Independent Code Review

PTI Root Zone Key Ceremony Key Tools and Ceremony DVD

Executive Summary

Snake Hill Labs Inc was retained by PTI under a commercial contract to review code changes relating to code used in root zone DNSSEC key ceremonies.

Snake Hill Labs is an independent consulting company that was not involved in the code changes that were subject to review. The personnel assigned to do this work[1] have a wide combined experience in software development in C, DNS, DNSSEC, and the specific processes and infrastructure relating to DNSSEC as deployed in the root zone of the DNS.

Our findings are that all changes identified in the Root Zone Ceremony Key Tools archive and in the candidate ceremony DVD image were accounted for and are fit for the described purpose. We include cryptographically-signed attestations in this report to that effect.

Revision History

2017-04-12	Joe Abley	Draft for internal review
2017-04-13	Joe Abley	Circulated to PTI for review of format, structure, level of detail and attestation language
2017-04-17	Bill Snow	Minor edits and review
2017-04-18	Joe Abley	Final review and insertion of signed attestations.

[1] Joe Abley and Bill Snow

Table of Contents

Root Zone Ceremony Key Tools

Scope

The following text describing the relevant deliverable is reproduced verbatim from the Scope of Work provided to Snake Hill Labs by ICANN.

> *Attestation that all source code changes have been reviewed against the last release icann-keytools-20161012, and those changes are fit-for-purpose versus the described features in this document on section "Changes in this version 20170403".*

Definitions

The archive containing the key tools software used in KSK ceremony 27, published with filename `icann-keytools-20161012.tar.gz`, will be referred to in this document as the Old Archive.

The archive that is proposed to replace the Old Archive, published with filename `icann-keytools-20170403.tar.gz`, will be referred to in this document as the New Archive.

Documented Changes in the New Archive

Below is a section quoted verbatim from the specification for this review - *Key Manager SW Update KSK Rollover Rev-20170403.pdf*. The paragraph and bullet numbers in this section will be referred to in the sections that follow.

> ```
> The following describes the new functionality in this version 20170403 of
> the Key Management Software versus the previous release
> (icann-keytools-20161012):
>
> Changes to the KSR Signer "ksrsigner" component
>
> 1. ksrsigner/ksrcommon.c:
> a. Added support to process the configuration file "kskschedule.json".
> b. Added support to process the configuration options to sign, publish or
> revoke the KSK.
>
> 2. ksrsigner/ksrcommon.h:
> a. Added definition of json key schedule parsing support.
>
> 3. ksrsigner/ksrsigner.c:
> a. Added support to locate the configuration file kskschedule.json in the
> same place as the KSR.xml file.
>
> 4. ksrsigner/wksr.c:
> ```

```
a. Added support to locate the configuration file kskschedule.json in the
same place as the KSR.xml file.

Please refer to the Appendix 1: SKR Configurations Schemas for more details.

Changes to other component

5. utils/printlog:
a. Removed top margin.
b. Added header with name and page.
c. Reduced font size to fit better on the page.

6. New: utils/hsmfd-hash:
a. A bash script that calculate, print and compare sha-256 hash for HSMFDs.

7. ChangeLog:
a. Added a change log file.

8. README.md:
a. Update copyright year to 2017
```

Methodology

1. Obtain the Old Archive and the New Archive;
2. Calculate the SHA256 digest of each archive and record them;
3. Verify that the digests calculated in (2) match those provided in correspondence from PTI;
4. Compare the list of files included in each archive and account for any differences;
5. Compare every file that is common to both archives and account for any differences;
6. Cross-reference each change with reference to the changelog provided by PTI and confirm in each case that the changes are fit for the stated purpose.

Analysis

Obtain the Archives to be Compared

The code is stored in the icann-keytools archives. The Old Archive was retrieved from the IANA repository of materials corresponding to KSK ceremony 27, at https://data.iana.org/ksk-ceremony/27/icann-keytools-20161012.tar.gz.

The New Archive, `icann-keytools-20170403.tar.gz` was provided directly by PTI, together with a SHA256 digest.

Calculate the SHA256 Digests of Each Archive

The New Archive and the Old Archive were obtained from the sources described above. An additional file ff was retrieved from the same location as the New Archive. This third file contained the SHA256 digest as calculated by PTI staff.

The SHA256 digests calculated by us over the Old Archive and the New Archive can be used by third parties to confirm that the analysis and attestations in this document are applicable to local copies of each archive, over which those third parties have calculated their own SHA256 digests.

```
$ sha256sum icann-keytools-20161012.tar.gz
f32b6fc5b13730ead6bdd2addb3bf7d832b1012f8fc4213322ac1de5cb1d2201
icann-keytools-20161012.tar.gz
$ curl https://data.iana.org/ksk-ceremony/27/icann-keytools-20161012.tar.gz.sha256
f32b6fc5b13730ead6bdd2addb3bf7d832b1012f8fc4213322ac1de5cb1d2201
icann-keytools-20161012.tar.gz
$ sha256sum icann-keytools-20170403.tar.gz
9d2f83eb4d7a1b134e2fc0d4234032a5d06222173e55115bab47b4abb91e8711
icann-keytools-20170403.tar.gz
$ cat icann-keytools-20170403.tar.gz.sha256
9d2f83eb4d7a1b134e2fc0d4234032a5d06222173e55115bab47b4abb91e8711
icann-keytools-20170403.tar.gz
```

Compare the List of Files in Each Archive

A list of files included in each of the two archives was obtained and compared, as follows.

```
$ tar tzf icann-keytools-20161012.tar.gz | sed -s \
's#icann-keytools-20161012/##'  > icann-keytools-20161012.index
$ tar tzf icann-keytools-20170403.tar.gz | sed -s \
's#icann-keytools-20170403/##'  > icann-keytools-20170403.index
$ diff icann-keytools-20161012.index icann-keytools-20170403.index
```

