILE) -c lex.yy.c

y.tab.o: y.tab.c y.tab.h sg.h sgEx.h
 $(COMPILE) -c y.tab.c

lex.yy.c: sg.l sg.h
 @if [ $(LEX) != ":" ]; then \
 $(LEX) sg.l  ; \
 else \
 echo " " ; \
 echo "No flex/lex found. Copy lex.yy.c.flex to lex.yy.c. " ; \
 echo " " ; \
 cp lex.yy.c.flex lex.yy.c ; \
 fi ;

lex.yy.c.flex: sg.l sg.h
 $(LEX) sg.l
 mv -f lex.yy.c lex.yy.c.flex

y.tab.c y.tab.h: sg.y sg.h
 @if [ "$(YACC)" = "yacc" ]; then \
    if [ ! -x $(YACC) ]; then \
       echo " " ; \
       echo "No yacc/bison found. Copy prepared files for y.tab.h
and y.tab.c over. " ; \
       echo " " ; \
       cp y.tab.h.bison y.tab.h ; \
       cp y.tab.c.bison y.tab.c ; \
    else \
       $(YACC) -d sg.y  ; \
    fi ; \
 else \
    $(YACC) -d sg.y  ; \
 fi ;

y.tab.c.bison y.tab.h.bison: sg.y sg.h
 $(YACC) -d sg.y
 mv -f y.tab.c y.tab.c.bison
 mv -f y.tab.h y.tab.h.bison

#
# Dependencies for installing
```

```
#

install::  install.bin
 @echo making $@ in `basename \`pwd\``

uninstall::  uninstall.bin
 @echo making $@ in `basename \`pwd\``

install.bin:: squidGuard
 @echo making $@ in `basename \`pwd\``
 @$(MKDIR) $(bindir) $(logdir) $(cfgdir)
 $(INSTALL_PROGRAM) squidGuard $(bindir)/squidGuard

uninstall.bin::
 @echo making $@ in `basename \`pwd\``
 $(RM) $(bindir)/squidGuard

update::
 @echo making $@ in `basename \`pwd\``

update::lex.yy.c.flex y.tab.c.bison y.tab.h.bison

#
# Dependencies for cleanup
#

clean::
 @echo making $@ in `basename \`pwd\``
 $(RM) *~ *.bak core *.log *.error
 $(RM) *.o y.tab.c y.tab.h squidGuard lex.yy.c

realclean:: clean
 @echo making $@ in `basename \`pwd\``
 $(RM) TAGS *.orig

distclean:: realclean
 @echo making $@ in `basename \`pwd\``
 $(RM) Makefile sg.h config.h

