ETPro offers a 'certified configs' feature which basically means "this server is running the unmodified version of the config file leagues uploaded to their site". It's used for stopping people from modifying limitations and other stuff.

Here's a small command line tool you can use for certifying official configs with your modifications. To compile it, you will need PolarSSL's sha1 library.

```
/* Implementation of the ETPro's config certification system */
/* chaplja.blogspot.com | chaplja@gmail.com */

#define _CRT_SECURE_NO_WARNINGS /* microsoft's deprecated function warnings */

#include
#include
#include
#include "sha1.h" /* polarssl.org */

/* helper funcs */
void *read_text_file(const char *);

unsigned char etpro_sig[1024] = {
  0x18, 0x73, 0xE1, 0x02, 0xCA, 0x8A, 0xDD, 0x08, 0xBA, 0xBC, 0x1D, 0xC6,
  0xE9, 0x8C, 0xCD, 0x7B, 0x54, 0x2A, 0xAC, 0x95, 0xAE, 0xDE, 0xB3, 0xE2,
  0xA7, 0x76, 0x94, 0x34, 0xB0, 0xEC, 0xAB, 0x08, 0x9A, 0xFD, 0x41, 0xAF,
  0x6F, 0x4F, 0x03, 0xF5, 0x1F, 0x86, 0x95, 0x97, 0xEE, 0xBB, 0x2D, 0x03,
  0x00, 0x93, 0xA0, 0xA3, 0x67, 0x00, 0x17, 0x4B, 0xC3, 0x56, 0x82, 0xD8,
  0xE8, 0xBB, 0x40, 0xFA, 0x9C, 0x4E, 0x17, 0x91, 0xE5, 0xAF, 0x0D, 0x25,
  0xCD, 0x25, 0x44, 0xE8, 0xA8, 0x5C, 0x3A, 0x15, 0x8A, 0x47, 0x7E, 0x36,
  0xA6, 0x2C, 0x70, 0x9B, 0xFD, 0x1E, 0x75, 0x0B, 0xEA, 0xDC, 0x7F, 0x14,
  0x70, 0x19, 0x09, 0x97, 0xC7, 0x7F, 0x29, 0xC4, 0xA0, 0x79, 0x46, 0x4D,
  0x42, 0x52, 0xBC, 0xAE, 0xD7, 0x42, 0xCF, 0xC5, 0x79, 0xF5, 0x6D, 0x01,
  0xCB, 0x34, 0x8D, 0x5B, 0x8C, 0x15, 0xE4, 0xCE, 0x71, 0x41, 0x94, 0xAC,
```

0x1D, 0x79, 0xBB, 0xFF, 0x32, 0x89, 0xC2, 0xFB, 0x14, 0xBD, 0xC4, 0x17,
0x4B, 0x1C, 0x5E, 0x1E, 0x5C, 0x19, 0x3F, 0xD3, 0xF8, 0xE3, 0x8A, 0xB5,
0x9F, 0xA2, 0xB6, 0x7C, 0x2E, 0x1C, 0x02, 0x3C, 0x04, 0x52, 0x33, 0x75,
0x0B, 0x7E, 0x1D, 0x5B, 0x66, 0x28, 0x31, 0xCD, 0x81, 0x84, 0xC0, 0xEC,
0x71, 0x93, 0x18, 0x07, 0x4F, 0x38, 0xF1, 0xA6, 0xE8, 0x62, 0xF6, 0x01,
0x3E, 0x28, 0x9A, 0x86, 0xE8, 0x89, 0xFB, 0x2B, 0x3F, 0x67, 0xC1, 0xDD,
0x38, 0x3E, 0xB9, 0xEC, 0xFB, 0xB7, 0xBE, 0x23, 0x4A, 0xE9, 0x50, 0x9C,
0x49, 0xCD, 0x6A, 0x59, 0xD5, 0x80, 0x3C, 0x81, 0x18, 0xE5, 0xA5, 0xFD,
0xEE, 0xA3, 0xAF, 0x58, 0x76, 0x0E, 0xBF, 0x0E, 0xC5, 0x35, 0x2E, 0xFF,
0x90, 0xE6, 0x9A, 0xCC, 0x64, 0x59, 0x77, 0x69, 0x31, 0x11, 0x92, 0x67,