The resulting number of differences was large. Most of the differences are understood to be an artefact of the way the most recent archive was created, consistent with known behaviour of tar(1) shipped with MacOS which uses the AppleDouble[2] format to store extended attributes and access control lists. The new archive has 104 such corresponding files, each associated with one of the files in the archive; the previous archive had none.

```
$ tar tzf icann-keytools-20170403.tar.gz | grep -c '\._'
105
$ tar tzf icann-keytools-20161012.tar.gz | grep -c '\._'
0
$
```

There are three other differences between the list of files in each archive:

```
$ tar tzf icann-keytools-20170403.tar.gz | \
sed -s 's#icann-keytools-20170403/##' | \
grep -v "\._" | \
sort > icann-keytools-20170403.index
$ tar tzf icann-keytools-20161012.tar.gz | \
sed -s 's#icann-keytools-20161012/##' | \
sort > icann-keytools-20161012.index
$ diff icann-keytools-20161012.index icann-keytools-20170403.index
2a3
```

[2] https://en.wikipedia.org/wiki/AppleSingle_and_AppleDouble_formats

```
> ChangeLog
66a68
> icann-keytools-20170403-source.sha256
99a102
> utils/hsmfd-hash
$
```

These new files are:

- `ChangeLog`, correponding to change 7a,
- `hsmfd-hash`, corresponding to change 6a and
- `icann-keytools-20170403-source.sha256`, which contains pre-computed hash values for the contents of the New Archive.

We consider all of these differences to be benign, and that they do not affect the operation of the software under examination. There are no other differences in the list of files included in each archive.

Compare Every File Common to Both Archives

Every file present in icann-keytools-20161012.tar.gz was compared with the corresponding file in icann-keytools-20170403.tar.gz, as follows. Where changes were found, analysis is interspersed below.

```
$ mkdir cmp
$ cd cmp
$ tar xzf ../icann-keytools-20161012.tar.gz
$ tar xzf ../icann-keytools-20170403.tar.gz
$ for NEXT in `grep -v "/$" ../icann-keytools-20161012.index`
do
echo "********** $NEXT ***********"
diff icann-keytools-20161012/$NEXT icann-keytools-20170403/$NEXT
done
********** autogen.sh ***********
********** changepin/changepin.c ***********
********** changepin/Makefile.am ***********
********** checks/baseline/KSK1.csr ***********
********** checks/baseline/KSK2.csr ***********
********** checks/baseline/ksr-root-2009-q4-2.xml ***********
********** checks/baseline/ksr-root-2010-q1-0.xml ***********
********** checks/baseline/ksr-root-2010-q2-0-revoke.xml ***********
********** checks/baseline/ksr-root-2010-q2-0.xml ***********
********** checks/baseline/skr-root-2009-q4-2.xml ***********
********** checks/baseline/skr-root-2010-q1-0.xml ***********
********** checks/baseline/skr-root-2010-q2-0-revoke.xml ***********
********** checks/baseline/skr-root-2010-q2-0.xml ***********
********** checks/Makefile.am ***********
********** checks/pkcs11spy.hsmconfig.in ***********
********** checks/README.txt ***********
********** checks/regress-kskgen.sh ***********
********** checks/regress-ksrsigner.sh ***********
```

```
********** checks/rsa2048-1.key **********
********** checks/rsa2048-2.key **********
********** checks/softhsm-dual.conf **********
********** checks/softhsm.hsmconfig.in **********
********** checks/softhsm-single.conf **********
********** common/base32.c **********
********** common/base32.h **********
********** common/base64.c **********
********** common/base64.h **********
********** common/cryptoki/cryptoki.h **********
********** common/cryptoki/pkcs11f.h **********
********** common/cryptoki/pkcs11.h **********
********** common/cryptoki/pkcs11t.h **********
********** common/dnssec.c **********
********** common/dnssec.h **********
********** common/logger.c **********
********** common/logger.h **********
********** common/mbuf.c **********
********** common/mbuf.h **********
********** common/pkcs11_dnssec.c **********
********** common/pkcs11_dnssec.h **********
********** common/rlder.c **********
********** common/rlder.h **********
********** common/sha1.c **********
********** common/sha1.h **********
********** common/sha2.c **********
********** common/sha2.h **********
********** common/sha2wordlist.h **********
********** common/util.c **********
********** common/util.h **********
********** compat/compat.h **********
********** compat/strlcat.c **********
********** compat/strlcat.h **********
********** compat/strlcpy.c **********
********** compat/strlcpy.h **********
********** configure.ac **********
********** Doxyfile.in **********
********** doxygen.am **********
********** extras/svnignore.sh **********
********** extras/svnignore.txt **********
********** keybackup/keybackup.c **********
********** keybackup/Makefile.am **********
********** kskgen/kskgen.c **********
********** kskgen/kskgen.h **********
********** kskgen/kskparams.h **********
********** kskgen/Makefile.am **********
********** kskgen/Makefile.pre **********
********** ksrsigner/ksrcommon.c **********
543,544c543,549
<       cX -= snprintf(&lbuf[jj],cX,"%05u%s",y->keyTag,
<                      y->Flags & DNSKEY_REVOKE_FLAG ? "/R" : "");
```

```
---
>       char kuse[5];
>       signature *s;
>       kuse[0] = '\0';
>       if( (y->Flags & DNSKEY_REVOKE_FLAG) ) strcat(kuse,"R");
>       for(s=rq->x_sig;s;s=s->next)
if(strcmp(s->keyIdentifier,y->keyIdentifier) == 0) break;
>       if(s) strcat(kuse,"S"); else strcat(kuse,"P"); // signed or
pub only
>       cX -=
snprintf(&lbuf[jj],cX,"%05u(%s)/%s",y->keyTag,y->keyIdentifier,kuse);
// note: Revoke flag changes tag value
```

Formerly printed "{keytag}" or "{keytag}/R" (if revoke is a flag in y) into syslog. Now prints "{keytag}({keyid})/[R][SP]" where R indicates revoke and S or P indicate Sign or Publish into syslog.