#
# Dependencies for maintenance
#


subdir = src

Makefile: Makefile.in ../config.status
 cd .. && CONFIG_FILES=$(subdir)/$@ CONFIG_HEADERS= ./config.status
```

Lampiran F: Data Uji Kecepatan Proses

Keterangan	Waktu Putaran (mikrodetik)		
	CRC32	Murmurhash2	Murmurhash3
Putaran 1	3.160.811	1.779.115	2.762.830
Putaran 2	3.130.725	1.767.018	2.769.030
Putaran 3	3.129.819	1.762.732	2.758.662
Putaran 4	3.131.941	1.754.967	2.771.500
Putaran 5	3.124.198	1.761.208	2.746.348
Putaran 6	3.127.765	1.757.558	2.749.060
Putaran 7	3.127.318	1.756.720	2.754.883
Putaran 8	3.131.590	1.758.269	2.758.398
Putaran 9	3.132.165	1.761.353	2.754.940
Putaran 10	3.124.604	1.761.321	2.755.889
Putaran 11	3.128.684	1.760.471	2.777.538
Putaran 12	3.124.984	1.762.288	2.774.104
Putaran 13	3.126.874	1.756.314	2.740.848
Putaran 14	3.125.950	1.758.259	2.751.877
Putaran 15	3.126.205	1.757.094	2.728.352
Putaran 16	3.122.771	1.760.365	2.751.777
Putaran 17	3.127.966	1.761.012	2.729.745
Putaran 18	3.134.104	1.759.646	2.768.305
Putaran 19	3.143.945	1.754.950	2.749.231
Putaran 20	3.129.917	1.727.695	2.747.970
Putaran 21	3.127.563	1.740.871	2.751.167
Putaran 22	3.122.461	1.749.179	2.749.608
Putaran 23	3.125.518	1.739.923	2.750.826
Putaran 24	3.121.866	1.743.952	2.752.310
Putaran 25	3.126.272	1.741.369	2.752.233
Putaran 26	3.127.307	1.735.221	2.765.516
Putaran 27	3.128.611	1.738.938	2.764.571
Putaran 28	3.124.791	1.740.860	2.753.928
Putaran 29	3.126.906	1.742.369	2.751.051
Putaran 30	3.127.225	1.742.697	2.770.363
Putaran 31	3.129.002	1.737.916	2.750.576
Putaran 32	3.130.603	1.742.342	2.751.313
Putaran 33	3.128.306	1.743.718	2.769.410
Putaran 34	3.134.921	1.741.295	2.749.667
Putaran 35	3.127.017	1.739.758	2.752.272
Putaran 36	3.125.201	1.750.216	2.751.776
Putaran 37	3.121.896	1.741.274	2.759.820
Putaran 38	3.140.996	1.739.392	2.748.778

Putaran 39	3.125.304	1.744.472	2.750.946
Putaran 40	3.124.313	1.740.662	2.751.381
Putaran 41	3.130.443	1.743.297	2.748.403
Putaran 42	3.123.721	1.741.545	2.749.989
Putaran 43	3.126.756	1.744.373	2.754.473
Putaran 44	3.123.918	1.736.429	2.750.364
Putaran 45	3.129.538	1.742.829	2.749.781
Putaran 46	3.122.350	1.741.620	2.767.411
Putaran 47	3.125.669	1.737.040	2.764.796
Putaran 48	3.120.558	1.738.758	2.753.246
Putaran 49	3.125.688	1.743.984	2.751.219
Putaran 50	3.115.971	1.741.916	2.748.639
Putaran 51	3.123.561	1.746.558	2.747.308
Putaran 52	3.117.588	1.746.153	2.749.882
Putaran 53	3.125.208	1.793.587	2.762.955
Putaran 54	3.124.785	1.778.872	2.738.877
Putaran 55	3.125.884	1.742.878	2.747.981
Putaran 56	3.117.843	1.744.274	2.748.262
Putaran 57	3.143.439	1.743.565	2.750.750
Putaran 58	3.126.770	1.743.945	2.753.257
Putaran 59	3.123.705	1.742.224	2.751.296
Putaran 60	3.122.301	1.743.618	2.745.918
Putaran 61	3.121.491	1.739.310	2.751.761
Putaran 62	3.125.960	1.738.103	2.750.984
Putaran 63	3.126.316	1.743.860	2.753.378
Putaran 64	3.123.033	1.740.377	2.768.256
Putaran 65	3.124.939	1.743.018	2.750.285
Putaran 66	3.125.571	1.739.130	2.748.810
Putaran 67	3.123.552	1.743.084	2.750.869
Putaran 68	3.126.777	1.737.946	2.748.023
Putaran 69	3.125.673	1.716.931	2.744.467
Putaran 70	3.130.071	1.742.633	2.751.033
Putaran 71	3.124.804	1.742.198	2.747.751
Putaran 72	3.126.679	1.743.658	2.750.632
Putaran 73	3.123.357	1.743.973	2.749.902
Putaran 74	3.118.808	1.743.394	2.750.827
Putaran 75	3.125.289	1.742.832	2.756.558
Putaran 76	3.118.695	1.744.287	2.757.344
Putaran 77	3.129.354	1.743.682	2.752.100
Putaran 78	3.122.165	1.743.455	2.752.884
Putaran 79	3.125.910	1.743.652	2.750.337
Putaran 80	3.120.345	1.743.477	2.753.407

Putaran 81	3.122.199	1.740.750	2.752.303
Putaran 82	3.124.608	1.739.053	2.753.467
Putaran 83	3.125.147	1.742.528	2.753.308
Putaran 84	3.115.239	1.738.467	2.755.015
Putaran 85	3.129.318	1.743.746	2.752.619
