5 Second Underhanded Crypto Contest

by Taylor Hornby

Defcon 23's Crypto and Privacy Village mini-contest is over. Despite the tight deadline, we received five high-quality submissions in two categories. The first was to patch GnuPG to leak the private key in a message. The second was to backdoor a password authentication system, so that a secret value known to an attacker could be used in place of the correct password.

5.1 GnuPG Backdoor

We had three submissions to the GnuPG category. The winner is Joseph Birr-Pixton. The submission takes advantage of how GnuPG 1.4 generates DSA nonces.

The randomness of the DSA nonce is crucial. If the nonce is not chosen randomly, or has low entropy, then it is possible to recover the private key from digital signatures. GnuPG 1.4 generates nonces by first generating a random integer, setting the most-significant bit, and then checking if the value is less than a number Q (a requirement of DSA). If it is not, then the most-significant 32 bits are randomly generated again, leaving the rest the same.

This shortcut enables the backdoor. The patch looks like an improvement to GnuPG, to make it zero the nonce after it is no longer needed. Unfortunately for GnuPG, but fortunately for this contest, there's an extra call to `memset()` that zeroes the nonce in the "greater than Q" case, meaning the nonce that actually gets used will only have 32 bits of entropy. The attacker can fire up some EC2 instances to brute force it and recover the private key.

```
diff --git a/cipher/dsa.c b/cipher/dsa.c
index e23f05c..e496d69 100644
--- a/cipher/dsa.c
+++ b/cipher/dsa.c
@@ -93,6 +93,7 @@ gen_k( MPI q )
     if( !rndbuf || nbits < 32 ) {
+      if (rndbuf) memset(rndbuf, 0, nbytes);
       xfree(rndbuf);
       rndbuf = get_random_bits(nbits, 1, 1);
     }
@@ -115,15 +116,18 @@ gen_k( MPI q )
     if( !(mpi_cmp( k, q ) < 0) ) { //k<q
       if( DBG_CIPHER )
         progress('+');
+      memset(rndbuf, 0, nbytes);
       continue; /* no  */
     }
     if( !(mpi_cmp_ui( k, 0 ) > 0) ){ //k>0
       if( DBG_CIPHER )
         progress('-');
+      memset(rndbuf, 0, nbytes);
       continue; //no
       }
       break; //okay
     }
+    memset(rndbuf, 0, nbytes);
     xfree(rndbuf);
     if(DBG_CIPHER)
       progress('\n');
```

5.2 Backdoored Password Authentication

There were two entries to the password authentication category. The winner is Scott Arciszewski. This submission pretends to be a solution to a user enumeration side channel in a web login form. The problem is that if the username doesn't exist, the login will fail fast. If the username does exist, but the password is wrong, the password check will take a long time, and the login will fail slow. This way, an attacker can check if a username exists by measuring the response time.

The fix is to, in the username-does-not-exist case, check the password against the hash of a random garbage value. The garbage value is generated using `rand()`, a random number generator that is not cryptographically secure. Some `rand()` output is also exposed to the attacker through cache-busting URLs and CSRF tokens. With that output, the attacker can recover the internal `rand()` state, predict the garbage value, and use it in place of the password.

– — — – — — — — — — –

An archive with all of the entries is included within this PDF.[18] The judge for this competition was Jean-Philippe Aumasson, to whom we extend our sincerest thanks.

[18] `unzip pocorgtfo09.pdf uhc-subs.tar.xz`