This correspond to change 1b, even though the change list does not explicitly indicate logging changes to accommodate the sign/revoke/publish configuration options.

```
904,912c909
<       if(debug)
<         logger_info("Verified private key ownership for
%05u",s->KeyTag);
< #ifdef VALIDATE_VALIDITY
<       /* find max VALIDATED expiration and inception time */
<       if(s->SignatureExpiration > rq->expmax)
<         rq->expmax = s->SignatureExpiration;
<       if(s->SignatureInception < rq->incmin)
<         rq->incmin = s->SignatureInception;
< #endif
---
>       if(debug) logger_info("Verified private key ownership for
%05u",s->KeyTag);
```

This drops the two checks, which existed only when VALIDATE_VALIDITY, in support of change 1b.

```
1025a1023,1038
>   kskslot *ks;
>   ks = NULL;
>   loadkeyschedule();   /* Try to load generalized key schedule json
configuration */
>   if(kskсh0) {
>     i = 0;
>     for(ks=kskсh0->s;ks;ks=ks->next) i++;
>     if(i != reqscnt) {
>       logger_error("Number of slots in KSR and JSON (file:%s)
config do not match",JSONKSCHEDULEFILE);
>       ksrinvalid++;
```

```
>       goto enderror;
>     }
>     ks = kskschO->s; // Note: multiple KSK schedules are loaded but
only first one is used.
>     myx_syslog(LOG_INFO,"Reading KSK schedule \"%s\" from
\"%s\"\n",kskschO->name,JSONKSCHEDULEFILE);
>     myx_syslog(LOG_INFO,"#  KSK Tag(CKA_LABEL)\n");
>   }
>
```

Corresponds to change 1a. This is the high-level invocation of config file loading.

```
1032a1046,1123
>     if(ks) { /* Use json config file */
>       int j;
>       for(i=0;i<nksk;i++) {
>     ksks[i]->signer = 0;
>     if(ksks[i]->user) {
>       free(ksks[i]->user);
>       ksks[i]->user = NULL;
>     }
>       }
>       for(j=0;j<ks->n_pub;j++) {
>     for(i=0;i<nksk;i++) {
>       if(strcmp((char *)ksks[i]->label->p0,ks->cka_label_pub[j]) ==
0) {
>         keys[keycnt++] = ksks[i];
>         ksks[i]->Flags &= ~DNSKEY_REVOKE_FLAG; /* clear prior use
*/
>         ksks[i]->keyTag = updatekeytag(ksks[i]);
>         break;
>       }
>     }
>     if(i == nksk) {
>       logger_error("Key not in HSM %s",ks->cka_label_pub[j]);
>       ksrinvalid++;
>       goto enderror;
>     }
>       }
>       for(j=0;j<ks->n_revoke;j++) {
>     for(i=0;i<nksk;i++) {
>       if(strcmp((char *)ksks[i]->label->p0,ks->cka_label_revoke[j])
== 0) {
>         keys[keycnt++] = ksks[i];
>         ksks[i]->Flags |= DNSKEY_REVOKE_FLAG;
>         ksks[i]->keyTag = updatekeytag(ksks[i]);
>         break;
>       }
>     }
>     if(i == nksk) {
>       logger_error("Key not in HSM %s",ks->cka_label_revoke[j]);
```

```
>       ksrinvalid++;
>       goto enderror;
>     }
>       }
>       for(j=0;j<ks->n_sign;j++) {
>     for(i=0;i<keycnt;i++) {
>       if(strcmp((char *)keys[i]->label->p0,ks->cka_label_sign[j])
== 0) { // Assumes no 2 KSK CKA_LABELS the same
>         if(pkcs11_have_private_key(keys[i]->pkcb) == 0) {
>           logger_error("Do not have private key for
%s",ks->cka_label_sign[j]);
>           ksrinvalid++;
>           goto enderror;
>         }
>         keys[i]->signer = 1;
>         break;
>       }
>     }
>     if(i == keycnt) {
>       logger_error("Key not in HSM %s",ks->cka_label_sign[j]);
>       ksrinvalid++;
>       goto enderror;
>     }
>       }
>
>       char lbuf[MAXPATHLEN];
>       krecord *y;
>       int cX,jj;
>       char kuse[5];
>       cX = sizeof(lbuf);
>       jj = 0;
>       for(i=0;i<keycnt;i++) {
>     y = keys[i];
>     kuse[0] = '\0';
>     if( (y->Flags & DNSKEY_REVOKE_FLAG) ) strcat(kuse,"R");
>     if(y->signer) strcat(kuse,"S"); else strcat(kuse,"P"); //
signed or pub only
>     if(i) { lbuf[jj] = ','; jj++; cX--; }
>     j =
snprintf(&lbuf[jj],cX,"%05u(%s)/%s",y->keyTag,y->keyIdentifier,kuse);
// note: Revoke flag changes tag value
>     cX -= j;
>     jj += j;
>       }
>       myx_syslog(LOG_INFO,"%d %s\n",ir+1,lbuf);
>
>       ks = ks->next; // next parralel ksk instructions
>     } else
```

This change is within the high level function to sign the KSR, `int signem(FILE *ftmp,xmlstate *xs)`. It adds another mechanism to set up the variable "keys" within the signem function, using a JSON-format configuration file, as described in change 1a.

```
1124c1215
<      * add ZSKs to sign
---
>      * add ZSKs to sign AFTER KSKs
```