Putaran 86	3.123.405	1.742.148	2.752.581
Putaran 87	3.131.412	1.755.373	2.752.120
Putaran 88	3.119.843	1.784.369	2.754.233
Putaran 89	3.123.264	1.744.676	2.753.033
Putaran 90	3.125.258	1.718.535	2.754.643
Putaran 91	3.129.209	1.749.143	2.751.290
Putaran 92	3.124.182	1.747.982	2.754.019
Putaran 93	3.124.272	1.748.546	2.753.505
Putaran 94	3.126.169	1.748.420	2.753.609
Putaran 95	3.161.560	1.746.539	2.755.991
Putaran 96	3.126.613	1.748.215	2.751.232
Putaran 97	3.126.136	1.749.635	2.770.057
Putaran 98	3.121.826	1.749.152	2.753.478
Putaran 99	3.121.420	1.748.638	2.750.296
Putaran 100	3.123.894	1.747.584	2.745.749
Rerata	3.126.679	1.747.047	2.753.458
Maksimum	3.161.560	1.793.587	2.777.538
Minimum	3.115.239	1.716.931	2.728.352
Std. Deviasi	6.769	11.308	7.817

Lampiran G: Data Uji Galat Isbat

CRC32 Keterangan	Jml Galat Isbat J	N	P = (J/N)	p
Putaran 1	962	1000000	0,0962 %	0,1 %
Putaran 2	1009	1000000	0,1009 %	0,1 %
Putaran 3	1012	1000000	0,1012 %	0,1 %
Putaran 4	989	1000000	0,0989 %	0,1 %
Putaran 5	1031	1000000	0,1031 %	0,1 %
Putaran 6	1035	1000000	0,1035 %	0,1 %
Putaran 7	1051	1000000	0,1051 %	0,1 %
Putaran 8	979	1000000	0,0979 %	0,1 %
Putaran 9	971	1000000	0,0971 %	0,1 %
Putaran 10	988	1000000	0,0988 %	0,1 %
Putaran 11	984	1000000	0,0984 %	0,1 %
Putaran 12	964	1000000	0,0964 %	0,1 %
Putaran 13	1016	1000000	0,1016 %	0,1 %
Putaran 14	980	1000000	0,0980 %	0,1 %
Putaran 15	1002	1000000	0,1002 %	0,1 %
Putaran 16	961	1000000	0,0961 %	0,1 %
Putaran 17	960	1000000	0,0960 %	0,1 %
Putaran 18	946	1000000	0,0946 %	0,1 %
Putaran 19	924	1000000	0,0924 %	0,1 %
Putaran 20	954	1000000	0,0954 %	0,1 %
Rerata	985,9	-	-	-
Maksimum	1051	-	-	-
Minimum	924	-	-	-
Std. Deviasi	31,75516	-	-	-

$$\Delta\bar{P} = \frac{|p - \bar{P}|}{p} \times 100\% = \mathbf{1,41}\%$$

Murmurhash2 Keterangan	Jml Galat Isbat J	N	P = (J/N)	p
Putaran 1	1020	1000000	0,1020 %	0,1 %
Putaran 2	943	1000000	0,0943 %	0,1 %
Putaran 3	1054	1000000	0,1054 %	0,1 %
Putaran 4	1012	1000000	0,1012 %	0,1 %
Putaran 5	1031	1000000	0,1031 %	0,1 %
Putaran 6	994	1000000	0,0994 %	0,1 %
Putaran 7	1001	1000000	0,1001 %	0,1 %
Putaran 8	1054	1000000	0,1054 %	0,1 %
Putaran 9	1012	1000000	0,1012 %	0,1 %
Putaran 10	1014	1000000	0,1014 %	0,1 %
Putaran 11	1024	1000000	0,1024 %	0,1 %
Putaran 12	991	1000000	0,0991 %	0,1 %
Putaran 13	974	1000000	0,0974 %	0,1 %
Putaran 14	1042	1000000	0,1042 %	0,1 %
Putaran 15	950	1000000	0,0950 %	0,1 %
Putaran 16	1036	1000000	0,1036 %	0,1 %
Putaran 17	976	1000000	0,0976 %	0,1 %
Putaran 18	1031	1000000	0,1031 %	0,1 %
Putaran 19	1015	1000000	0,1015 %	0,1 %
Putaran 20	1033	1000000	0,1033 %	0,1 %
Rerata	1010,35	-	-	-
Maksimum	1054	-	-	-
Minimum	943	-	-	-
Std. Deviasi	30,46683	-	-	-

$$\Delta\bar{P} = \frac{|p - \bar{P}|}{p} \times 100\% = \mathbf{1,035}\%$$

Murmurhash3 Keterangan	Jml Galat Isbat J	N	P = (J/N)	p
Putaran 1	996	1000000	0,0996 %	0,1 %
Putaran 2	976	1000000	0,0976 %	0,1 %
Putaran 3	989	1000000	0,0989 %	0,1 %
Putaran 4	1012	1000000	0,1012 %	0,1 %
Putaran 5	1008	1000000	0,1008 %	0,1 %
Putaran 6	964	1000000	0,0964 %	0,1 %
Putaran 7	963	1000000	0,0963 %	0,1 %
Putaran 8	1003	1000000	0,1003 %	0,1 %
Putaran 9	1042	1000000	0,1042 %	0,1 %
Putaran 10	979	1000000	0,0979 %	0,1 %
Putaran 11	975	1000000	0,0975 %	0,1 %
Putaran 12	1022	1000000	0,1022 %	0,1 %
Putaran 13	991	1000000	0,0991 %	0,1 %
Putaran 14	994	1000000	0,0994 %	0,1 %
Putaran 15	1029	1000000	0,1029 %	0,1 %
Putaran 16	1009	1000000	0,1009 %	0,1 %
Putaran 17	974	1000000	0,0974 %	0,1 %
Putaran 18	1012	1000000	0,1012 %	0,1 %
Putaran 19	973	1000000	0,0973 %	0,1 %
Putaran 20	1047	1000000	0,1047 %	0,1 %
Rerata	997,9	-	-	-
Maksimum	1047	-	-	-
Minimum	963	-	-	-
Std. Deviasi	24,10166	-	-	-

$$\Delta\bar{P} = \frac{|p - \bar{P}|}{p} \times 100\% = \mathbf{0,21}\%$$