0xDD, 0xB6, 0x8E, 0x91, 0x11, 0x29, 0x2C, 0x15, 0x84, 0xB2, 0x01, 0xA2,
0x6D, 0xB8, 0xCA, 0xDD, 0x02, 0xBE, 0x4F, 0x32, 0x82, 0x4E, 0x1E, 0x9D,
0x18, 0xB9, 0x05, 0x44, 0x5B, 0xAF, 0x1D, 0xB2, 0xBE, 0x2D, 0x09, 0xA2,
0xD6, 0x81, 0xCD, 0x32, 0xFD, 0x48, 0xFF, 0x42, 0x8A, 0x43, 0xBC, 0x65,
0xB3, 0x54, 0x4F, 0x94, 0x83, 0xA6, 0x8D, 0x49, 0xAC, 0x02, 0x39, 0x57,
0x25, 0x29, 0xD4, 0x6F, 0x5E, 0x94, 0x8B, 0xB5, 0x2E, 0x75, 0xDF, 0x37,
0x0A, 0x72, 0xAB, 0x81, 0x56, 0x9B, 0xBE, 0xC5, 0x1E, 0x81, 0x6E, 0x65,
0xDE, 0x14, 0x03, 0xDA, 0xFF, 0xA2, 0xE7, 0x9A, 0xE1, 0x27, 0x6D, 0xD2,
0x04, 0x82, 0x34, 0x7C, 0xDF, 0x81, 0xA4, 0x61, 0xE1, 0xF9, 0xD7, 0x7D,
0xE9, 0x11, 0x5E, 0xD6, 0x40, 0x7C, 0x49, 0x46, 0xD1, 0xCC, 0x2A, 0x3E,
0x5D, 0x3C, 0xF4, 0xE1, 0x78, 0x04, 0xE2, 0x9F, 0xB4, 0x31, 0xB1, 0xBF,
0xFF, 0x38, 0x94, 0x7D, 0x09, 0x53, 0x3D, 0x69, 0x68, 0xBB, 0xF5, 0x85,
0x0A, 0xC9, 0x4A, 0x95, 0xE3, 0xFB, 0x63, 0xDB, 0xFD, 0xE8, 0x7A, 0x0A,
0x87, 0x5A, 0x70, 0x89, 0xE4, 0xAB, 0xF6, 0xE8, 0x1D, 0x79, 0xE7, 0x7B,
0xFC, 0x2B, 0xC7, 0xD1, 0x77, 0xF8, 0x20, 0xAE, 0x85, 0x24, 0xF1, 0x54,
0x20, 0x53, 0x09, 0x0C, 0x3C, 0x2A, 0x07, 0xB6, 0x49, 0xE3, 0x37, 0xD1,
0x93, 0xDA, 0x0E, 0x0A, 0x65, 0x65, 0x08, 0x5C, 0x3E, 0x73, 0x41, 0x84,
0x43, 0x5D, 0xA0, 0xD8, 0x90, 0xA9, 0x78, 0x1E, 0x2A, 0xC1, 0x8E, 0x23,
0x37, 0x5B, 0x25, 0x7E, 0x78, 0x5B, 0x81, 0x0E, 0xD3, 0xD5, 0x78, 0x38,
0x14, 0x1C, 0xBF, 0x0B, 0x1C, 0x5A, 0x20, 0xE4, 0x1C, 0xC0, 0x35, 0x36,
0x3E, 0x01, 0x30, 0x79, 0x59, 0x5B, 0xBF, 0x02, 0x14, 0x0E, 0xB0, 0x49,
0x77, 0x45, 0xBD, 0x41, 0x2B, 0x79, 0xB9, 0x1A, 0xD2, 0xEF, 0x93, 0x30,
0x7F, 0xED, 0x87, 0x7E, 0x4D, 0xA3, 0xEE, 0xE7, 0xF7, 0x1B, 0xCA, 0xD7,
0x57, 0x9C, 0x7E, 0xB3, 0x1F, 0x87, 0xC4, 0xE1, 0xC7, 0x82, 0xE8, 0x4F,
0xB1, 0xE7, 0xD2, 0xD5, 0x09, 0x45, 0xA7, 0x19, 0xF4, 0x5F, 0x7C, 0x22,
0x26, 0xA0, 0xD8, 0x18, 0x6C, 0x81, 0xB3, 0x55, 0xFD, 0x9E, 0xC7, 0x27,
0x43, 0x58, 0xEB, 0x72, 0x14, 0x3E, 0xEB, 0xC5, 0x46, 0xBA, 0x09, 0x39,
0x28, 0x29, 0x97, 0x9D, 0x76, 0x8A, 0x18, 0x25, 0x69, 0x8C, 0x98, 0x0A,
0xBD, 0x00, 0x47, 0xD7, 0xBB, 0x32, 0x4E, 0x7A, 0xB2, 0xD0, 0x64, 0xF3,
0x7F, 0x01, 0x76, 0xD5, 0x40, 0xFB, 0x7E, 0xEE, 0xFD, 0xE1, 0xCE, 0x1A,
0xC7, 0x0F, 0xB9, 0x0D, 0x52, 0x45, 0x46, 0x47, 0x29, 0xE0, 0xAE, 0xB9,