This is a comment change, and helpful, loosely related to change 1b.

```
1256,1261d1346
< #ifdef VALIDATE_VALIDITY
<       if(s->SignatureExpiration > rq->expmax)
<         rq->expmax = s->SignatureExpiration;
<       if(s->SignatureInception < rq->incmin)
<         rq->incmin = s->SignatureInception;
< #endif
```

This drops two checks, which existed only when the macro `VALIDATE_VALIDITY` is defined, in support of change 1b.

```
1439a1525,1739
> /*! Read in JSON config files
>   Added 13 January 2017 to support Jakob Schlyter config files
>   for flexible key scheduling for first root KSK rollover. RHLamb
> */
>
> /*
> Note: CKA_LABEL for Root KSK Operations is time() encoded in
BASE32, i.e.,
> static const char cb32[] = "abcdefghijklmnopqrstuvwxyz234567";
> static const char cb32_ucase[] =
"ABCDEFGHIJKLMNOPQRSTUVWXYZ234567";
> */
>
> static kskschedule *ksksch=NULL;
>
> static void kskschedule_add(char *label)
```

Only called from `ksrcommon.c`.

```
> {
>   kskschedule *ks;
>
>   if(label == NULL) return;
>   ks = (kskschedule *)malloc(sizeof(kskschedule));
>   memset(ks,0,sizeof(kskschedule));
>   if(ksksch0 == NULL) {
>     ksksch0 = ks;
>   } else {
>     if(ksksch->next) {
```

```
>       printf("%s: Error!!\n",__func__);
>       exit(1);
>     }
>     ksksch->next = ks;
>   }
>   ksksch = ks;
>   ksksch->name = strdup(label);
> }
> static void kskschedule_add_seq(char *label)
```

Only called from `ksrcommon.c`.

```
> {
>   kskslot *ks;
>
>   if(label == NULL) return;
>   if(ksksch == NULL) {
>     printf("%s: Error!!\n",__func__);
>     exit(1);
>   }
>   ks = (kskslot *)malloc(sizeof(kskslot));
>   memset(ks,0,sizeof(kskslot));
>   ks->seq = atoi(label);
>   if(ksksch->s == NULL) {
>     ksksch->s = ks;
>   } else {
>     kskslot *s;
>     for(s=ksksch->s;s;s=s->next) {
>       if(s->next == NULL) break;
>     }
>     s->next = ks;
>   }
> }
> static void kskschedule_add_key(char *label,char *key)
```

Only called from `ksrcommon.c`.

```
> {
>   kskslot *ks;
>   kskslot *s;
>   char *t_key,*args[JSON_NKEYS];
>   int i;
>
>   if(label == NULL || key == NULL) return;
>   if(ksksch == NULL) {
>     printf("%s: Error A!!\n",__func__);
>     exit(1);
>   }
>   if(ksksch->s == NULL) {
>     printf("%s: Error B!!\n",__func__);
>     exit(1);
>   }
```

```
>   for(s=kskschch->s;s;s=s->next) {
>     if(s->next == NULL) break;
>   }
>   if(strcmp(label,"publish") == 0) {
>     if(s->cka_label_pub[0]) {  printf("%s: Error C
publish!!\n",__func__); exit(1); }
>     t_key = strdup(key);
>     s->n_pub = lparse(t_key,args,JSON_NKEYS,',');
>     for(i=0;i<s->n_pub;i++) s->cka_label_pub[i] = strdup(args[i]);
>     free(t_key);
>   } else if(strcmp(label,"sign") == 0) {
>     if(s->cka_label_sign[0]) {  printf("%s: Error C
sign!!\n",__func__); exit(1); }
>     t_key = strdup(key);
>     s->n_sign = lparse(t_key,args,JSON_NKEYS,',');
>     for(i=0;i<s->n_sign;i++) s->cka_label_sign[i] =
strdup(args[i]);
>     free(t_key);
>   } else if(strcmp(label,"revoke") == 0) {
>     if(s->cka_label_revoke[0]) {  printf("%s: Error C
revoke!!\n",__func__); exit(1); }
>     t_key = strdup(key);
>     s->n_revoke = lparse(t_key,args,JSON_NKEYS,',');
>     for(i=0;i<s->n_revoke;i++) s->cka_label_revoke[i] =
strdup(args[i]);
>     free(t_key);
>   } else {
>     printf("%s: Error C |%s|!!\n",__func__,label);
>     exit(1);
>   }
> }
> static void kskschedule_print()
```

Only called from `ksrcommon.c`.

```
> {
>   kskschedule *ks;
>   kskslot *s;
>   int j;
>   char kpub[80],krev[80],ksign[80];
>   for(ks=kskschch0;ks;ks=ks->next) {
>     printf("%s: %s\n    %15s %15s
%15s\n",__func__,ks->name,"Pub","Revoke","Sign");
>     for(s=ks->s;s;s=s->next) {
>       kpub[0] = krev[0] = ksign[0] = '\0';
>       for(j=0;j<s->n_pub;j++) { if(j) strcat(kpub,",");
strcat(kpub,s->cka_label_pub[j]); }
>       for(j=0;j<s->n_revoke;j++) { if(j) strcat(krev,",");
strcat(krev,s->cka_label_revoke[j]); }
>       for(j=0;j<s->n_sign;j++) { if(j) strcat(ksign,",");
strcat(ksign,s->cka_label_sign[j]); }
>       printf(" %d: %15s %15s %15s\n",s->seq,kpub,krev,ksign);
```

```
>     }
>   }
> }
>
> #define LABELBUFLEN 2560
> static FILE *jsonfp=NULL;
> static int jsondepth=0;
>
> static int jparse(void)
> {
>   int c,quote,bracket,n;
>   char *p,buf[LABELBUFLEN],label[LABELBUFLEN];
>
>   jsondepth++;
>   p = label;
>   n = 0;
>   quote = 0;
>   bracket = 0;
>   while((c=fgetc(jsonfp)) != EOF) {
>
>     if(c == '\n' || c == '\r' || c == ' ' || c == '\t') continue;
>
>     if(quote == 1) {
>       if(c == '"') {
>         quote = 0;
>       } else {
>     if(c != '\\') { *p++ = c; n++; }
>       }
>       continue;
>     }
>     if(c == '"') { quote = 1; continue; }
>
>     else if(bracket == 1) {
>       if(c == ']') {
>     bracket = 0;
>       } else {
>     if(c != '\\') { *p++ = c; n++; }
>       }
>       continue;
>     }
>     if(c == '[') { bracket = 1; continue; }
>
>
>     else if(c == '{') {
> #ifdef JSONTEST
>       printf("\n");
> #endif
>       *p++ = '\0';
>       p = label;
>       n = 0;
>       jparse(); // go till next '}'
```

```
> #ifdef JSONTEST
>       printf("*eol*%d*\n",jsondepth);
> #endif
>     } else if(c == ':') { // label
>       *p++ = '\0';
> #ifdef JSONTEST
>       printf("%s:",label);
> #endif
>       if(jsondepth == 3) kskschedule_add(label);
>       if(jsondepth == 4) kskschedule_add_seq(label);
>
>       p = buf;
>       n = 0;
>
>       //*p = '\0';
>
>     } else if(c == ',' || c == '}') { // contents
>
>       *p++ = '\0';
> #ifdef JSONTEST
>       printf("%s",buf);
> #endif
>       if(jsondepth == 5) kskschedule_add_key(label,buf);
>
>       p = label;
>       n = 0;
>
>       if(c == '}') {
>     jsondepth--;
>     return 0;
>       }
>     } else if(c == '\\') {
>       continue;
>     } else {
>       *p++ = c;
>       n++;
>     }
>   }
>   return 0;
> }
> int loadkeyschedule(void)
```

Only called from `signem()`.

```
> {
>   int ret;
>   char lbuf[MAXPATHLEN];
>   extern char *ksrpath;
>
>   if(ksrpath)
snprintf(lbuf,sizeof(lbuf),"%s/%s",ksrpath,JSONKSCHEDULEFILE);
```

```
>   else snprintf(lbuf,sizeof(lbuf),"%s",JSONKSCHEDULEFILE);
>   if((jsonfp=fopen(lbuf,"r")) == NULL) return -1;
>   ret = jparse();
>   fclose(jsonfp);
>   if(debug) kskschedule_print();
>   return ret;
> }
```

This block is the bulk of the config loading code, and is straightforward. It corresponds to change 1a.

```
********** ksrsigner/ksrcommon.h ***********
161a162,181
> /* json key schedule parsing support */
>
> #define JSONKSCHEDULEFILE "kskschedule.json"
> typedef struct _kskslot {
>   struct _kskslot *next;
>   int seq;
>   int n_pub,n_revoke,n_sign;
> #define JSON_NKEYS 5
>   char *cka_label_pub[JSON_NKEYS];
>   char *cka_label_sign[JSON_NKEYS];
>   char *cka_label_revoke[JSON_NKEYS];
> } kskslot;
> typedef struct _kskschedule {
>   struct _kskschedule *next;
>   char *name;
>   kskslot *s;
> } kskschedule;
> static kskschedule *ksksch0=NULL;
> int loadkeyschedule(void);
>
```

Defines the structures as in change 2a.

```
********** ksrsigner/ksrpolicy.h ***********
********** ksrsigner/ksrsigner.c ***********
39a40
> char *ksrpath=NULL;
116c117
<       char *ksrpath;
---
>
136c137
<       free(ksrpath);
---
>
```

This just uses an `extern` (in `ksrcommon.c`) for existing `ksrpath`, consistent with change 3a.

```
********** ksrsigner/ksrsigner.h **********
********** ksrsigner/Makefile.am **********
********** ksrsigner/Makefile.pre **********
********** ksrsigner/wksr.c **********
42a43
> char *ksrpath=NULL;
```

Required by `ksrcommon.c` (extern) in order to build the `wksr` binary, consistent with change 4a. Functional change happens in the library.

```
********** m4/acx_32bit.m4 **********
********** m4/acx_64bit.m4 **********
********** m4/acx_analyze.m4 **********
********** m4/acx_pedantic.m4 **********
********** m4/acx_strict.m4 **********
********** m4/ax_check_opensc.m4 **********
********** m4/ax_check_openssl.m4 **********
********** m4/ax_check_softhsm.m4 **********
********** m4/ax_prog_doxygen.m4 **********
********** Makefile.am **********
********** README.md **********
11c11
<     Copyright (c) 2010-2016 Internet Corporation for Assigned Names
and
---
>     Copyright (c) 2010-2017 Internet Corporation for Assigned Names
and
```

Updated date, consistent with change 8a.

```
********** utils/digest **********
********** utils/displaycsr **********
********** utils/Makefile.am **********
********** utils/printlog **********
15c15,18
< top_margin=200
---
> ## --margins=left:right:top:bottom
> ## top_margin=200 #for ICANN header
> ##--margins=::${top_margin}: \
> ##--no-header \
22,25c25,28
<     --no-header \
<     --margins=::${top_margin}: \
<     --font=Courier10 \
<     --setpagedevice=Collate:true
---
>     --font=Courier7.5 \
>     --setpagedevice=Collate:true \
>     --header=$LOGFILE'||Page $% of $='
```

```
>
```

Only a change to formatting, consistent with change 5.

```
********** utils/ttyaudit ***********
********** version.m4 ***********
```

Results

All observed differences are related to the documented change list except for the following which we consider to be minor:

- All `._*` (AppleDouble) files present in the new archive, artefacts of the way the archive was constructed;
- Presence of `icann-keytools-20170403-source.sha256` in the new archive;
- Removal of conditional validation of expiration and inception time, which was in any case not previously enabled and hence whose removal has no effect;
- Syslog output to include more configuration detail;
- Comments in code (`ksrcommon.c`).