0x05, 0x09, 0x40, 0xD6, 0x16, 0x36, 0x89, 0x6B, 0x1E, 0xA6, 0x5A, 0x28,
0x09, 0x2D, 0x5B, 0x74, 0xBD, 0xD3, 0xF4, 0x53, 0xEB, 0x67, 0xE8, 0x1B,
0xC5, 0xFD, 0x14, 0x74, 0xFC, 0xA9, 0x04, 0x5F, 0xE0, 0x82, 0xFD, 0xF0,
0x51, 0x06, 0xC6, 0xAA, 0x7D, 0x73, 0x30, 0x5B, 0xD1, 0x2E, 0xF5, 0x0E,
0x41, 0xD8, 0x06, 0x3B, 0x5E, 0xF3, 0x44, 0xA6, 0x05, 0x30, 0xF5, 0x88,
0x03, 0x2E, 0x54, 0xA2, 0x33, 0x89, 0xC1, 0x82, 0x73, 0xC0, 0x33, 0x5A,

```
  0xBD, 0xD5, 0xCE, 0x99, 0x12, 0x26, 0xF2, 0x0B, 0x75, 0x4D, 0xDA, 0x75,
  0x43, 0x18, 0xD2, 0xB1, 0x01, 0xC1, 0x9F, 0x86, 0x56, 0x7A, 0xED, 0x0E,
  0x50, 0x50, 0xD5, 0x26, 0x6F, 0x83, 0x4F, 0x94, 0x36, 0x52, 0x18, 0xCB,
  0x00, 0xAE, 0x94, 0x90, 0x52, 0xBB, 0x30, 0x6A, 0xD2, 0xE4, 0x96, 0x07,
  0x71, 0x42, 0x92, 0xC6, 0x67, 0x85, 0x30, 0xF7, 0xAB, 0xCA, 0x4F, 0xD8,
  0x77, 0xAE, 0xA7, 0x2B, 0xBD, 0x9F, 0x2E, 0x18, 0x2B, 0x05, 0xF2, 0x30,
  0x5B, 0xA5, 0xDE, 0x99, 0x33, 0xA5, 0x06, 0x48, 0x76, 0x2C, 0xCA, 0x4F,
  0x0E, 0xA6, 0x05, 0xBD, 0x01, 0x0B, 0x3A, 0xE2, 0xEA, 0x94, 0xD3, 0x75,
  0x7F, 0x87, 0xBE, 0x49, 0xD6, 0x52, 0x9A, 0x07, 0x9D, 0x19, 0xB5, 0x24,
  0x99, 0xA9, 0x97, 0x1A, 0xC4, 0x2C, 0xAA, 0x16, 0x2E, 0xE4, 0x7A, 0x35,
  0x96, 0xF7, 0x09, 0xB6, 0xFC, 0x31, 0xEA, 0xE4, 0x5E, 0x17, 0x66, 0x79,
  0x84, 0x5C, 0x00, 0x0D, 0x98, 0x7B, 0x1C, 0x14, 0xC2, 0x98, 0xCC, 0xC3,
  0x4F, 0xAB, 0x0F, 0xD9, 0x5E, 0x57, 0x4D, 0x50, 0x19, 0x37, 0x35, 0x59,
  0x46, 0x1D, 0x38, 0xBF, 0x66, 0x35, 0x3C, 0xC1, 0x0E, 0x45, 0x84, 0xAC,
  0x1B, 0xA9, 0x91, 0x21, 0x42, 0x7F, 0xC8, 0xFC, 0xAF, 0xC2, 0x0C, 0x73,
  0x56, 0xFD, 0xFB, 0x20, 0xD8, 0xE7, 0x58, 0x6F, 0x7A, 0x6D, 0xEA, 0xDB,
  0x1C, 0xC4, 0xC1, 0x2C, 0x26, 0x26, 0x13, 0xDF, 0xC3, 0xE0, 0x5E, 0x5F,
  0xCC, 0x6F, 0xE4, 0x2F, 0xDD, 0x09, 0x5E, 0x20, 0x87, 0xC5, 0x74, 0x68,
  0x21, 0xD4, 0x5C, 0x3E, 0xB6, 0xF7, 0x8E, 0x6D, 0x1D, 0x56, 0x6F, 0x60,
  0xA6, 0xDA, 0x57, 0xDD, 0x05, 0x69, 0x43, 0xC1, 0x77, 0xDB, 0xB1, 0x34,
  0x1C, 0xF8, 0x3A, 0x2D, 0xE8, 0x25, 0x28, 0x95, 0x3B, 0xFE, 0x58, 0xD5,
  0xC6, 0x27, 0x20, 0x12, 0x43, 0x80, 0xC9, 0x2C, 0xAB, 0x0A, 0xF3, 0xA0,
  0xFB, 0xF0, 0xC7, 0x06, 0x4D, 0x65, 0x64, 0xF9, 0x91, 0x74, 0xC7, 0x88,
  0xF0, 0xAE, 0x03, 0x05, 0x25, 0xDD, 0x76, 0xF7, 0xAA, 0x0B, 0x1B, 0x36,