Based on our baseline assumption that the software contained within the Old Archive was fit for the purpose, and the analysis described above of the differences between the Old Archive and the New Archive, it is our considered opinion that the software contained within the New Archive is also fit for the purpose.

Attestation[3]

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

Snake Hill Labs Inc was retained by PTI under a commercial contract
to review source code changes relating to code used during root
zone DNSSEC key ceremonies.

The personnel involved in the analysis and technical assessment
described in this attestation were William Snow and Joseph Abley.

We were provided with two archives containing source code.  The
names of those archives, together with their respective SHA256
digests, are as follows:

$ sha256sum icann-keytools-20161012.tar.gz | fmt
f32b6fc5b13730ead6bdd2addb3bf7d832b1012f8fc4213322ac1de5cb1d2201
icann-keytools-20161012.tar.gz
$

$ sha256sum icann-keytools-20170403.tar.gz | fmt
9d2f83eb4d7a1b134e2fc0d4234032a5d06222173e55115bab47b4abb91e8711
icann-keytools-20170403.tar.gz
$
```

[3] This attestation is signed using PGP key 0xE1B9976C which is published on various public key servers including the cluster reachable at `hkps.pool.sks-keyservers.net`.

```
We accounted for and documented the differences in the contents of
those two archives which did not correspond to source code changes.

We analysed the differences in the contents of those two archives
that did correspond to source code changes. In every case, the
changes corresponded to documented and intended functionality, as
documented in the change log provided to us by PTI.

We found no changes in source code that appeared to have any other
purpose.

Our assessment is that the changes to source code described above
are fit for the purpose described in the aforementioned change log.

This attestation has been signed using OpenPGP key 0xE1B9976C in
London, Ontario, Canada on Tuesday 18 April 2017.

Joseph Abley
Director, Principal Consultant
Snake Hill Labs Inc

-----BEGIN PGP SIGNATURE-----
Comment: GPGTools - https://gpgtools.org

iQIcBAEBCgAGBQJY9jeAAAoJEDu28RLhuZdsrmEP/05BnngrW5uc9iLgABX7GrZJ
+NCGDQXK4qn+MRp3Wj84A0c1Gn/vk0kGPevLza9NqKR2i2jAsPbETjQME1Aigv38
7G38QfTt3ouJAPKD1fJgUXf3fKSI8yqOiEBCFskYH1cXa8ibxp22UrDNhyxguiaO
UBHop5YFkyBmJ+0JnoMDu2EQPhRvf4hglTzjGtDUe8CyMvmDOFlOKX6ySYWTTqPL
QKNT3HIwbxqo5EONAGtlvalS+IOfPhaoS1Z8NoCMgqZU41N6CqysJOJgrQWtF2xW
4BMj+duAOt66DGHkRrNg/8K+f1fRStXs1o2YRWyOHQHC50mEw0hI1y27KmQd9gfR
GKaaPXv+GKAJR9x3cRrkoaYkZx/n40mR0ADjQb8+9W3fqomsUuotXF71CyMk53xE
Nd1woFQm9l+QoTvE2NKDoKZFI1OYbiepmOsSShE/7FerhuJ4WJbtchP+xzXiG8VL
rEUTAmM52cFJQFtG7ZkkSjfHArakuEDf05gc8X/gcIueBacqxjKaQa4rrCJKS+Ir
ubikkPMWc3n9IZ+BMzGyZLEV67uD4qE6DBL333R25t7rrCD6lPI1bZpJY0TZs8vO
9VyykcE+lBmMMlglBjonXCixHufg5z2PP7816ldjSp9tCdqCh0lo1U8ysD4A8HqY
3X3qLRYHXbbp+jWXIsM1
=JugL
-----END PGP SIGNATURE-----
```

Root Zone Key Ceremony DVD Image

Scope

The following text describing the relevant deliverable is reproduced verbatim from the Scope of Work provided to Snake Hill Labs by ICANN.

> *Attestation that the new version of the Operating System image only includes changes that are related to the Key Management Software and the OS (Operating System) described in this document, otherwise remains the same as the previous version KC-20161014.iso.*

Definitions

The Root Zone Key Ceremony DVD Image corresponding to the DVD media most recently used in KSK ceremony 27, published with filename `KC-20161014.iso`, will be referred to in this document as the Old Image.

The DVD Image that is proposed to replace the Old Image, published with filename `KC-20170403.iso`, will be referred to in this document as the New Image.

Methodology

1. Obtain the Old Image and the New Image;
2. Calculate the SHA256 digest of each image and record them;
3. Verify that the digests calculated in (2) match those provided in correspondence from PTI;
4. Mount both images on a test machine;
5. Compare the filesystems below each mount point, including all squashfs and ext3fs filesystems contained within the image and account for any differences.

Analysis

Obtain the Images to be Compared

The Old Image and the New Image were provided to us by PTI. We obtained four files, `KC-20161014.iso`, `KC-20161014.iso.sha256`, `KC-20170403.iso` and `KC-20170403.iso.sha256`. The files with suffix `.sha256` contain the SHA256 digests of the corresponding `.iso` files, as calculated by PTI.

Calculate the SHA256 Digests of Each Image

There is no integrity protection of the SHA256 digests that were provided to us by PTI, and they are included in this report only in the interests of completeness. The SHA256 digests calculated by us over the Old Image and the New Image can be used by third parties to confirm that the analysis and attestations in this document are applicable to local copies of each image, over which those third parties have calculated their own SHA256 digests.