  0x24, 0x1C, 0xD1, 0x9D, 0xEC, 0x88, 0x45, 0xC3, 0xB5, 0xBC, 0x04, 0x6A,
  0x29, 0xA4, 0x3F, 0xAC, 0x43, 0xA7, 0x15, 0x73, 0xFF, 0x86, 0x53, 0xFA,
  0x72, 0x21, 0xD7, 0xB9, 0xB9, 0x06, 0x4D, 0xDA, 0x9C, 0x0D, 0x1D, 0xE9,
  0x55, 0x9F, 0x0F, 0xE5
};

#define OUTPUT_FILENAME    "certified.config"

int main(int argc, char **argv)
{
  sha1_context ctx;
  FILE *fout = NULL;
  unsigned char hash[20];
  char *input, *p;
  int c;
  size_t len;

  /* application name + author */
  puts("etpro config certification by chaplja");
  puts("chaplja.blogspot.com | chaplja@gmail.com ");

  /* usage information */
  if (argc != 2) {
```

```
 printf("Usage: %s \n", argv[0]);
 exit(1);
}

/* read and open necessary files */
if (!(input = read_text_file(argv[1]))) {
 printf("Failed reading '%s'\n", argv[1]);
 goto end;
}

if (!(fout = fopen(OUTPUT_FILENAME, "wb"))) {
 puts("Failed opening '" OUTPUT_FILENAME "'\n");
 goto end;
}

/* perform sha on the 'junk' data */
sha1_starts(&ctx);
sha1_update(&ctx, etpro_sig, sizeof(etpro_sig));

/* remove the existing signature from the config if it exists */
p = strstr(input, "\nsignature");
if (p)
 len = p - input + 1;
else
 len = strlen(input);

/* hash the actual config data and new line */
sha1_update(&ctx, (unsigned char *) input, len);
sha1_update(&ctx, (unsigned char *) "\r\n", 2);
sha1_finish(&ctx, hash);

/* write the certified config */
fwrite(input, len, 1, fout);
fwrite("\r\nsignature ", 12, 1, fout);
for (c = 0; c < 20; c++)
 fprintf(fout, "%02x", hash
);

/* success message */
puts("Successfully written a certified config to " OUTPUT_FILENAME);

/* cleanup */
end:
free(input);
if (fout)
fclose(fout);
```

```
return 0;
}

void *read_text_file(const char *path)
{
FILE *fp;
size_t flen;
void *file;
size_t cnt;

/* sanity check */
if (!path || !(*path))
return NULL;

/* open the file */
fp = fopen(path, "rb");
if (!fp) {
return NULL;
}

/* get file length */
fseek(fp, 0, SEEK_END);
flen = ftell(fp);
fseek(fp, 0, SEEK_SET);

/* allocate memory for the file data */
file = malloc(flen + 1);
if (!file) {
fclose(fp);
return NULL;
}

/* read the file from disk and append the nul char */
cnt = fread(file, 1, flen, fp);
((char *) file) [cnt]= 0;

/* data loaded, close the file */
fclose(fp);

return file;
}
```

Source