```
$ sha256sum KC-201*.iso | fmt
991f7be8cfbc3b4bdb6f5e5f84092486755a08a3c36712e37a26ccd808631692
KC-20161014.iso
4d127c7db1a564399c0f4e00b34d6a7611e23cdb96cd64f3a428a16319285041
KC-20170403.iso
$ cat KC-201*.sha256 | fmt
991f7be8cfbc3b4bdb6f5e5f84092486755a08a3c36712e37a26ccd808631692
KC-20161014.iso
4d127c7db1a564399c0f4e00b34d6a7611e23cdb96cd64f3a428a16319285041
KC-20170403.iso
```

```
$
```

Mount Both Images

Each image was mounted as follows:

```
$ sudo mount -o loop KC-20161014.iso KC-20161014
mount: /dev/loop0 is write-protected, mounting read-only
$ sudo mount -o loop KC-20170403.iso KC-20170403
mount: /dev/loop1 is write-protected, mounting read-only
$
```

Compare the Filesystems below each Mount Point

Each change identified below are categorised into one of three types:

1. Changes to Root Zone Ceremony Key Tools (q.v.);
2. Operating system changes;
3. Changes of dates and serial numbers relating to the overall replacement of images.

We carried out the comparison as follows.

```
$ diff -r KC-20161014/ KC-20170403/
diff -r KC-20161014/isolinux/isolinux.cfg
KC-20170403/isolinux/isolinux.cfg
6c6
< menu title Welcome to KC-20161014!
---
> menu title Welcome to KC-20170403!
22c22
<   append initrd=initrd0.img root=CDLABEL=KC-20161014
rootfstype=iso9660 ro quiet liveimg rhgb
---
>   append initrd=initrd0.img root=CDLABEL=KC-20170403
rootfstype=iso9660 ro quiet liveimg rhgb
```

A simple change in labelling, categorised as a change of type 3.

```
Binary files KC-20161014/LiveOS/squashfs.img and
KC-20170403/LiveOS/squashfs.img differ
```

This is a squashfs version 3 filesystem. We unpack it and continue the comparison.

```
$ mkdir KC-20161014.squashfs
$ cd KC-20161014.squashfs/
$ unsquashfs ../KC-20161014/LiveOS/squashfs.img
Parallel unsquashfs: Using 16 processors
1 inodes (65536 blocks) to write
```

```
[==============================================================
==============================================================
==================================\] 65536/65536 100%

created 1 files
created 2 directories
created 0 symlinks
created 0 devices
created 0 fifos
$ cd ../
$ mkdir KC-20170403.squashfs
$ cd KC-20170403.squashfs
$ unsquashfs ../KC-20170403/LiveOS/squashfs.img
Parallel unsquashfs: Using 16 processors
1 inodes (65536 blocks) to write

[==============================================================
==============================================================
==================================|] 65536/65536 100%

created 1 files
created 2 directories
created 0 symlinks
created 0 devices
created 0 fifos
$ diff -r KC-20161014.squashfs/ KC-20170403.squashfs/
Binary files KC-20161014.squashfs/squashfs-root/LiveOS/ext3fs.img and
KC-20170403.squashfs/squashfs-root/LiveOS/ext3fs.img differ
```

The only change in the squashfs is in an ext3fs image, so we must recurse into that.

```
$ mkdir KC-20161014.ext3fs
$ mkdir KC-20170403.ext3fs
$ sudo mount -o loop
KC-20161014.squashfs/squashfs-root/LiveOS/ext3fs.img
KC-20161014.ext3fs/
$ sudo mount -o loop
KC-20170403.squashfs/squashfs-root/LiveOS/ext3fs.img
KC-20170403.ext3fs/
$ sudo diff -r --no-dereference KC-20161014.ext3fs/
KC-20170403.ext3fs/
File KC-20161014.ext3fs/etc/localtime is a regular file while file
KC-20170403.ext3fs/etc/localtime is a symbolic link
```

There is a change to the time zone, categorised as a change of type 2.

```
$ ls -l KC-20161014.ext3fs/etc/localtime
-rw-r--r-- 1 root root 3519 Feb 26  2006 KC-20161014.ext3fs/etc/localtime
$ ls -l KC-20170403.ext3fs/etc/localtime
lrwxrwxrwx 1 root root 23 Apr  3 14:22 KC-20170403.ext3fs/etc/localtime ->
/usr/share/zoneinfo/UTC
$ diff KC-20161014.ext3fs/etc/localtime KC-20170403.ext3fs/usr/share/zoneinfo/UTC
```

```
Binary files KC-20161014.ext3fs/etc/localtime and
KC-20170403.ext3fs/usr/share/zoneinfo/UTC differ
```

In this change, the time zone data was changed from “posixrules” to “UTC”:

```
$ sha256sum KC-20161014.ext3fs/etc/localtime
KC-20161014.ext3fs/usr/share/zoneinfo/posixrules
KC-20170403.ext3fs/usr/share/zoneinfo/posixrules
5306deb14fbd5afeb22b2da69d2c165665deba82efc3bf642499f77b7e8a9d58
KC-20161014.ext3fs/etc/localtime
5306deb14fbd5afeb22b2da69d2c165665deba82efc3bf642499f77b7e8a9d58
KC-20161014.ext3fs/usr/share/zoneinfo/posixrules
5306deb14fbd5afeb22b2da69d2c165665deba82efc3bf642499f77b7e8a9d58
KC-20170403.ext3fs/usr/share/zoneinfo/posixrules
$ sha256sum KC-20161014.ext3fs/usr/share/zoneinfo/UTC
KC-20170403.ext3fs/usr/share/zoneinfo/UTC
ab1ddb33c7187e56408033a8165ea3b1432e024710e7ab2aafec036a8bd7f09a
KC-20161014.ext3fs/usr/share/zoneinfo/UTC
ab1ddb33c7187e56408033a8165ea3b1432e024710e7ab2aafec036a8bd7f09a
KC-20170403.ext3fs/usr/share/zoneinfo/UTC
$
```

The same posixrules time zone file appears in both filesystems, and the same UTC timezone file appears in both filesystems. No new data is introduced, but the time zone in /etc/localtime changes from a copy to a symlink, and changes from posixrules to UTC.

```
diff -r --no-dereference KC-20161014.ext3fs/etc/SERIAL
KC-20170403.ext3fs/etc/SERIAL
1c1
< Serial: ICANN-DNSSEC-KC-20161014
---
> Serial: ICANN-DNSSEC-KC-20170403
Only in KC-20170403.ext3fs/opt/icann/bin: hsmfd-hash
```

This is only a labelling change, categorised as a change of type 3.

```
Binary files KC-20161014.ext3fs/opt/icann/bin/ksrsigner and
KC-20170403.ext3fs/opt/icann/bin/ksrsigner differ
```

This is the essential change to the signing code, categorised as a change of type 1.

```
diff -r --no-dereference KC-20161014.ext3fs/opt/icann/bin/printlog
KC-20170403.ext3fs/opt/icann/bin/printlog
15c15,18
< top_margin=200
---
> ## --margins=left:right:top:bottom
> ## top_margin=200 #for ICANN header
> ##--margins=::${top_margin}: \
> ##--no-header \
22,25c25,28
```

```
<     --no-header \
<     --margins=::${top_margin}: \
<     --font=Courier10 \
<     --setpagedevice=Collate:true
---
>     --font=Courier7.5 \
>     --setpagedevice=Collate:true \
>     --header=$LOGFILE'||Page $% of $='
>
```

This change is part of the key management software changes, and hence categorised as a change of type 1.

```
Only in KC-20170403.ext3fs/opt/icann/dist:
icann-keytools-20170403.tar.gz
Only in KC-20170403.ext3fs/opt/icann/dist:
icann-keytools-20170403.tar.gz.sha256
```

These additional files are included as part of a change in how the key management software was updated, again categorised as a change of type 1.

Results

All observed differences were successfully categorised as one of the three types described above. All differences are considered to be either benign or directly related to the goal of making the new Root Zone Ceremony Key Tools available for use in a key ceremony.

Attestation

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

Snake Hill Labs Inc was retained by PTI under a commercial contract
to review source code changes relating to code used during root
zone DNSSEC key ceremonies.

The personnel involved in the analysis and technical assessment
described in this attestation were William Snow and Joseph Abley.

We were provided with two ISO images corresponding to DVDs used in
KSK ceremonies. The names of those images, together with their
respective SHA256 digests, are as follows:

$ sha256sum KC-20141014.iso | fmt
991f7be8cfbc3b4bdb6f5e5f84092486755a08a3c36712e37a26ccd808631692
KC-20161014.iso
$

$ sha256sum KC-20170403.iso | fmt
4d127c7db1a564399c0f4e00b34d6a7611e23cdb96cd64f3a428a16319285041
KC-20170403.iso
$

We analysed the differences between those two images. In every case,
```

the changes corresponded to documented and intended functionality,
as documented in the change log provided to us by PTI.

We found no differences that appeared to have any other purpose.

Our assessment is that the changes to the DVD images described above
are fit for the purpose described in the aforementioned change log.

This attestation has been signed using OpenPGP key 0xE1B9976C in
London, Ontario, Canada on Tuesday 18 April 2017.

Joseph Abley
Director, Principal Consultant
Snake Hill Labs Inc

-----BEGIN PGP SIGNATURE-----
Comment: GPGTools - https://gpgtools.org

iQIcBAEBCgAGBQJY9jeCAAoJEDu28RLhuZds0XQP/0ms/hbZtG7tDclfFKjOqfDe
RWwMZHufHcd+VxW6U67coZtCKOdE8pvKaHh0yqo/HYN8yAHBhhJP6XKfFCeM96qm
4UeNh2gQN+KOkkRhvvN8aIF7mNoWaNK0WGNbIuagJBh+rA4GM20cS7NBsN0JSzmf
iUfRURzbLTaalHpVrJc5MURM5ILGRIV28VIYMMJXS858ofjpg+UXlTCnlk/d7COK
kH+bf5CufcHctxqngGECkpdjfFfGA8DZ/Wa40Nu5uC+TYTenjKcPCmUZJq4i4z4O
7VO69mYPmdv+e3CxNa2UpMjkPh4rzaaHWbhrwCU2NX6IVfrZ/s/Cgfyvkjk6Hcu3
ZiBUeP0wkDFTI50butXQSaHPGGJV2mQ7oaScVvCoydbQytcf41h8AeLc03na9cs9
Mp/X0TbRZsqHCBL5b4VMnWnCZDpg7Zv3t6NhASEjmS3qBaRB4pSSqSb7slRVmsvg
gPPyzE0PQNSQk7b+CTRL0gKtBV/BLnsf+5cWEX0Xm/kPhn+L7ArK8v9cAyr5jR5e
obacP4A737jQ7C2SRq0PUYn3wY0K+yEiGLibkytBsfKJjWonD53jjivMIHymnUZi
/lE/vipm3YxRChpLerNphh2nAFsdEunimgrXLkUQhQTr2BbWCCjWlxJgilUiFl4n
jrFVOYy5R4pVJcSdpab5
=bmuw
-----END PGP SIGNATURE-----